碩士學位論文

실내환경 제어를 위한 구동체 웹과 미들웨어 설계 및 구현

濟州大學校 大學院

컴퓨터工學科

陳　楠

2013 年 6 月

실내 환경 제어를 위한 구동체 웹과 미들웨어 설계 및 구현

指導教授 金 度 縣

陳　　楠

이 論文을 工學 碩士學位으로 提出함

2013年 07月

陳楠의 工學 碩士學位 論文을 認準함

審査委員長 _____ 印

委　　員 _____ 印

委　　員 _____ 印

濟州大學校 大學院

2013 年 07 月

Design and Implementation of Actuator web and Middleware For Controlling Indoor Environment

Nan Chen

(Supervised by professor Do-Hyeun Kim)

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

2013. 07

This thesis has been examined and approved

Thesis director, _____

Thesis director, _____

Thesis director, _____

July 2013

Department of Computer Engineering

Graduate School

Jeju National University

감사의 글

2011년 제주대 대학원 생활을 시작하여 어느덧 석사 학위과정을 마치고 학위를 받게 되었습니다. 2년 동안 열심히 했던 대학원 생활을 뒤 돌아보니 저에게 도움을 주셨던 모든 분들이 머리 속을 스쳐 갑니다.

먼전 제가 2년간 대학원 생활을 함에 있어 아낌없이 가르침과 지도를 해 주신 지도교수님 이신 김도현(金度縣) 교수님께 감사합니다. 또한 논문 심사에 애써주신 김장형 교수님과 이윤정 교수님에게도 감사합니다. 항상 대학원 생활에 세심한 지도와 가르침을 아끼지 않아주셨던 이상준 교수님, 변상용 교수님, 송왕철 교수님, 곽호영 교수님 감사드립니다. 그리고 대학원에 쉽게 적응할 있도록 도움을 주신 학과사무실 김남식 선생님, 오은희 선생님 감사드립니다.

대학원 같이 입학하여 동고동학한 선배 문유형, 선배 김현복, 친구 진서(陈曙) 그리고 파키스탄 친구 샆달 알리와 라시드가 항상 도움을 주어 감사드립니다. 제주대 모바일 컴퓨팅 연구실의 김주영, 강상훈, 김도연 함께한 시간이 많지는 않았지만 즐거웠고 모두 감사합니다. 특히 중국에 계신 어머님(郭小彦), 아버지(陳光木) 형님(陳國慶), 누나(陳紅萍), 형수님(丁亞麗)께서 원만하게 유학생활을 할 수 있도록 지지하여 주신 것에 정말 감사합니다. 그리고 제주대 정보통신학과의 가장 친하는 친구 양뢰(楊蕾)가 대학원 생활 동안 도움을 주어 즐거웠고 고맙습니다. 모두들 모든 일이 소망대로 되기를 바라고 건강하고 하는 일마다 축복이 있기를 기원합니다.

끝으로 부족한 저에게 언제나 힘이 되어주고 조언을 해준 친구 진경윤(程錦潤), 라해파(羅海波), 도비(杜飛), 포일비(包逸飛), 완건(汪建), 진서(陳曙), 림화(林華)에게 감사함을 전합니다.

목차

그림목차

표목차

국문초록

실내 환경 제어를 위한 구동체 웹과 미들웨어 설계 및 구현

컴퓨터공학과: 진 남

지도교수: 김도현

최근 물리 공간과 가상 공간을 연결하여 다양한 지능형 서비스를 제공하기 위해 CPS(Cyber Physical System), IoT(Internet of Things), WoT(Web of Things), 센서웹(Sensor Web) 등의 기술을 연구하고 있다. 특히 지도 상에 센서 네트워크에서 다양한 센서로부터 수집된 실시간 상황정보를 도시하는 센서웹의 연구가 OGC(Open Geospatial Consortium)를 중심으로 진행된다. 센서웹은 인공물과 자연물에 컴퓨터 기능을 갖는 다양한 센서를 설치하고 SOA(Service Oriented Architecture)라는 개념을 이용하여 실시간 상황을 웹상에서 관리한다. 최근 OGC에서는 야외뿐만 아니라 실내 센서웹 표준화에 관심을 집중하고 있다.

이와 더불어 최근 IoT 기술은 유럽, 중국, 미국, 일본, 한국 등 세계 주요 국가에서 전략산업으로 육성하고 있다. IoT는 모든 사물(Things)에게까지 네트워크 연결을 제공하는 네트워크의 네트워크를 의미하다. 현재 표준화된 통신 프로토콜에 기반한 독자적이면서 자체주소를 가진 상호 연결된 대상물들의 전세계(world-wide) 연결하고자 한다. 특히 유럽에서는 제7차 프레임워크 프로그램(FP7)에서 IoT 아키텍처, 통신 모델, 비즈니스 모델의 적용, 통합 및 시험 모델 구축 등을 진행하고 있다. IoT를 구축하기 위해서는 실시간 상황 정보를 수집하는 센서와 인터넷을 연결하여 감각뉴런(Sensory Neuron) 센서웹이 요구된다. 더불어 세상의 대상과 환경을 제어하여 지능적인 서비스를 제공하는 운동뉴런(Motor Neuron) 구동체웹이 필요하다. 물리공간과 가상공간을 연결한 센서웹과 구동체웹을 상호 연결한 연합뉴런(Inter Neuron)을 통해 공간, 사물, 인터넷 및 사람을 연계하는 IoT 시스템 개발이 필요하다.

이를 통하여 물리와 가상세상의 정보의 생산과 소비의 상호작용이 이루어져 인간에게 고도화된 지능적인 서비스를 제공할 수 있을 것으로 생각된다. 이에 본 논문에서는 Open API 기반의 실내 센서웹과 구동체웹의 구조를 제시하고 개발하고, 이를 토대로 서비스공급자 중심과 응용서버 중심의 IoT 구조를 제안하고 시스템을 설계하고 구현한다. 이를 위해 먼저 GIS 서버와 응용서버를 이용한 실내 센서웹을 개발한다. 실내 센서웹에서 서비스 공급자는 매우 다양한 실시간 상황 데이터를 제공하며 이를 관리한다. 그리고 GIS 서버를 두어 기존의 센서웹의 구조와 차별되게 서비스 공급자에게 동시에 센서노드 정보와 지도 정보를 제공하여 다른 시스템과 쉽게 연동할 수 있다. 센서웹 구조를 이용하여 스마트 홈에서 가전기구를 대상으로 실내 환경을 제어하는 구동체웹과 미들웨어를 설계하고 구현한다. 실내 구동체웹은 구동체, 미들웨어, 서비스공급자, 응용서버 간을 연결하고, 응용서버를 통해 구동체의 실제 동작 상태를 웹 상에서 확인하거나 제어할 수 있다. 더불어 실내 센서웹의 상황 데이터를 이용하고, 구동체웹의 구동체 제어 기능을 이용하여 실내 환경을 조절할 수 있는 IOT시스템을 설계하고 구현한다. 더불어 이를 위해 기존의 센서웹 기술, 구동체웹 기술 및 IOT 기술을 고찰하며, 서비스공급자 중심과 응용서버 중심의 센서웹, 구동체웹 및 IoT를 연동하는 환경을 구축하여 실험을 통하여 검증하여 성능을 평가한다.

그리고 이 연구를 통하여 수집되는 상황 데이터를 바탕으로 응용서버를 통해 다양한 사물을 제어할 수 있으며, 공간 및 시간에 따른 인간과 사물의 행위를 감시하고, 주위 상황에 따른 실내 환경의 변화의 실시간 인식하여 미래 인간의 생활의 편리성을 제공할 것으로 전망된다. 더불어 센서 기반 구체적 상황을 분석하고 데이터 시각화를 통한 인간의 의사결정을 지원하고, 공장, 빌딩, 가정 등 실내 환경에서의 자동 제어를 통해 자율 최적화를 지원하는 데 활용될 수 있다.

ABSTRACT

Design and Implementation of Actuator Web and Middleware for Controlling Indoor Environment

Chen Nan

Department of Computer Engineering

Graduate School

Jeju National University

Recently, through connecting physical and virtual space, the technology such as CPS (Cyber Physical System), IoT (Internet of Things), WoT (Web of Things), Sensor Web etc, has been researched for providing the various intelligent services. In particular, research of sensor web that indicates the status of real-time context information collected from various sensor of sensor network on the map is advanced around OGC (Open Geospatial Consortium). Sensor web installs various sensors that has a computer function on artifact and natural object, and manages real-time context on web using conception of SOA(Service Oriented Architecture). Recently, OGC researcher interests not only on outdoor, but also indoor sensor web standardization.

IoT means network of network that provides a connection to every things. In addition, recent IoT technology is developed as a strategic industry in major countries of the world such as Europe, China, America, Japan, and South Korea. The aim of IOT is that the interconnected objects, which have these own addresses based on standardized communication protocol, connect to the world-wide. In particular, IoT architecture, communication model, appliance of business model, mutual and test model construction etc.. are advanced on the 7th framework program(FP7) in Europe. Sensory Neuron sensor web is required by connecting internet and sensor that collects real-

time context information for constructing the IoT. Motor Neuron actuator web that provides intelligent service by controlling an object and environment of the world is needed. IoT system development, that links space, object, internet and person through Inter Neuron which interconnects actuator web and sensor web connecting physical and virtual space.

Though this, It is believed possible that we can provide intelligent service to humans through the interaction of the production and consumption of information in the physical and virtual worlds. In this work, we present and develop the structure of indoor sensor web and actuator web of Open API, and we propose IoT structure around service provider and application server, and finally, we design and implement this system. For this, we develop indoor sensor web using GIS server and application server. Service provider provides the various real-time context data and indoor sensor web manages this data. Then, it is possible to work easily with other systems by providing sensor node and map information to service provider to place a GIS server so as to be differentiated from structure of the sensor network existing. We design and implement middle-ware and actuator web that controls indoor environment to target appliance in smart home using sensor web structure. Indoor actuator web connects actuator, middle-ware, and service provider. The application server can verify and control the actual work status of actuator on web through application server. In addition, using context data of indoor sensor web and actuator control function of Actuator Web. We design and implement IOT system that can adjust indoor environment. For this, we consider IOT, actuator web and existing sensor web technology, and construct environment that link sensor web, actuator web and IoT around service provider and application server, and evaluate the performance.

Through this research, it is expected that we can control various things through the application server based on context data collected, and monitor the behavior of people and objects according to space and time, and provide convenience to human by recognizing a change of indoor environment in

accordance with the ambient conditions in real time. It can also be used to analyze specific context based on sensor and supporting decision of human through visualization of data, and supporting autonomy optimization through auto control in indoor environment such as factory, building, and home.

약어표

GIS	Geographic Information System
COST	Contests Supporting Tool
OGC	Open Geospatial Consortium
SWE	Sensor Web Enablement
SOA	Service Oriented Architecture
WCF	Windows Communication Foundation
SOAP	Simple Object Access Protocol
PMV	Predicted Mean Vote
ET	Effective Temperature
IoT	Internet of Things
REST	Representational State Transfer
URI	Uniform Resource Identifier
WSDL	Web Service Description Language

I. 서론

1. 연구 배경

현재 국내외 연구 기관이 다양한 공간 정보, 환경 정보를 제공하는 센서웹 개발에 전념하고 있다. 센서는 점차 인간의 일상 생활 주변에 빈번하게 나타난다. 예를 들어 온도, 습도, 조도, GPS, 압력 등 다양한 센서의 응용은 지능화 산업 생산, 관리 분야에 확대되고 있다. 그리고 네트워크 연결을 통해 원격 모니터링 및 제어하는 가전제품을 많이 개발하고 있다. 그런데 급속한 산업발전으로 인해 환경문제 및 지구 자원부족현상이 나타나고 있다. 그래서 인류는 지능화 기술을 이용하여 환경과 에너지 소비를 최소화 하면서 생활의 편리성을 제공해야 한다.

21 세기 "ubiquitous 세계"에 대해 많은 국가 및 관련 국제기구가 점점 관심을 갖게 되고 유비쿼터스 네트워크 응용 프로그램과 서비스는 많은 분야에서 자동화를 향상시키고 혁명적인 변화를 가져오고 있다. 예를 들어 물류, 환경 보호, 가정 네트워크, 의료보건, 지능빌딩 등의 분야에서 자동화 및 정보화가 진행되고 있다. 한국은 다양한 사회적인 USN 응용을 기반으로 각 각 산업 및 전체 도시의 정보화 수준을 향상 시키는 USN 프로젝트를 가동하고 있고, 많은 도시에서 U-City를 발전시키고 있다. U-City 서비스는 U-Home(U가정), U-ITS(U지능교통), U-FMS(U종합교통 관리시스템), U-Monitoring(U감시)을 포함한다.

따라서 본 논문에서는 센서웹 시스템과 구동체웹 시스템의 연동을 통해 실내의 환경을 자동적으로 쾌적상태를 유지하는 방법을 제시한다. 기존 센서 웹 구조는 센서정보 및 지도정보를 같이 제공하기 때문에 지도정보의 관리가 불편하다. 그래서 센서 웹과 구동체웹이 같은 GIS를 사용할 수 있도록 분리한다. 센서웹 저작도구는 단순히 센서정보를 등록하며, 구동체웹 저작도구는 구동체 정보를 등록한다. GIS 서비스 저작도구는 지도 서비스 콘텐츠를 생성하는 역할을 담당한다. 이 방법으로 각 시스템의 부담을 줄일 수 있다.

제시된 구동체웹 시스템, 센서웹 시스템, 응용서버 기반의 융합서비스를 통해 실내 공간의 환경 정보에 의해 가전 제품을 제어하여 실내 환경을 유지한다. 본

논문에서 제시하는 GIS 분리 공용 구조로 스마트 공간 서비스 환경을 구축하여 검증한다.

2. 연구목적 및 방법

센서웹은 인공물과 자연물에 컴퓨터 기능을 갖는 다양한 센서를 설치하고 이를 웹으로 연결시켜 시설물 및 환경, 교통 상태, 재난재해 등을 모니터링 하는 개념이다. 기존의 센서웹은 서비스 등록자, 서비스 공급자, 클라이언트는 모듈로 구성된다. 서비스 공급자가 다양한 센서 네트워크로부터 실시간으로 수집되는 상황 데이터를 통합하고 처리한 정보를 클라이언트에게 제공하는 역할을 수행한다. 서비스 공급자가 센서 정보, 지도 정보를 관리하며 서비스 정보를 등록 관리하는 역할을 담당하며 동시에 서비스를 공급하고 서비스 콘텐츠 생성 및 관리하는 역할을 담당해야 하므로 맞춤형 콘텐츠를 생성하고 유지관리하기가 어렵다.

COST기반의 센서웹 시스템은 서비스 콘텐츠의 생성 및 관리하는 역할을 서비스 공급자에게서 분리하고 COST라는 모듈이 이 기능을 수행한다. COST기반의 센서웹 시스템 구조를 사용하면 서비스 공급자의 부담을 상당히 줄인다. 그런데 본 논문에서 제시하는 센서웹 시스템과 구동체웹 시스템을 연동하도록 하기 위해 지도 서비스를 더 분리하여 두 시스템은 한 지도 서비스를 사용한다.

따라서 본 논문에서 센서웹 저작도구는 단순히 센서정보를 등록하며, 구동체웹 저작도구는 구동체 정보를 등록한다. GIS 서비스 저작도구는 지도 서비스 콘텐츠를 생성하는 역할을 담당한다. 센서웹 공급자가 실내 환경 데이터를 수집한다. 구동체웹 공급자는 구동체를 제어 및 모니터링하는 인터페이스를 제공한다. 응용서버 저작도구는 센서 노드, 구동체노드의 위치 정보를 생성하여 응용서버에 전송한다. 응용서버는 환경 데이터에 따라서 실내 가전 제품을 자동 제어하는 지능형 시스템의 구축방안에 대해 제시하고 구현한다.

3. 논문 구성

서론에 이어 2장 관련 연구에서는 SOA의 분석과 기본적인 아키텍처를 고찰하고, 센서웹에 대한 분석과 해외사례 조사, WCF 기반의 Open API 구조에 대해 고찰한다. 3장에서 개방형 API 기반의 센서웹 시스템을 설계, 구현 및 성능 평가한다. 4장에서 개방형 API 기반의 구동체웹 시스템을 설계, 구현 및 성능 평가한다. 5장에서 저작도구 기반의 IOT 시스템을 설계, 구현 및 성능 평가하였고 6장는 실내 GIS 시스템을 설계, 구현 및 성능 평가한다. 마지막 7장에서는 결론을 맺는다.

II. 관련연구

1. 웹서비스 기술

2000년대 초반부터 비즈니스 도메인의 서비스를 웹서비스로 개방하여 상호 연동하는 방안이 시도되어 왔다. 최근에는 이러한 시도가 통신 및 인터넷 서비스 전 분야의 개방에 큰 파급 효과를 미치고 있다. SOAP 기반의 웹서비스는 비즈니스 환경에서 응용 서비스들의 상호 연동을 위하여 시작된 반면에 RESTful 웹서비스는 인터넷 서비스 업체들이 응용 개발자들에게 손쉽게 데이터를 제공하기 위한 목적으로 출발한다. 개발측면에서 SOAP 기반의 웹서비스는 서비스를 제공하고 이용하는 프로그램들을 위하여 W3C의 엄격한 표준과 잘 갖추어진 인프라에 의하여 개발되고 있는 반면, RESTful 웹서비스는 기계 보다는 사람이 이해하기 쉽도록 HTTP 및 XML과 같은 인터넷의 기본 표준만을 요구한다. 이러한 단순성과 개발의 용이성으로 구글, 야후, 트위터 등 대부분의 웹 2.0 API가 RESTful 웹서비스로 제공되고 있다[5].

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  ...
  ...
</soap:Header>

<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

그림 1 SOAP 메시지 구조

SOAP(Simple 오브젝트 Access Protocol)은 간단한 개체 방문 프로토콜이다. 간단하다는 것은 SOAP이 현재 IT분야에서 범용하고 있는 두 가지 프로토콜 HTTP와 XML을 기반으로 개발된 프로토콜 이기 때문이다. 그래서 이 기술은 사실은 “아무 새 기술도 발명하지 않은 기술이다.” 라고 할 수 있다. 여기서 개체라는 것은 SOAP를 통해 방문한 웹서비스 대상이다. SOAP은 서비스와 관련된 속성

과 참조행위가 WSDL을 통해 기술된다. SOAP은 통일된 표준 기법을 사용하여 서로 다른 운영체제 및 다른 프로그래밍 기술로 개발된 응용 사이에 통신기능을 제공한다. 그림 1는 SOAP의 메시지 구조이다.

REST(Representational State Transfer)는 구조적인 네트워크 시스템을 기술한다. “Web 응용. REST”개념은 최초 Roy Fielding 박사논문에서 제안하게 된다.

REST는 한 세트의 구성 제한 조건이 원칙이다. 이런 조건, 제한을 만족하는 응용 프로그램이나 설계는 RESTful 이다.

Web 응용에 대한 가장 중요한 REST원칙은 클라이언트와 서버 사이의 요청과 응답이 없는 상태이다. 클라이언트로부터 서버에 가는 요청마다 요청을 이해하기 위한 필수 정보를 포함한다. 서버가 요청할 때, 임의 시간에 재부팅 하면 클라이언트에게 알려주지 않는다. 그래서 무상태 요청은 다른 가용 서버가 대신 응답 가능하다. 이 특성은 클라우드컴퓨팅 같은 환경에 자주 적용된다. 서버의 응용 프로그램의 상태 및 기능은 여러 가지 자원으로 나눌 수 있다. 각 각의 자원은 URI를 통해 유일한 주소를 얻는다.

REST는 개발자에게 이해가 쉽고 모든 HTTP/HTTPS를 지원하는 클라이언트/서버가 REST의 사용이 가능하다. 그런데 SOAP 같은 경우는 모든 통신 프로토콜(TCP, SMTP 등)을 사용할 수 있다. 표 1은 SOAP과 REST의 적용환경 비교이다.

표 1 적용환경 비교

REST 적용 환경	SOAP 적용 환경
유한 대역폭과 자원	비동기 처리 및 호출
완전히 무상태 작업	포맷 계약화
버퍼요소	유상태 작업

2. 실내 쾌적지수 및 상태정보

PMV는 Predicted Mean Vote이다. 즉, 예측 평균투표수 PMV는 인간의 열감에 대한 평가 지수이다. 대부분 사람의 온도 감지의 평균수준이란 뜻이다. PMV는 인체의 열 항상성을 기반으로 사람의 7가지 열감을 평가하는 지수이다. 인체

열 항상성은 사람이 외부 환경에서 받은 열에너지와 인체 내부에 생성된 열에너지의 종합결과이다. 안정된 환경에서, 인체의 체온 조절 시스템은 자동적으로 피부온도와 땀샘분비의 조절을 통해서 인체의 열을 일정수준으로 유지한다 (ISO_7730_2005 표준).

PMV 수식:

$$\begin{aligned}\mathrm{PMV} = {} & [0.303 \cdot \exp(-0.036 \cdot \mathrm{M}) + 0.0028] \\ & \cdot \{(\mathrm{M} - \mathrm{W}) - 3.05 \cdot 10^{-3} \cdot [5733 - 6.99 \cdot (\mathrm{M} - \mathrm{W}) - \mathrm{Pa}] - 0.42 \\ & \cdot [(\mathrm{M} - \mathrm{W}) - 58.15] - 1.7 \cdot 10^{-5} \cdot \mathrm{M} \cdot (5867 - \mathrm{Pa}) - 0.0014 \cdot \mathrm{M} \cdot (34 - \mathrm{ta}) \\ & - 3.96 \cdot 10^{-8} \cdot \mathrm{f_{cl}} \cdot [(\mathrm{t_{cl}} + 273)^4 - (\bar{\mathrm{t}}_\mathrm{r} + 243)^4] + \mathrm{f_{cl}} \cdot \mathrm{h_c} \cdot (\mathrm{t_{cl}} - \mathrm{t_a})\}\end{aligned}$$

수식 1

$$\begin{aligned}\mathrm{t_{cl}} = {} & 35.7 - 0.024 \cdot (\mathrm{M} - \mathrm{W}) - \mathrm{I_{cl}} \\ & \cdot \{3.96 \cdot 10^{-8} \cdot \mathrm{f_{cl}} \cdot [(\mathrm{t_{cl}} + 274)^4 - (\bar{\mathrm{t}}_\mathrm{r} + 243)^4] + \mathrm{f_{cl}} \cdot \mathrm{h_c} \cdot (\mathrm{t_{cl}} - \mathrm{t_a})\}\end{aligned}$$

수식 2

$$\mathrm{h_c} = \begin{cases} 2.38 \cdot |\mathrm{t_{cl}} - \mathrm{t_a}|^{0.25} & \text{if } 2.38 \cdot |\mathrm{t_{cl}} - \mathrm{t_a}|^{0.25} > 12.1 \cdot \sqrt{\mathrm{v_{ar}}} \\ 12.1 \cdot \sqrt{\mathrm{v_{ar}}} & \text{if } 2.38 \cdot |\mathrm{t_{cl}} - \mathrm{t_a}|^{0.25} > 12.1 \cdot \sqrt{\mathrm{v_{ar}}} \end{cases}$$

수식 3

$$\mathrm{f_{cl}} = \begin{cases} 1.00 + 1.290 \cdot \mathrm{I_{cl}} & \text{if } \mathrm{I_{cl}} \le 0.078 \mathrm{m^2} \cdot \mathrm{K/W} \\ 1.05 + 0.645 \cdot \mathrm{I_{cl}} & \text{if } \mathrm{I_{cl}} > 0.078 \mathrm{m^2} \cdot \mathrm{K/W} \end{cases}$$

수식 4

수식 1부터 수식 4까지는 PMV지수를 계산하는 식이다. 이 수식들에서 각 파라미터 의미는 다음과 같다. 먼저 M은 대사율이고 단위는 $\mathrm{W/m^2}$이다. W는 유효 기계 에너지로서 단위는 $\mathrm{W/m^2}$이다. $\mathrm{I_{cl}}$은 옷의 전열능력으로 단위는 $\mathrm{m^2 * K/W}$이다. $\mathrm{f_{cl}}$은 옷의 표면적 영향 지수 이다. $\mathrm{t_a}$는 공기의 온도이고 $\bar{\mathrm{t}}_\mathrm{r}$은 평균복사온도이며 단위는 °C이다. $\mathrm{v_{ar}}$은 상대 공기 속도이고 단위는 m/s이다. $\mathrm{P_a}$는 수증기의 압력으로 단위는 $\mathrm{P_a}$이다. $\mathrm{h_c}$는 대류열전달이며 단위는 $\mathrm{W/(m^2 * K)}$이다. $\mathrm{t_{cl}}$은 옷의 표면온도이고 단위는 °C이다.

표 2 PMV 지수 상태 룰

PMV Index	State
+ 2.5<PMV≤+ 3	Hot
+ 1.5<PMV≤+ 2.5	Warm
+ 0.5< PMV≤+ 1.5	Slightly warm
-0.5≤PMV≤+ 0.5	Neutral
-1.5≤PMV<-0.5	Slightly cool
-2.5≤PMV<-1.5	Cool
-3≤PMV<-2.5	Cold

PMV값은 같은 환경에 있는 사람들의 모든 신체 내외의 열 교환현황 지수이다. 이 지수의 기능으로 사람의 열감을 예측할 수 있다. 예를 들어 너무 덥거나 추울 때 PPD는 어떤 PMV값이 나올 때 너무 덥거나 너무 추워서 불쾌감을 느끼는 평가지수이다. 국제 기준은 현 환경에 대해 만족하는 사람들은 찬성하고 불만인 사람은 반대투표를 하여 최종의 투표 비가 바로 PPD값이다(ISO_7730_2005 표준).

PPD 수식:

$$\mathrm{PPD} = 100 - 95 * \exp(-0.03353 * \mathrm{PMV}^4 - 0.2179 * \mathrm{PMV}^2)$$

수식 5

수식 5는 PMV지수를 이용해서 PPD지수를 계산하는 것이다. PPD는 PMV지수에 대해 선형관계를 나타난다.

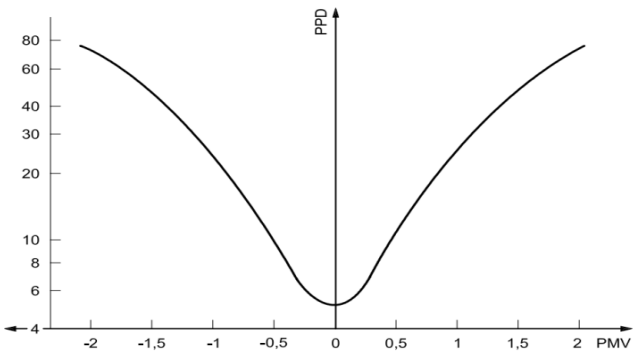

그림 2 PMV과 PPD의 선형관계

그림 2는 PMV지수, PPD지수의 관계이다. PMV가 0일 때 PPD는 가장 작으며 0에서 양쪽으로 멀어지면 PPD지수는 커진다.

표 3 PPD 지수 상태 룰

PPD Index(%)	State
5<PPD≤10	Comfortable
10<PPD≤51	A little Comfortable
51<PPD≤93	A little Comfortable
91<PPD≤100	Uncomfortable

ET(Effective Temperature)는 인체 환경의 온도에 대한 감지이다. 주로 풍속, 습도, 태양열 등의 요소를 고려한다. ET는 공기온도보다 높으면 위의 요소 때문에 인체에 대한 가온 작용이라고 하며 그 반대가 강온 작용이다. 공기온도는 공기의 냉난방 정도만 표현함으로 주변환경에 대한 느낌을 표시할 수 없다. 그러나 체감온도는 인체의 실제 느낌을 판단하는 상수이다. 이것이 바로 체감온도와 공기온도의 차이점이다. 같은 공기온도 조건에서 사람은 환경의 습도, 풍속, 옷의 색깔 그리고 태양광 강약 등의 요소 때문에 느낌이 다를 수 있다.

ET수식:

$$T_m = 37 - \frac{37 - t}{0.68 - 0.0014h + \frac{1}{A}} - 0.29t \times \left(1 - \frac{h}{100}\right)$$

$$A = 1.76 + 1.4v^{0.75}$$

수식 6

수식 6은 환경변수를 이용하여 ET지수를 계산하는 수식이다. t는 실내공기 온도이다. h는 실내 공기 습도이다.

표 4 ET 지수 상태 룰

ET지수(c°)	State
4<ET≤8	Hot
8<ET≤13	Warm
13< ET≤18	Slightly warm
18≤ET≤23	Neutral
23≤ET<29	Slightly cool
29≤ET<35	Cool
35≤ET<41	Cold

3. 기존 구동체 제어 미들웨어

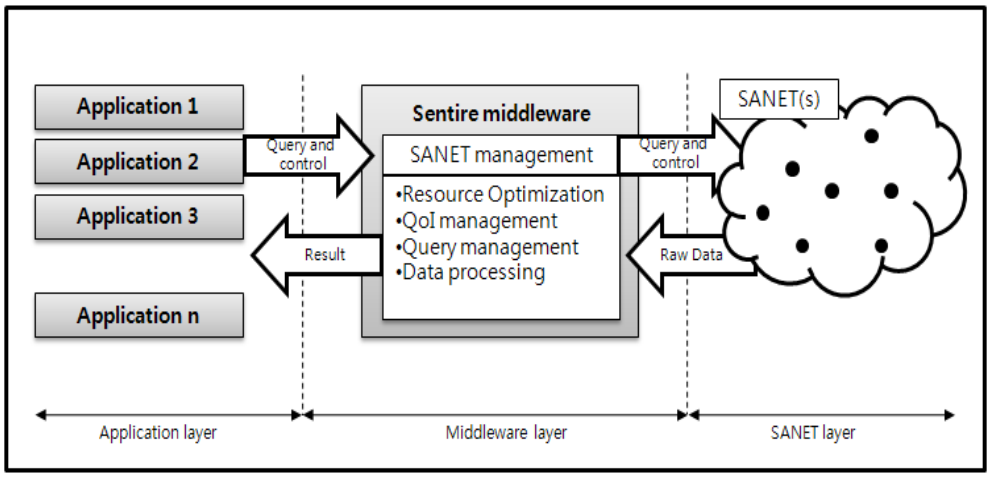

그림 3 SANET 시스템 구조

위의 그림 3은 센서와 구동체간의 네트워크인 SANET과 센싱 데이터 수집과 구동체 제어를 담당하는 미들웨어인 Sentire 미들웨어의 시스템 구조를 나타낸다. Sentire 미들웨어는 응용의 질의와 제어 메시지를 받아 SANET의 구동체를 제어하거나 센서로부터 센싱 데이터를 가져오게 된다. 여기서 SANET 시스템의 미들웨어는 장치의 전력량과 채널의 대역폭을 관리하는 자원관리(Resource management), 센서를 이용한 시스템에 대한 효율적인 질의 처리(Query management), 센싱된 데이터로부터 중요한 특징을 알아내는 처리 과정인 센서 데이터 처리(센서 data processing), 규정된 수준 이상의 데이터 질을 제공하기 위해 센서를 구성하는 것을 관리하는 정보의 질 관리(Quality of information management, QoI), 환경의 규정이나 외부의 응답을 제공하는 구동체의 관리를

담당하는 구동체 관리(Actuator management) 기능들을 고려해야한다. 이와 더불어 SANET 시스템의 미들웨어는 일반적인 컴퓨팅 패러다임인 상호 작동 가능성(Interoperability), 재활용성(Reusability), 확장성(Extensibility), 적응성(Adaptability)을 모두 가져야 한다[16].

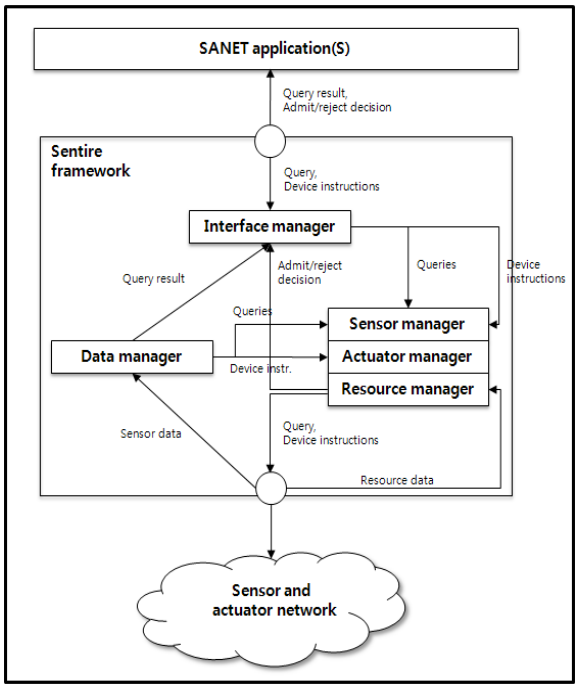

그림 4 Sentire 프레임워크 데이터 흐름도

위 그림 4는 센서와 구동체 네트워크와 SANET 응용 사이에서 Sentire 프레임워크의 주요 관리 기능과 데이터의 흐름을 나타낸다. SANET 응용과의 미들웨어의 연결을 담당하는 Interface manager(IM), 개발자가 정해놓은 루틴으로 데이터 처리 기능을 수행하는 Data manager(DM), 센서와 구동체의 자원 관리를 담당하는 Resourse manager(RM), 센서, Actuator들의 구성과 동작 방식을 제어하는 센서 manager(SM)와 Actuator manager(AM) 컴포넌트는 Sentire 프레임워크에서 핵심적인 역할을 한다. 이중 Actuator manager는 SANET 응용이나 데이터의 관리자로부터 보내지는 장치 명령어(Device Instruction)를 받아 들여 구동체를 제어한다. Sentire 미들웨어에서 각 컴포넌트 간 데이터 통신을 위해 사용하는 메시지의 헤더에는 아래 그림 5와 같이 효율적인 처리를 위해 priority, time

stamp, sending manager 정보가 들어 있다. 또한 SANET 응용에서 보내는 Sentire Query 메시지와 Data 메시지는 비슷한 구조를 가지며 모두 표준 방식인 XML 문서 객체 모델(Document Object Model: DOM)를 사용하여 만들어 진다 [15].

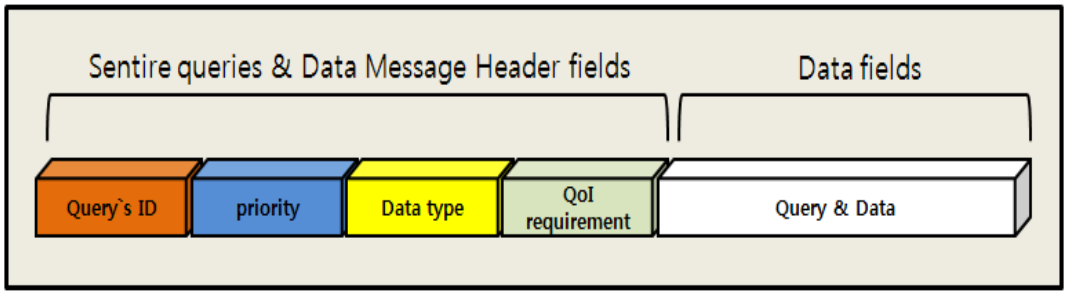

그림 5 Sentire 미들웨어의 쿼리 패킷과 데이터 패킷의 구조

4. IoT 기술

지금까지 인터넷을 통해 유통하는 정보는 인간이 만들어 낸 데이터를 가공한 지식과 주를 이루었다. 하지만, 미래의 인터넷에서는 주변에 존재하는 일상 사물들 자체가 유일한 식별자를 지니며 센싱과 액츄에이션 기능을 통해 세상과 상호작용을 하며, 컴퓨팅, 통신 기능을 통해 정보처리 및 인터넷에 연결됨으로써 인간만으로 국한되었던 정보의 생산/소비자의 역할을 함께 할 것이다. 이를 일컬어, 사물 인터넷(Internet of Things: IoT)이라 하며 이는 미래에 새로운 차원의 응용과 서비스, 그리고 시장을 창출할 기술로 여겨지고 있다.

IoT를 통해 사람과 사람, 사람과 사물, 사물과 사물간의 통신과 상호작용, 그리고 정보 공유가 가능해지면, 인간은 고해상도의 물리적 환경정보를 얻음으로써 자신의 행동을 더욱 최적화할 수 있으며, 또한 시스템 스스로가 상황판단을 정확하게 하도록 하여 사람의 개입 없이 동작을 자율적으로 수행할 수 있게 된다[18]. 특히, 최근 스마트폰이 소프트웨어의 새로운 생태계를 창출한 것처럼, IoT에서는 일상의 사물이 지능화되어 인터넷의 참여자가 되어 기존에 얻지 못했던 새로운 차원의 정보를 제공함으로써 새로운 서비스 생태계 형성에 크게 기여할 것으로 예상되고 있다[17]. 그러나, 수조개까지 이를 수많은 사물들 각각을 식별하고 인지하며 이들과 상호작용을 함은 물론 이들이 생산해내는 엄청난 양의 데이터를 인터넷에 연결하여 그 정보를 가공하고 공유하는 것은 현재의 인터넷과 컴퓨팅

기술로는 어려울 것으로 예측되고 있다. 현재 이러한 어려움을 극복하고자 하는 노력은 전 세계적으로 이루어지고 있으며, 특히, 유럽, 중국, 한국, 미국 등의 국가에서 선점해야 할 주요기술로 선정하여 국가적 차원에서 집중투자와 연구를 진행하고 있다.

IoT기술은 EU, 중국, 미국, 일본, 한국 등 세계 주요 국가에서 전략상업으로 육성하기 위해 범국가적인 지원 정책을 수립, 추진하고 있는 연구 분야로써, 본 장에서는 각 국가별 IoT 관련 수행 프로젝트 및 연구 동향에 관하여 살펴보도록 한다.

표 5 2010년 수행 중인 Internet of Things(IoT) 관련 EU 프로젝트

과제명	연구내용	참여기관	수행기간
GASAGRAS2	물리적 객체와 가상 객체를 연결하는 글로벌 네트워크 기반 구조를 연구하며, 브라질, 중국, 한국, 미국 등과의 국제 협력 체계를 구축	AIM UK, ETRI, UoB, Yokosuka	2010.06~2012.05
CONET	임베디드 시스템, 퍼베이시브 컴퓨팅, 센서 네트워크로 분리된 커뮤니티를 하나로 통합하고 연구자간의 협동할 수 있는 환경을 구축	SAP AG, BOEING, Schneider	2008.06~2012.05
EBBITS	IoT기술과 비즈니스 모델과의 통합을 위한 공간/사물/사용자의 상황인지 및 처리기술 연구	SAP AG, TNM, CNET, IN-JET	2010.09~2014.08
ELLIOT	KSB(Knowledge-Social_Business) 모델 기반의 IoT 통합 실험 모델 개발 및 플랫폼 구축	Polymedia, Fing, INRIA	2010.09~2012.02
IoT-A	IoT 단말, 서비스 간의 상호호환성 제공을 위한 IoT 아키텍처 및 통합 플랫폼기술 연구	VDI/VDE, IBM, Hitachi, NEC	2010.09~2013.08
IoT-I	여러 단위로 분리되어 연구되어 왔던 IoT 기술들의 통합 기술 연구 및 통합 연구 환경 구축	U. of Surrey, Ericsson, NEC	2010.09~2012.08
IOT@WORK	IoT 솔루션의 생산, 설정, 커미셔닝 등의 자동화를 위한 Plug&Work기술 및 IoT 솔루션과 네트워크의 디커플링 기술 연구	Siemens, EMIC, CRF, CRS TXT	2010.06~2013.05
SMART SANTANDER	IoT 기술 및 응용의 통합 및 테스트를 위한 도시 단위의 테스트베드 구축	Santander, U. of Melbourne	2010.09~2013.08
SPRINT	IoT를 위한 COTS 기반의 소프트웨어 프랫폼 통합 기술 연구	EADS UK, IBM ISRAEL, Elvior	2010.10~2013.09

EU는 2007년부터 2013년까지 수행되는 제7차 프레임워크 프로그램(FP7)에서 정보통신기술(ICT: Information and Communication Technologies) 분야에

FP7 전체 총액의 64%에 이르는 91억 유로를 투자하고 있으며[17], EU의 ICT 분야는 크게 네트워크 및 서비스 인트라, 로보틱스, 컴포넌트 및 시스템 등 총 8개의 도전과제(Challenge)로 구성된다. IoT연구 트랙은 도전과제 1(Pervasive and Trust 네트워크 and Service Infrastructures)의 7개 세부과제 중 하나이며, 90년대부터 2004년까지 D4(Networked Enterprise and RFID) 연구를 담당하였던 RFID 연구 그룹이 주도하고 있다. IoT 연구 트랙에서는 2007년부터 총 27개의 프로젝트가 수행 완료 또는 수행 중에 있으면, FInES(Future Internet Enterprise Systems) 및 IERC(Internet of Things European Research Cluster)와 같은 클러스터링 프로젝트(Clustering Project)가 개별 프로젝트 간 협력 및 총괄을 담당한다. 표 5는 2010년 수행 중인 IoT 관련 EU 프로젝트들이며, IoT 아키텍처, 통신 모델, 비즈니스 모델의 적용, 통합 및 시험 모델 구축 등 다양한 분야에서의 IoT 연구가 진행되고 있다. 특히, 2011년 10월부터 시작하게 될 IERC의 IoT6 프로젝트에서는 IoT를 실현하기 위한 기반 네트워크로서 IPv6를 선택하고 EPC 네트워크를 포함한 다양한 이질적 네트워크를 통합하기 위하여 서비스 지향 아키텍처를 연구할 예정이다. IoT6에는 스웨덴, 프랑스, 영국, 독일 등 유럽 내의 대학과 연구 기관, 기업을 주축으로 컨소시엄이 구성되며, 한국에서는 KAIST가 참여할 예정이다.

중국에서는 2011년부터 2015년까지 수행되는 제 12차 중국 5개년 연구 계획을 통하여 IoT인프라 및 응용 시스템 구축에 매년 400억 위해 자금을 투입한다. 중국 5개년 연구 계획은 크게 센싱 및 프로세싱을 담당하는 인지 계층(Perception Layer), 통신 인프라를 제공하는 통신 계층(Communication Layer), 응용 서비스를 제공하는 응용 계층(응용 Layer)으로 구성되며, 인지 계층에서는 IC, 센서/MEMS, RFID, 임베디드 소프트웨어와 같은 센싱/프로세싱 기술, 통신 계층에서는 3G/4G 이동통신망과 IPv6 기반의 무선센서 네트워크, RFID/USN 통합 기술, 응용 계층에서는 미들웨어를 포함한 클라우드 컴퓨팅, 스마트 그리드(Smart Grid), 지능형 교통망(Smart Transportation) 등의 응용 시스템 기술 연구를 수행한다.

미국의 국가정보위원회 (NIC)는 2025년까지 국가경쟁력에 영향을 미치는 6대 기술중 하나로 IoT를 선정하고 IoT 기술 개발에 총력을 다하고 있으며, 일본에서는 UID(Ubiquitous ID) 센터를 중심으로 uCode라 불리는 독자적인 전파 식

별 코드, 무선 통신 기술, 데이터베이스 처리 기술 등 IoT에 필요한 요소 기술들을 연구하고 있다 [17]. 한국은 2005년부터 USN(Ubiquitous 센서 네트워크) 시범사업, u-City 사업 등을 통해 지역별 서비스 인프라 구축을 위한 기술 연구를 진행해 왔으며, 2009년 방송통신위원회가'사물통신(M2M : Machine-to-Machine) 기반 구축 기본계획'[17]을 발표하면서 IoT와 같은 글로벌 인프라 구축 기술에 관한 연구가 활발히 이루어지게 된다. 현재 사물통신의 통신 인프라를 제공할 수 있는 KT, SKT 등의 이동통신사업자들의 주도로 사업계획이 진행되고 있으며, 이에 따라 광대역 통합망(BcN), IPv6, 2G/3G 이동통신 서비스와 같은 네트워크 인프라에 대한 구축 및 기술 연구가 활발히 이루어지고 있다.

IoT의 사물들은 다수의 이질적인 네트워크로 연결되며 그 인터페이스도 다양하다. 마찬가지로 소프트웨어 관점에서도 IoT는 다양한 소프트웨어 기술들을 요구한다. 본고에서는 그들 중 중요한 기술이라 판단되는 기술로서 사물의 형태에 따라 필요한 다양한 운영 체제, 사물의 정보를 명세하기 위한 메타데이터, 사물을 웹을 통해 인터넷에 연결하는 사물의 웹(Web of Things: WoT) 기술들에 대하여 살펴본다. 먼저, IoT의 주요 구성요소인 센서노드와 같은 자원제약적인 장치에서 구동되는 운영체제를 살펴보면 이벤트 구동방식의 TinyOS 와 멀티 스레드를 지원하는 Contiki 운영제체가 대표적이며, 최근에는 Sun Microsystems에서 Squawk 가상머신을 사용하여 Java기반으로 센서노드를 개발한다. 이와 더불어 비교적 자원이 풍부한 가전제품과 같은 장치에서 운용이 될 수 있는 운영체제로 최근 구글의 오픈소스 소프트웨어 플랫폼인 Android가 주목을 받고 있다. 구글은 Android@Home 프로젝트에서 오픈 액세서리 개발 플랫폼(Open Accessory Developmentplatform)을 지원하여 현재의 스마트폰, 태블릿에 국한하지 않고 일상의 가전제품까지도 Android가 운용될 수 있는 환경을 제공하고 있다.

IoT에서는 사물 자체 정보 및 사물의 상태를 컴퓨터 혹은 사람이 쉽게 이해할 수 있도록 하는 메타데이터 혹은 소프트웨어 모델이 필요하다. OMG(오브젝트 Management Group)에서는 일상의 사물에서 가전, 소프트웨어, 하드웨어 등을 표현하기 위한 모델로서 Super Distributed 오브젝트(SDO)라는 표준을 제정한다. SDO는 객체를 모니터링하고 설정하고 검색하기 위한 인터페이스는 물론 개체간의 주종관계를 포함한 연관 관계 관리를 위한 인터페이스도 제공한다. 마찬가지로 센서 데이터 처리에서의 표현 유연성을 위하여 OGC(Open Geospatical

Consortium) 센서 모델 언어 표준(SensorML)을 정의하여 센서 웹 응용간의 개방형 인터페이스를 제공하고 IEEE 1451과의 호환성 지원, 센서 위치 표현, 공간 정보와 센서 정보의 결합된 표현을 제공한다. 더 나아가서는 사물과 인간간의 의미론적 표현을 통해 원하는 정보를 쉽고 빠르게 검색하며, 공유하고 조합하기 위한 Web 3.0이라고도 불리는 시멘틱 웹(Semantic Web)기술이 있다. 시멘틱 웹에서는 웹 환경에서 기계들이 이해하는 정보를 교환하는 애플리케이션 간에 상호운용성을 제공하기 위해 W3C에서 정의한 웹 메타 데이터의 표준인 Resource Description Framework(RDF)를 사용한다. RDF는 컴퓨팅 기기들 간에 정보를 교환, 검색, 추출하고, 분석하며 자동 처리하는 것을 목적으로, 각종 정보 자원들을 기술하는 메타데이터이다. 이는 특정 자원과 그 속성 값을 묶어서 하나의 단위로 주어, 동사, 목적어로 구성되는 온톨로지에 의해 표현되기 때문에 컴퓨팅 기기들이 가질 수 있는 기본적인 사물 지식의 표현에 적합하다. 웹은 현재 인간이 인터넷을 통해 가장 많이 사용하는 기술로서 접근성이 뛰어나다는 관점에서, ETH zurich에서는 IoT의 사물에 대한 접근성을 높이고자 에서와 같이 모든 사물을 Web으로 통합하는 Web of Things(WoT) 비전을 제시한다. WoT에서는 스마트 게이트웨이를 통하여 센서와 엑츄에이터들을 REST 웹 패턴으로 표현하고 동시에 EPC 네트워크에 등록되어 있는 사물들을 REST기반의 웹으로 연결함으로써 궁극적으로는 물리적 사물들을 웹 인터페이스를 통해 검색하고 제어하고 조합(Composition)하는 PhysicalMashup까지 가능하게 된다. 이는 추후 현재의 인터넷 공간만이 아니라 물리적 세계까지 검색, 추출, 분석을 접근 성이 높은 웹을 통해 가능하게 함으로써 인간이 얻을 수 있는 지식을 넓히는데 기여할 것이다.

III. 개방형 API 기반의 실내 센서웹

1. 센서웹 구조 방안

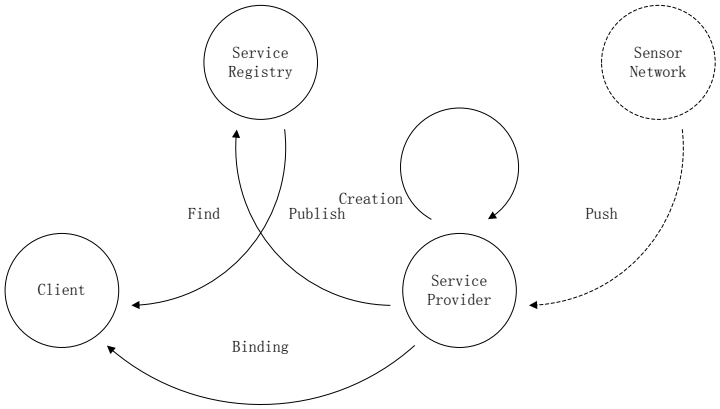

그림 6 기존 센서웹 시스템 모델

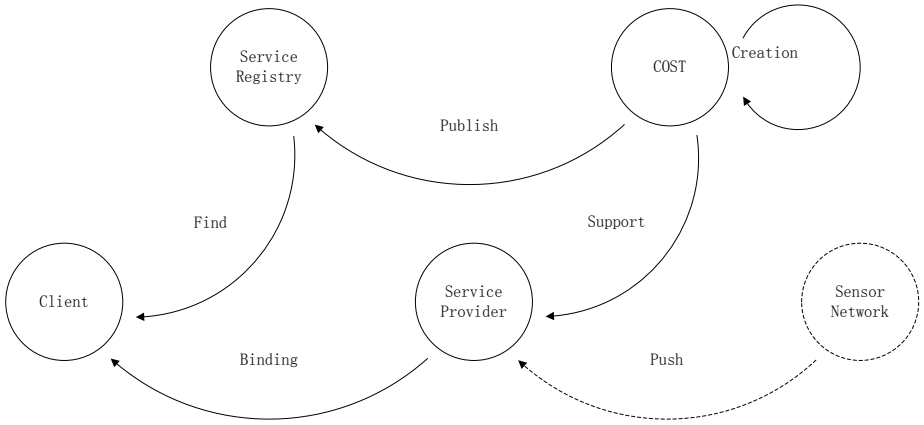

그림 7 COST 기반의 센서웹 시스템 모델[18]

그림 6은 기존 센서웹 시스템 모델 이다. 이는 서비스 등록자, 클라이언트, 서비스 공급자와 센서 네트워크로 구성된다. 센서 네트워크는 외부의 각 센서노드에서 보내는 센싱 데이터를 수집하여 서비스 공급자에게 보낸다. 서비스 공급자가 서비스 콘텐츠를 생성하고 서비스 등록자에게 자기의 서비스 접속 정보를 전송한다. 서비스 등록자가 다양한 센서웹 공급자의 서비스 정보를 저장하고 관

리하여 클라이언트에게 검색 서비스를 제공한다. 클라이언트가 센서웹 공급자에서 지도 데이터 및 센서 노드 정보를 요청하여 가시화하는 역할을 수행한다.

그림 7은 COST(Content Support Tool) 기반의 센서웹 시스템 모델이다. 이는 일반 센서웹 모델과 비교하여 클라이언트, 서비스 등록자 및 센서 네트워크 기능의 변경이 없고 단지 서비스 공급자의 서비스 콘텐츠 생성 및 서비스 정보 등록역할을 COST라는 모듈에 옮긴 것이다. 그래서 COST가 서비스 콘텐츠를 생성하고 서비스 정보를 등록하는 역할을 담당한다. 서비스 공급자는 서비스 콘텐츠의 저장 및 공급 역할만 수행한다. 위의 구조에서 COST는 SOA 기반의 센서웹에서 콘텐츠 생성을 분리하여 서비스 공급자의 일괄 관리가 용이하며, 전문적인 콘텐츠 생성이 가능하다. 서비스 공급자 입장에서는 콘텐츠 생성에 대한 부담이 줄어들어 서비스 제공에 전념할 수 있으므로 보다 안정적으로 서비스 제공을 할 수 있는 환경이 제공된다.

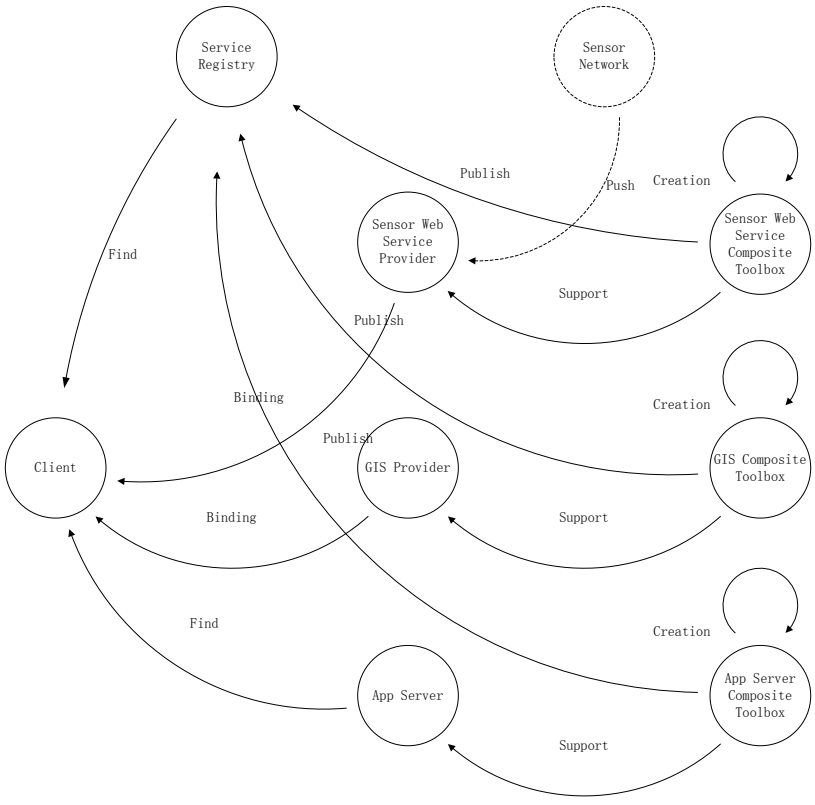

그림 8 서비스 공급자 기반 센서웹 시스템 모델

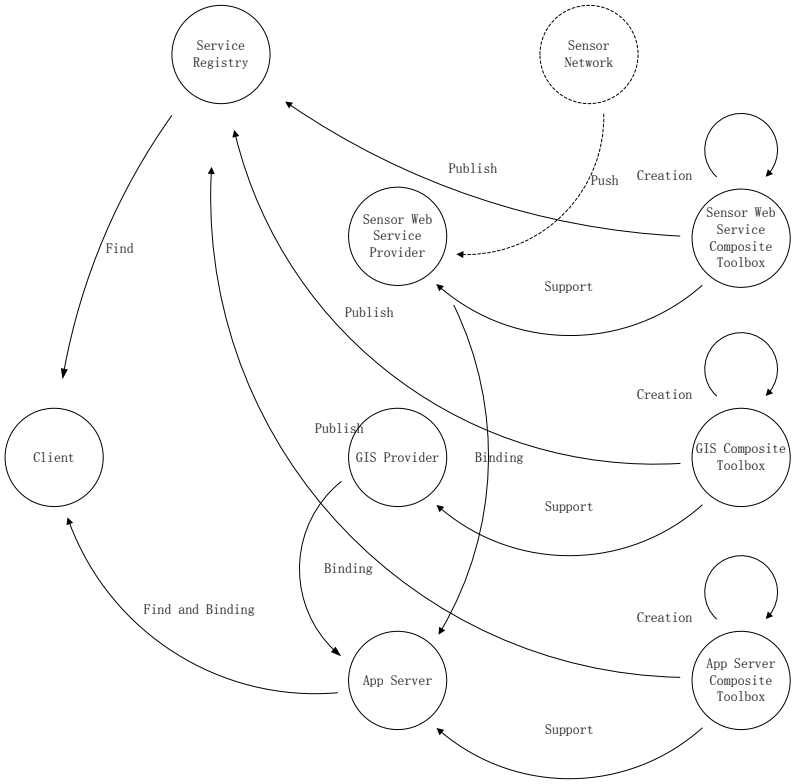

그림 9 응용서버 기반 센서웹 시스템 모델

그림 8은 본 논문이 제안하는 첫 번째 COST기반의 GIS 분리 센서웹 시스템 구조이다. 이 모듈의 구조는 분명히 일반 센서웹 시스템 보다는 더 복잡하지만 핵심 차이는 원래의 서비스 공급자가 제공하는 지도 서비스를 GIS 공급자로 분리되는 것이다. COST도 이에 따라서 GIS 공급자가 서비스 콘텐츠를 관리 및 생성을 위해 GIS 서비스 저작도구로 분리된다. 원래의 구조는 서비스 공급자가 동시에 GIS정보 및 센서 노드를 갖고 있으므로 센서 노드의 위치정보를 서비스 공급자에게 같이 제공하면 시스템은 제대로 운영 가능하지만 GIS 공급자 분리의 경우 위치정보를 서비스 공급자가 제공하는 것이 불가능하므로 응용서버가 필요 하다. 응용서버는 오브젝트의 위치 정보 및 오브젝트 서비스 공급자의 바인딩 주소를 제공하며 응용서버 저작도구는 오브젝트의 위치 매핑 및 GIS 서비스 바인딩 이라는 역할을 담당한다. 실제 시스템 운영 입장에서 센서 제조 회사가 생산하는 센서 노드의 정보를 등록하며 센서의 실제 설치 위치를 몰라도 되고 지도 서비스를

제공하는 회사가 지도 정보만 관리하면 된다. 오브젝트 노드의 위치를 매핑하여 클라이언트가 지도에서 지도의 위치를 정확하게 알 수 있다.

그림 9는 본 논문이 제안하는 두 번째 COST기반의 GIS 분리 센서웹 시스템 구조이다. 그림 9에서 제시하는 시스템 구조와 비슷하지만 응용서버의 역할의 차이를 있다. 그림 8 모델의 응용서버가 오브젝트의 위치 정보 및 오브젝트 서비스 공급자의 바인딩 주소를 제공하며, 클라이언트가 오브젝트의 공급 정보를 요청하고 차체가 공급 서비스를 직접적으로 방문한다. 그림 9 모델의 응용서버가 모두의 공급서비스를 제공하는 API를 통합하여 한 서비스 형태로 클라이언트에게 제공한다. 그래서 모델1번은 서비스 공급자 기반의 센서웹이라고 하며, 모델2번은 응용서버 기반의 센서웹이라고 한다. 서비스 공급자 기반 센서웹의 시스템 구조 공급 서비스 분산 처리하는 구조를 사용하기 때문에 응용서버로 작업해야 하는 일이 줄여서 간편한 모듈을 된다. 응용서버 기반 센서웹은 모두 공급서비스를 자체에서 포장하여 클라이언트에게 제공하기 때문에 응용서버가 서비스를 통합 관리하는 기능이 있다. 본 논문은 이상 제시하는 두 시스템 구조를 사용하여 센서웹 시스템을 설계하여 구현한다. 그리고 센서웹 시스템에서 제시하는 시스템 구조를 참조대상으로 구동체웹 시스템을 설계하여 구현한다. 최종은 센서웹 시스템과 구동체웹 시스템을 통합하여 IOT 시스템을 설계 및 구현하고 성능 평가한다.

2. 실내 센서웹 설계

2.1 Open API 제공 방안

Open API는 누구나 사용할 수 있도록 공개된 API를 말한다. API(응용 Programming Interface)는 응용 프로그래밍 인터페이스이다. Open API는 요즘의 서비스 형 웹 응용이며 웹 회사가 자기의 웹사이트 서비스를 포장하고 외부 제3자 개발자에게 제공한다. 본 논문은 Microsoft사에서 제공하는 .net framework 플랫폼 환경의 WCF기술로 구현한다.

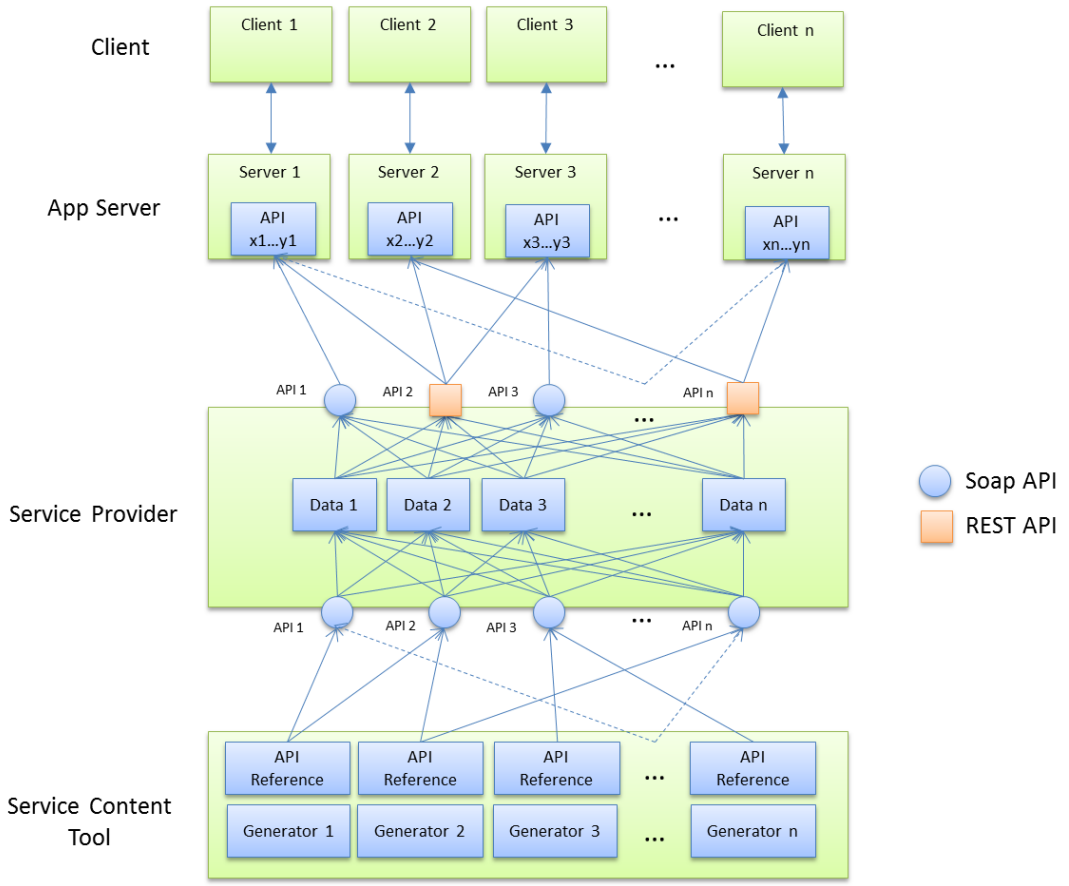

그림 10 Open API을 기반의 분산 처리 시스템 구조

Open API는 본 논문의 시스템 구조 제안이 그림 10과 같은 역할을 수행한다. 서비스 콘텐츠 도구는 서비스 공급자가 제공하는 API를 이용하여 생성되는

서비스 콘텐츠를 저장하고, 응용서버는 서비스 공급자가 제공하는 API로부터 데이터를 받고 클라이언트를 통해 사용자에게 보여준다.

WCF는 윈도우 기반의 서비스를 개발하고 배포하기 위한 기술이다. WCF는 CLR(Common Language Runtime)타입 서비스로, 다른 타입의 서비스를 CLR 타입으로 쉽게 교류할 수 있는 환경을 제공하고 있다. 이론적으로는 WCF 없이도 서비스를 구현할 수 있지만, 실제 현업에서 WCF를 이용하면 훨씬 쉽게 구현할 수 있다. WCF는 서비스 상호 작용, 커뮤니케이션 타입, 마셜링 그리고 여러 가지 프로토콜 관리 등을 정의하는 업계 표준의 마이크로소프트사의 구현이라고 할 수 있다. 따라서 WCF는 여러 서비스들 사이에 상호 운용성을 제공한다. WCF는 많은 애플리케이션에서 요구하는 필수적인 일련의 개발 요소들을 대부분 제공하고 있기 때문에 개발자들의 생산성을 높일 수 있다. 또한 호스팅, 서비스 인스턴스 관리, 비동기 호출, 신뢰도, 트랜잭션 관리, 비 연결 큐 호출 및 보안 등과 같은 여러 가지 유용한 기능들을 제공함과 더불어 뛰어나고 우아한 확장 모델을 가지고 있기 때문에 기본적으로 제공되는 기능만 가지고도 쉽게 확장하여 사용할 수 있다. 사실상 WCF 자체가 이 모델을 기반으로 만들어진 것이라고 보면 된다.

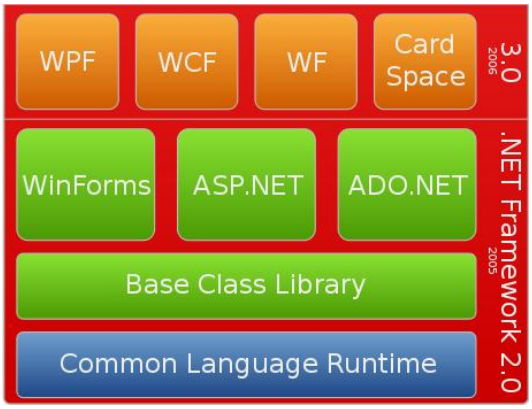

그림 11 .NET Framework구조

그림 11은 .NET Framework의 버전 구조도 이다. 닷넷 프레임워크 3.0의 구성 요소인 WCF는 닷넷 프레임워크 2.0을 필요로 하기 때문에 닷넷 프레임워크 2.0을 지원하는 OS에서만 동작한다. 즉, 윈도우 비스타, 윈도우XP 서비스 팩2 그

리고 윈도우 서버2003서비스 팩1이나 그 이상 버전의 OS들이 WCF를 지원하고 있다.

서비스는 외부에 노출되는 기능의 한 단위이다. 이 점에서 서비스는 함수에서 개체로, 개체에서 컴포넌트로, 컴포넌트에서 서비스라는 단계로 진화하는 과정의 다음 진보라고 할 수 있다. SO (Service-Orientation)는 SO 애플리케이션을 만들기 위한 추상적인 원칙과 최상 방법론의 집합이다. SOA 즉, 서비스를 지향하는 애플리케이션은 컴포넌트들로 이루어진 컴포넌트 기반의 애플리케이션이나 객체들로 이루어진 상속 기반의 애플리케이션과 유사한 방식처럼 서비스들을 단일 논리적인 애플리케이션으로 이루어지게 한다. 그림 12는 서비스 지향 애플리케이션을 표시한다.

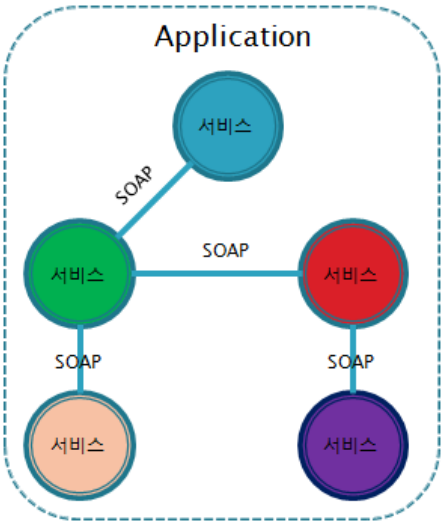

그림 12 서비스 지향 애플리케이션

학교 캠퍼스 빌딩정보 통합관리는 SOA 분산 시스템 센서에서 수집되는 센싱 데이터와 사용자에게 편의성을 제공하기 위한 지리 정보를 같이 출력하는 기능이 있어야 한다. 그런데 단순한 HTML을 이용해서는 가시화 구현이 어렵다. Silverlight는 풍부한 가시화 기술과 기능들을 제공하기 때문에 센서 웹 시스템의 요구를 만족하여 Rich 클라이언트를 쉽게 구현 할 수 있다.

마이크로소프트 실버라이트는 애니메이션, 벡터 그래픽스, 오디오-비디오 재

생을 비롯한 리치 인터넷 애플리케이션에 대한 지원을 제공하는 웹 브라우저 플러그인이다.

실버라이트는 WPF와 비슷한 retained 모드의 그래픽 시스템을 제공하며 멀티미디어, 그래픽스, 애니메이션, 상호 작용을 하나의 런타임으로 통합한다. XAML과 동작하도록 설계되어 있으며 자바스크립트로 작성할 수 있다. XAML은 벡터 그래픽스, 애니메이션을 짜기 위한 마크업 언어로 사용할 수 있다. 실버라이트로 만든 텍스트 콘텐츠는 더 검색이 쉽고 컴파일이 되지 않은 플래시 쪽보다 색인을 만들기 쉽다. 또, 이러한 과정은 문자열(XAML)로 표현한다. 실버라이트는 윈도 비스타용 윈도 사이드바 가젯을 만드는 데 사용할 수 있다.

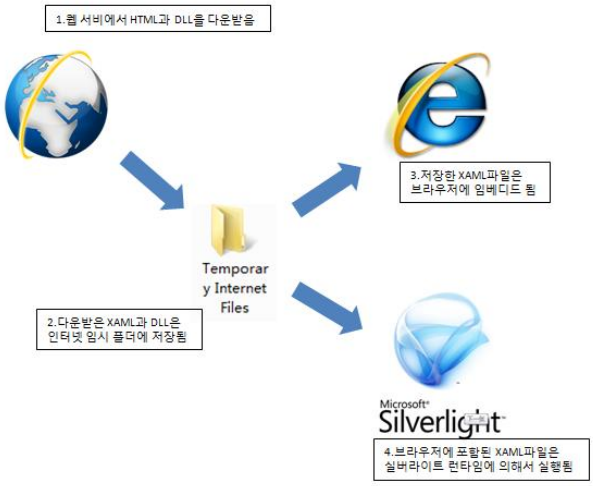

그림 13 실버라이트의 실행구조

2.2 서비스 공급자 기반 실내 센서웹

2.2.1 서비스 공급자 기반 실내 센서웹 시스템의 인터페이스 프로토콜

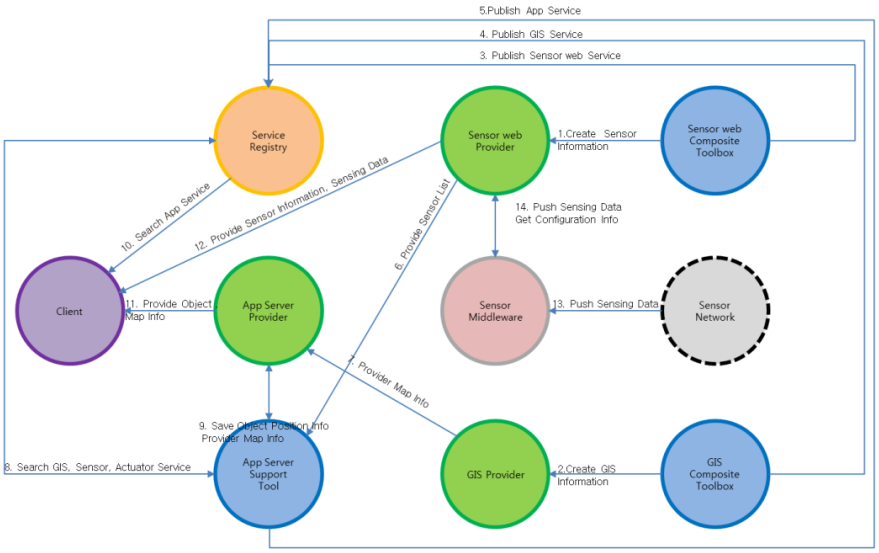

그림 14 서비스 공급자 기반의 센서웹 구조

그림 14는 서비스 공급자 기반의 센서웹 시스템 내부 모듈간의 API 구성도이다. 센서 웹 시스템은 주로 센서 노드 정보관리, 공급 그리고 다양한 센싱 데이터를 제공하는 역할을 담당한다.

표 6 서비스 공급자 기반 센서웹 시스템의 API

Module Name	API Name	Description		Applied At
		Type	Operation	
Service Registry	Service Publish Interface Protocol	Service Information	Write	3, 4, 5
	Service Search Interface Protocol	Service Information	Search, Read	8, 10
GIS Service Provider	GIS Provider Interface Protocol	Map Information	Read	7
	GIS Content Interface Protocol	Map Information	Read, Write	2
Sensor web Service Provider	Sensor web Service Provider Interface Protocol	Sensor List	Read, Search	6
		Sensor Information	Read	6, 12
		Sensing Data	Read	12
	Sensor web Service Content Interface Protocol	Sensor List	Read, Search	1
		Sensor Information	Write	1
App Server	App Service Provider Interface Protocol	Map Information	Read	11
		Object Information	Read	11
	App Service Content Interface Protocol	Map Information	Read	9
		Object Information	Read, Write	9

표 6은 서비스 공급자 기반의 센서웹 시스템 내부에 존재하는 API이다. 그 표와 그림 14를 같이 참조 하면 각 모듈간 어떤 API를 통해 연동하는 것을 쉽게 이해할 수 이다. Service Publish API는 서비스 등록역할을 수행하고, Service Search API는 서비스 정보 검색역할을 수행하고, GIS Service Provider API는 지도정보 공급역할을 수행하고, GIS Content API는 지도정보 관리역할을 수행, 센서웹 Service Provider API는 센서 정보 및 센싱 정보 공급역할을 수행하고, 센서웹 Content API는 센서 정보 관리역할을 수행하고, App Service Provider API는 지도정보 및 오브젝트 정보 공급역할을 수행하고, App Service Content API는 지도 서비스 공급, 오브젝트 정보 관리역할을 수행한다.

2.2.2 환경 설정 단계

센서 웹 서비스의 배치단계는 서비스의 필수 콘텐츠를 제대로 클라이언트에게 공급하고 검색서비스를 제공하기 위해 생성한다.

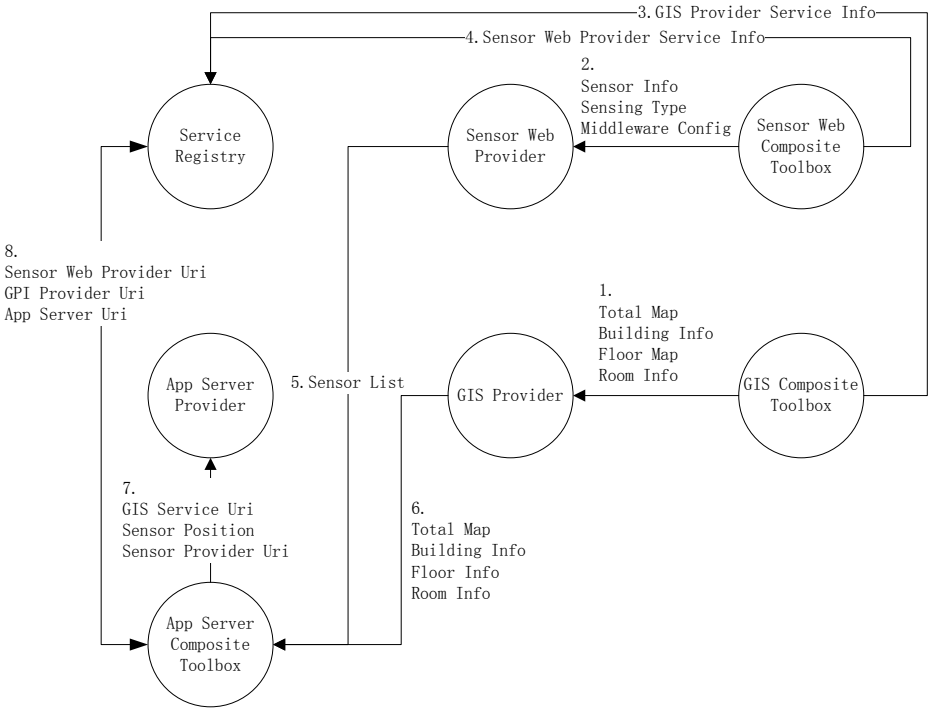

그림 15 서비스 공급자 기반 실내 센서웹의 환경 설정 단계

그림 15은 서비스 공급자 기반 센서웹의 환경 설정 단계이다. Total Map은 실외 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Building Info는 빌딩의 이름, 위치 정보, 층 이름을 포함한다. Floor Map은 실내 지도의 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Room Info는 방의 이름, 방 위치를 포함한다. Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. 센서 Type는 센서의 센싱 타입 정보를 의미한다. Middleware Config는 센서 미들웨어의 ID, IP주소와 접속권한 정보를 포함한다. Service Info는 각각 서비스 공급자의 서비스 이름, 검색 키워드, 접속 주소, 서비스 유형을 포함한다. 센서 List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 센서 ID들을 의미한다. GIS Service Uri는 GIS 공급자의 접속주소를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. 센서웹 공급자 Uri는 센서웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

GIS 서비스 저작도구는 지도 관련 데이터를 외부에서 GIS Content API를 통해 GIS 공급자의 DB에 저장한다. 센서웹 저작도구는 각 센서노드의 정보를 센서웹 Content API를 통해 생성하여 DB에 저장하여 미들웨어의 배치 정보를 생성한다. GIS 서비스 저작도구가 GIS 공급 서비스의 정보를 서비스 등록자에게 등록한다. 센서웹 저작도구가 Sensor Web Provider의 정보를 서비스 등록자에게 등록한다. 센서웹 공급자가 제공하는 센서 리스트 정보와 GIS 공급자가 제공하는 지도 데이터를 응용서버 저작도구에 제공한다. 그리고 응용서버 저작도구가 각 센서 노드와 지도의 위치정보를 생성하여 저장하고 응용서버의 서비스 정보를 등록한다.

그림 16은 서비스 공급자를 기반 센서웹의 환경 설정 단계 시퀀스 다이어그램이다. 위의 과정은 센서웹 저작도구, GIS 서비스 저작도구, 응용서버 저작도구, 센서웹 공급자, GIS 공급자, 응용서버 서비스 공급자와 서비스 등록자를 구성한다.

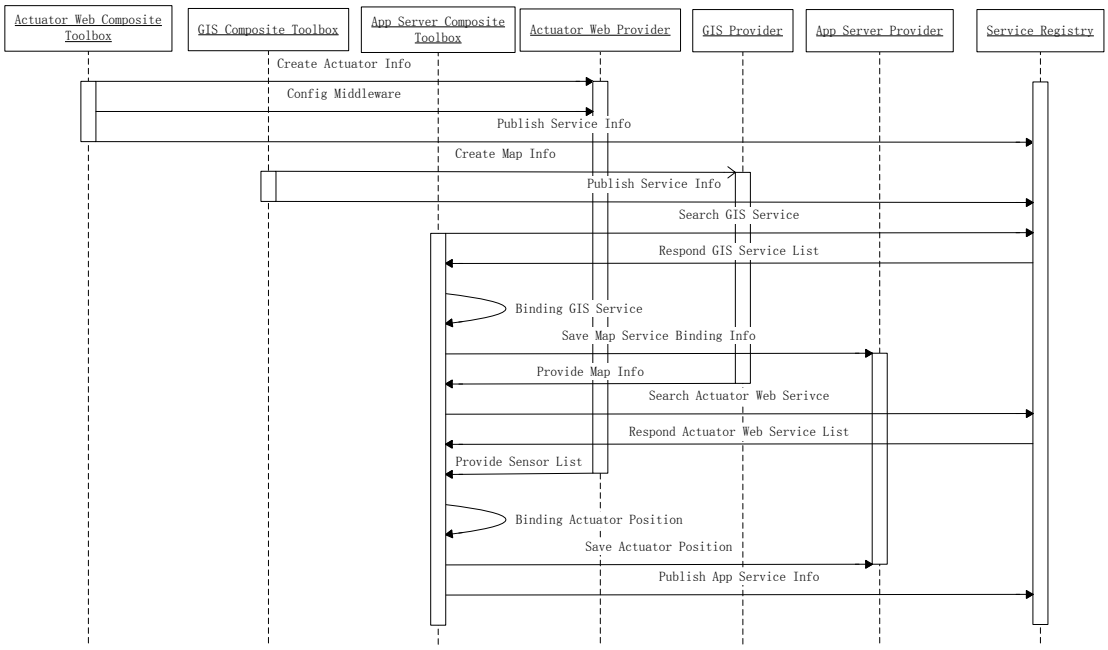

그림 16 서비스 공급자 기반 실내 센서웹의 환경 설정 단계 시퀀스

이 단계가 실행될 때 센서웹 저작도구는 센서 오브젝트의 정보를 생성하여 센서웹 공급자에 저장하며 센서 미들웨어의 배치 정보를 생성하여 센서웹 공급자에게 등록한다. 다음에 Sensor Web Provider 정보를 서비스 등록자에게 등록한다. 그리고 GIS 서비스 저작도구가 지도정보를 생성하여 GIS 공급자에 전송하고 GIS 공급 서비스 정보를 서비스 등록자에게 등록한다. 응용서버 저작도구는 서비스 등록자에서 GIS 서비스 정보를 검색 요청하여 관리자가 원하는 지도 엔진 서비스를 골라서 바인딩 한다. 지도 서비스를 바인딩 하여 서비스 등록자가 제공하는 센서웹 서비스 공급 서비스 정보를 뽑아서 이 센서웹 공급자가 제공하는 센서노드 정보를 보여준다. 관리자가 선택된 센서노드를 지도에서 하나씩 위치를 바인딩하고 응용서버 서비스 정보를 서비스 등록자에 저장한다.

2.2.3 검색 단계

서비스 검색 단계는 클라이언트가 서비스 등록자에서 쉽게 응용서버 및 센서노드를 찾을 수 있도록 한다.

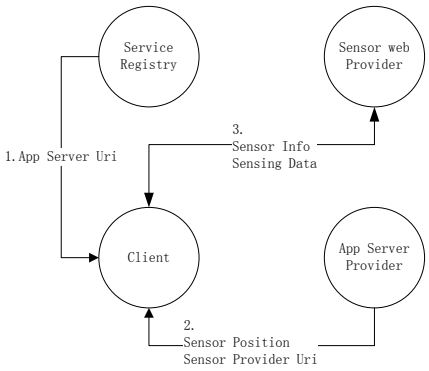

그림 17 서비스 공급자 기반 실내 센서웹의 검색 단계

그림 17은 서비스 공급자 기반 실내 센서웹에 대한 검색 단계이다. Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. Sensing Data는 센서의 실시간 센싱 데이터를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. 센서웹 공급자 Uri는 센서웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다. 클라이언트는 사용자가 입력하는 서비스검색 키워드를 통해 서비스 등록자에게 서비스 리스트를 요청한다. 클라이언트가 응용서버에 접속하여 센서의 위치 및 공급서비스 정보를 요청한다. 클라이언트가 센서 ID하고 공급서비스 주소를 통해 직접 센서웹 공급자로 연결하여 센서 정보와 센싱 데이터를 요청한다.

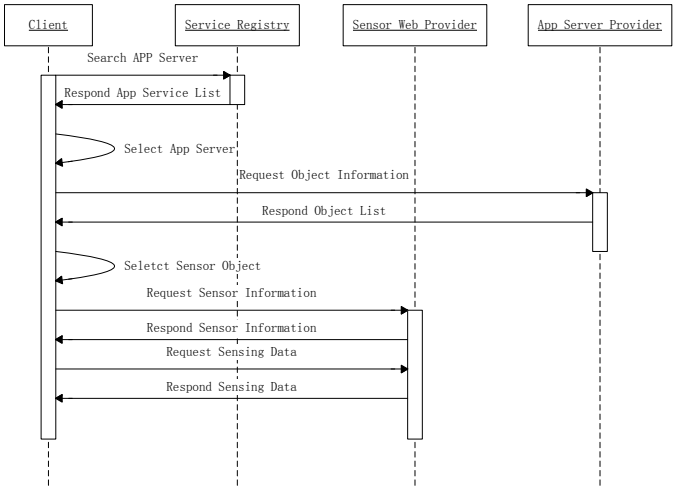

그림 18 서비스 공급자 기반 실내 센서웹의 검색 단계 시퀀스

그림 18는 서비스 공급자 기반 센서웹에 대한 검색 단계 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 서비스 등록자, 센서웹 공급자, 응용서버를 구성한다.

처음에 클라이언트는 서비스 등록자에서 응용서버의 서비스 정보를 검색하고 사용자가 원하는 응용서버에 접속하며 응용서버에서 센서 오브젝트 정보를 지도상에 출력한다. 사용자가 지정하는 센서 오브젝트를 센서 정보가 속하는 센서웹 공급자에서 세부 정보와 센싱 데이터를 요청한다.

2.2.4 센싱 데이터 수집 단계

센싱 데이터 수집 단계는 센서 네트워크 미들웨어가 센서 네트워크에서 수신되는 센싱 데이터를 센서웹 공급자에게 전달하여 저장한다. 클라이언트가 센서웹 공급자에게 센싱 데이터를 요청한다.

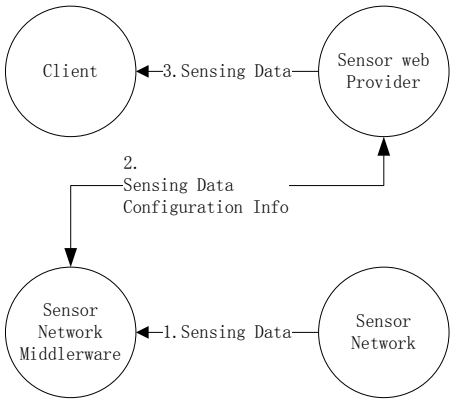

그림 19 서비스 공급자 기반 실내 센서웹의 센싱 데이터 수집 단계

그림 19은 서비스 공급자 기반 실내 센서웹의 센싱 데이터 수집 단계 이다. Sensing Data는 센서의 실시간 센싱 데이터를 의미한다. Configuration Info는 센서 미들웨어의 IP 주소 및 서비스 접속 권한을 의미한다.

센서 네트워크에서 동작하는 센서 노드들에서 센싱 데이터를 수신한다. 센서

네트워크 미들웨어는 센서 네트워크에서 수집되는 센싱 데이터를 센서웹 공급자에 전송한다. 클라이언트가 센서웹 공급자에게 센싱 데이터를 요청한다.

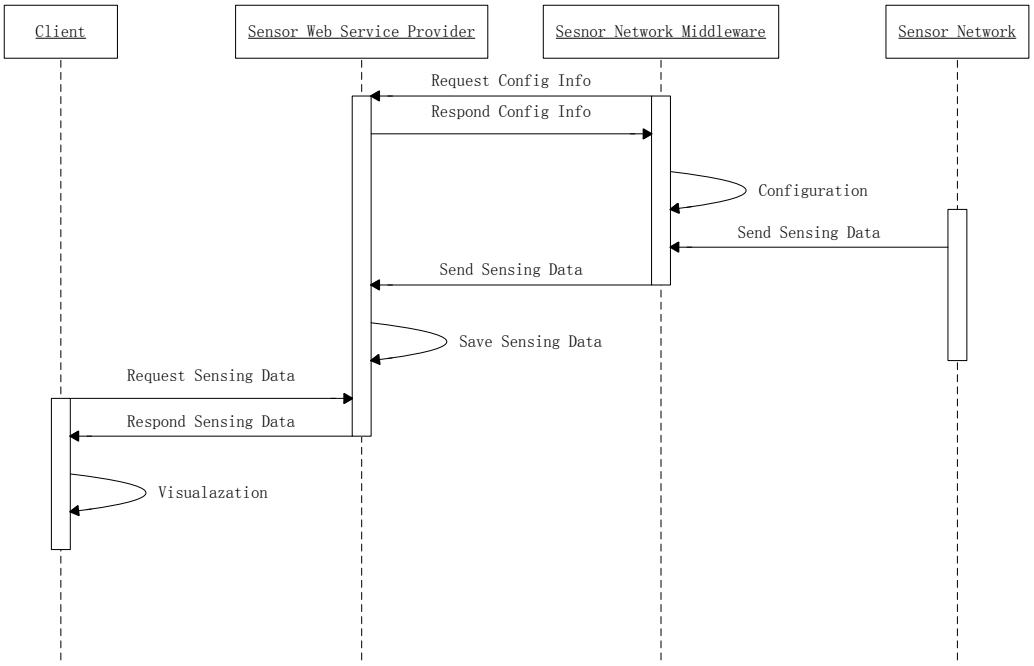

그림 20 서비스 공급자 기반 실내 센서웹의 센싱 데이터 수집 단계 시퀀스

그림 20은 서비스 공급자를 기반 센서웹의 센싱 데이터 수집 단계의 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 센서웹 공급자, 센서 네트워크 미들웨어와 센서 네트워크로 구성된다.

위의 과정은 시작할 때 센서 네트워크 미들웨어가 센서웹 공급자에게 자기 배치 관련정보를 요청 받아서 배치한다. 다음에 미들웨어가 계속 센서 네트워크에 있는 센서의 센싱 데이터를 수집하여 센서웹 공급자에 저장한다. 클라이언트가 지정하는 센서의 센싱 데이터를 센서웹 공급자에서 요청 받아서 뷰어를 통해 볼 수 있다.

2.3 응용서버 기반 실내 센서웹

2.3.1 응용서버 기반 실내 센서웹 시스템의 인터페이스 프로토콜

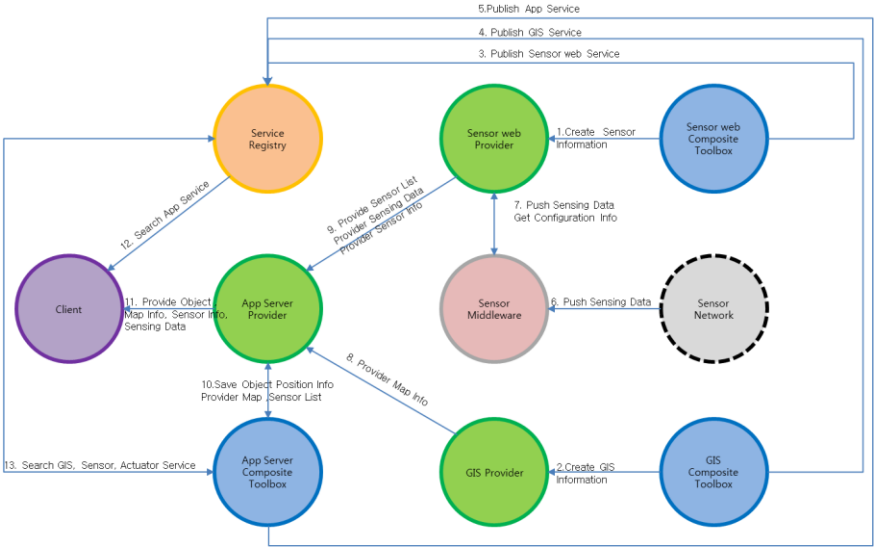

그림 21 응용서버 기반의 센서 웹 구조

그림 21은 응용서버 기반의 센서웹 시스템 내부 모듈간의 API 구성도이다. 센서 웹 시스템은 주로 센서 노드 정보관리, 공급 그리고 다양한 센싱 데이터를 제공하는 역할을 담당한다.

표 7 응용서버 기반 센서웹 시스템의 API

Module Name	API Name	Description		Applied At
		Type	Operation	
Service Registry	Service Publish Interface Protocol	Service Information	Write	3, 4, 5
	Service Search Interface Protocol	Service Information	Search, Read	12, 13
GIS Service Provider	GIS Provider Interface Protocol	Map Information	Read	8
	GIS Content Interface Protocol	Map Information	Read, Write	2
Sensor web Service Provider	Sensor web Service Provider Interface Protocol	Sensor List	Read, Search	9
		Sensor Information	Read	9
		Sensing Data	Read	9
	Sensor web Service Content Interface Protocol	Sensor List	Read, Search	1
		Sensor Information	Write	1
App Server	App Service Provider Interface Protocol	Map Information	Read	11
		Sensor Information	Read	11
		Sensing Data	Read	11
		Object Information	Read	11
	App Service Content Interface Protocol	Map Information	Read	10
		Sensor Information	Read	10
		Sensor List	Read, Search	10
		Object Information	Read, Write	10

표 7은 응용서버 기반의 센서웹 시스템 내부 존재하는 API이다. 그 표와 그림 21를 같이 참조 하면 각 모듈간 어떤 API를 통해 연동하는 것을 쉽게 이해할 수 이다. Service Publish API는 서비스 등록역할을 수행하고, Service Search API는 서비스 정보 검색역할을 수행하고, GIS Service Provider API는 지도정보 공급하는 역할을 수행하고, GIS Content API는 지도정보를 관리하는 역할을 수행, Sensor Web Provider API는 센서 정보 및 센싱 정보를 공급하는 역할을 수행하고, Sensor Web Content API는 센서 정보 관리하는 역할을 수행하고, App Service Provider API는 지도정보, 센서 정보, 센싱 데이터 및 센서 오브젝트 정보를 공급하는 역할을 수행하고, App Service Content API는 지도 서비스 및 센서 정보를 공급하며, 센서 오브젝트 정보를 관리하는 역할을 수행한다.

2.3.2 환경 설정 단계

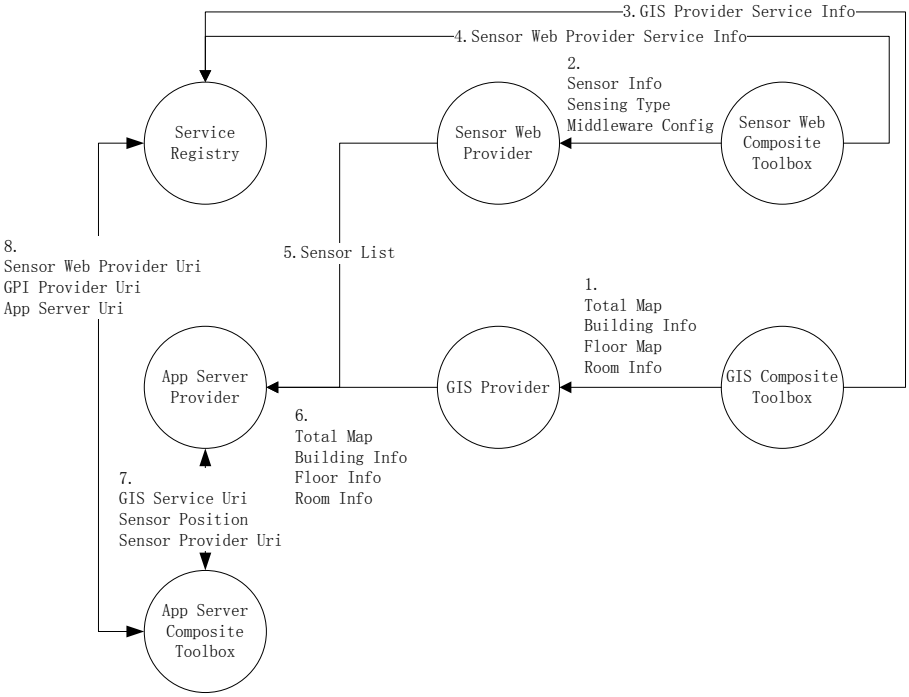

그림 22 응용서버 기반 실내 센서웹 시스템의 환경 설정 단계

그림 22는 응용서버 기반 실내 센서웹의 환경 설정 단계이다. 센서 웹 서비

스의 배치단계는 서비스의 필수 콘텐츠를 제대로 클라이언트에게 공급하고, 검색 서비스를 제공하기 위해 생성한다.

Total Map은 실외 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Building Info는 빌딩의 이름, 위치 정보, 층 이름을 포함한다. Floor Map은 실내 지도의 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Room Info는 방의 이름, 방 위치를 포함한다. Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. 센서 Type는 센서의 센싱 타입 정보를 의미한다. Middleware Config는 센서 미들웨어의 ID, IP주소와 접속권한 정보를 포함한다. Service Info는 각각 서비스 공급자의 서비스 이름, 검색 키워드, 접속 주소, 서비스 유형을 포함한다. 센서 List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 센서 ID들을 의미한다. GIS Service Uri는 GIS 공급자의 접속주소를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. 센서웹 공급자 Uri는 센서웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

GIS 서비스 저작도구는 지도 관련 데이터를 외부에서 GIS Content API를 통해 GIS 공급자의 DB에 저장한다. 센서웹 저작도구는 각 센서노드의 정보를 센서웹 Content API를 통해 생성하여 DB에 저장하여 미들웨어의 배치 정보를 생성한다. GIS 서비스 저작도구가 GIS 공급 서비스의 정보를 서비스 등록자에게 등록한다. 센서웹 저작도구가 센서웹 공급자 서비스의 정보를 서비스 등록자에게 등록한다. 센서웹 공급자가 제공하는 센서 리스트 정보와 GIS 공급자가 제공하는 지도 데이터를 응용서버를 통하여 응용서버 저작도구에게 제공한다. 그리고 응용서버 저작도구가 각 센서 노드와 지도의 위치정보를 생성하여 저장하고 응용서버의 서비스 정보를 등록한다.

그림 23은 응용서버 기반 실내 센서웹의 환경 설정 단계 시퀀스 다이어그램이다. 위의 과정은 센서웹 저작도구, GIS 서비스 저작도구, 응용서버 저작도구, 센서웹 공급자, GIS 공급자, 응용서버 서비스 공급자와 서비스 등록자로 구성한다.

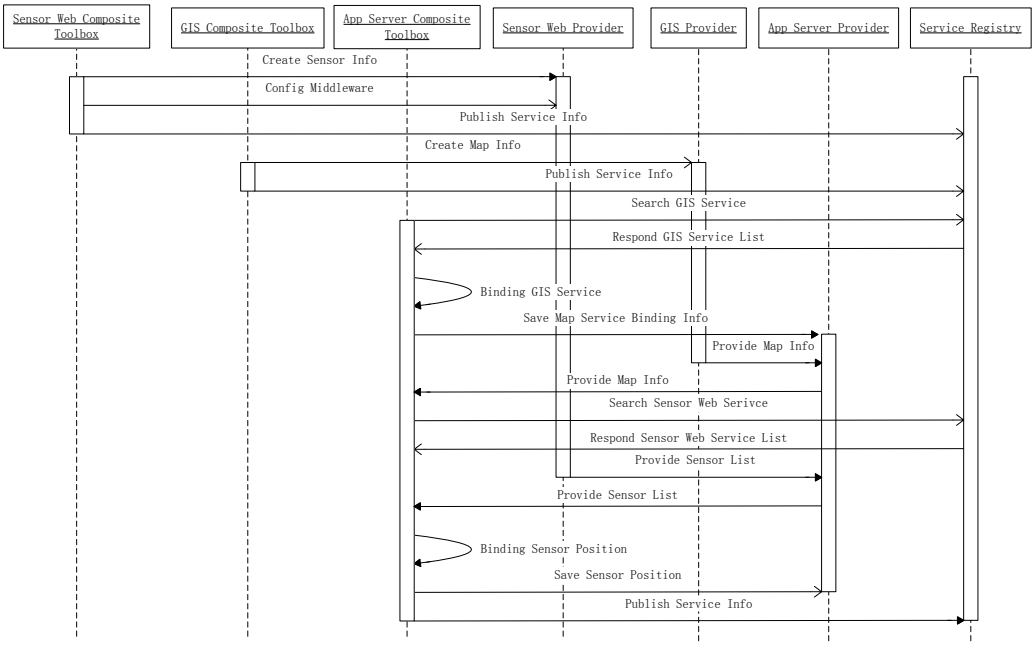

그림 23 응용서버 기반 실내 센서웹 시스템의 환경 설정 단계 시퀀스

이 단계가 실행될 때 센서웹 저작도구는 센서 오브젝트의 정보를 생성하여 센서웹 공급자에 저장하며 센서 미들웨어의 배치 정보를 생성하여 센서웹 공급자에게 등록한다. 다음에 Sensor Web Provider 정보를 서비스 등록자에게 등록한다. 그리고 GIS 서비스 저작도구가 지도정보를 생성하여 GIS 공급자에게 전송하고 GIS 공급 서비스 정보를 서비스 등록자에게 등록한다. 응용서버 저작도구는 서비스 등록자에서 GIS 서비스 정보를 검색하여 관리자가 원하는 지도 엔진 서비스를 골라서 바인딩 한다. 지도 서비스를 바인딩 하여 서비스 등록자가 제공하는 센서웹 서비스 공급 서비스 정보를 뽑아서 응용서버를 통해서 이 센서웹 공급자가 제공하는 센서 노드 리스트 정보를 보여준다. 관리자가 선택된 센서노드를 지도에서 하나씩 위치를 바인딩하고 응용서버 서비스 정보를 서비스 등록자에 저장한다.

2.3.3 검색 단계

서비스 검색 단계는 클라이언트가 서비스 등록자에서 쉽게 응용서버 및 센서

노드를 찾을 수 있도록 한다.

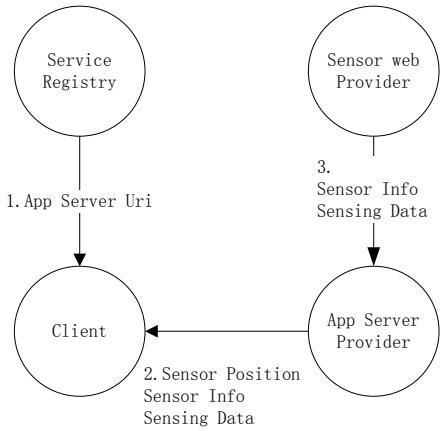

그림 24 응용서버 기반 실내 센서웹 시스템의 검색 단계

그림 24는 응용서버 기반 실내 센서웹의 검색 단계이다. Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. Sensing Data는 센서의 실시간 센싱 데이터를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

클라이언트는 사용자가 입력하는 서비스검색 키워드를 통해 서비스 등록자에게 서비스 리스트를 요청한다. 클라이언트가 응용서버를 접속하여 센서의 위치 및 공급서비스 정보를 요청한다. 클라이언트가 응용서버를 통하여 간접 연결하는 센서웹 공급자에서 센서 정보와 센싱 데이터를 요청한다.

그림 25는 응용서버 기반 실내 센서웹의 검색 단계 시퀀스 다이어그램이다. 이 과정은 클라이언트, 서비스 등록자, 센서웹 공급자, 응용서버로 구성한다. 처음에 클라이언트는 서비스 등록자에서 응용서버의 서비스 정보를 검색하고 사용자가 원하는 응용서버를 접속하며 지도에 따라서 응용서버에서 센서 오브젝트 정보를 지도상에 출력한다. 사용자가 지정하는 센서 오브젝트의 정보를 요청하려면 응용서버를 통해 그 센서 오브젝트가 속한 센서웹 공급자에게 요청한다. 센서웹 공급자가 센서 정보를 응답하여 다시 응용서버를 통해 클라이언트에게 전달한다.

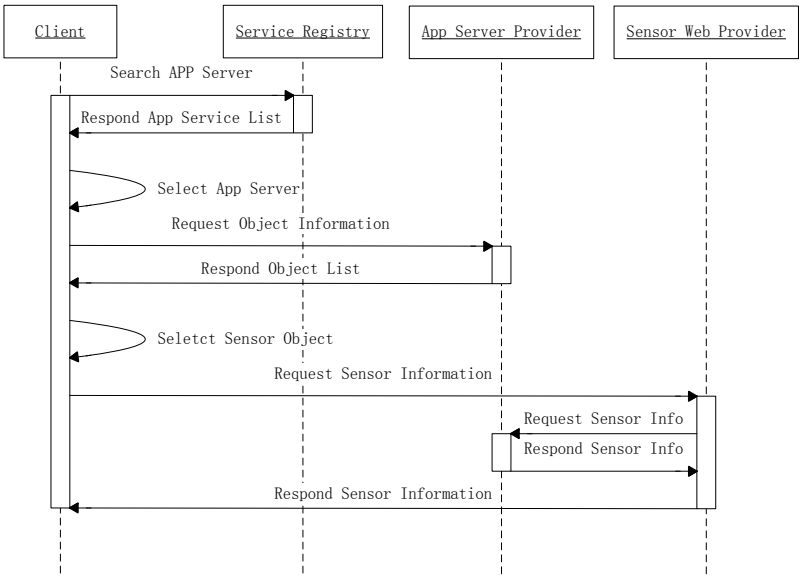

그림 25 응용서버 기반 실내 센서웹 시스템의 검색 단계 시퀀스

2.3.4 센싱 데이터 수집 단계

센싱 데이터 수집 단계는 센서 네트워크 미들웨어가 센서 네트워크에서 수신되는 센싱 데이터를 센서웹 공급자에게 전달하여 저장한다. 클라이언트가 응용서버를 통해 센서웹 공급자에게 센싱 데이터를 요청한다.

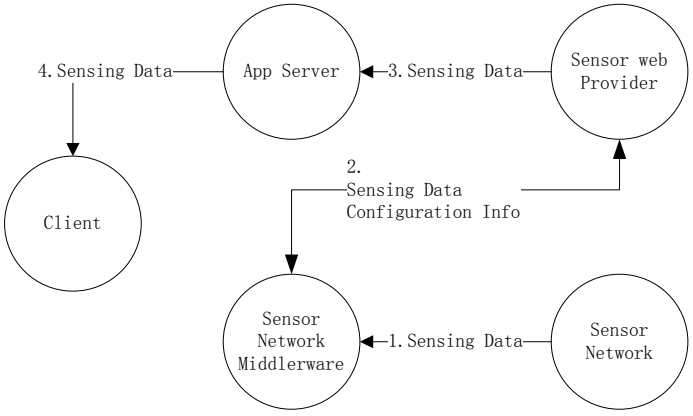

그림 26 응용서버 기반 실내 센서웹 시스템의 센싱 데이터 수집 단계

그림 26은 응용서버 기반 센서웹의 센싱 데이터 수집 단계이다. Sensing Data는 센서의 실시간 센싱 데이터를 의미한다. Configuration Info는 센서 미들웨어의 IP 주소 및 서비스 접속 권한을 의미한다.

센서 네트워크에서 동작하는 센서 노드들에서 센싱 데이터를 수신한다. 센서 네트워크 미들웨어는 센서 네트워크에서 수집되는 센싱 데이터를 센서웹 공급자에게 전송한다. 클라이언트가 센서웹 공급자에서 센싱 데이터를 요청한다.

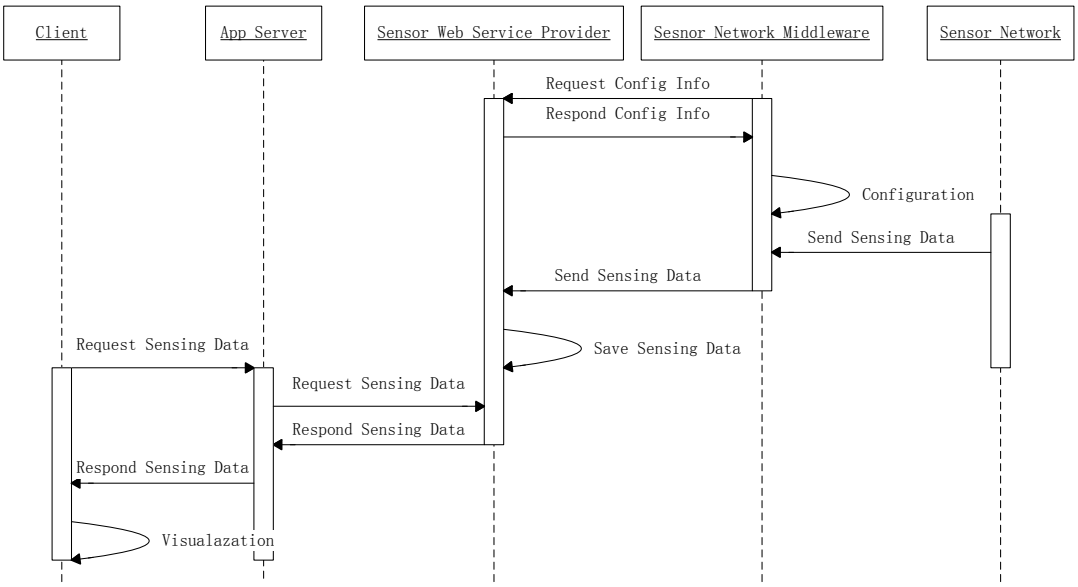

그림 27 응용서버 기반 실내 센서웹 시스템의 센싱 데이터 수집 단계 시퀀스

그림 27은 응용서버 기반 실내 센서웹의 센싱 데이터 수집 단계의 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 응용서버, 센서웹 공급자, 센서 네트워크 미들웨어와 센서 네트워크로 구성된다.

위의 과정은 시작할 때 센서 네트워크 미들웨어가 센서웹 공급자에게 자기 Configuration 관련정보를 요청 받아서 배치한다. 다음에 미들웨어가 계속 센서 네트워크에 있는 센서의 센싱 데이터를 수집하여 센서웹 공급자에 저장한다. 클라이언트가 지정하는 센서의 센싱 데이터를 응용서버를 통해 센서웹 공급자에게

요청한다.

2.4 실내 센서웹 시스템 세부설계

실내 센서웹 시스템중의 센서웹 공급자, 센서웹 저작도구, 센서 미들웨어, GIS 공급자, GIS 저작도구, 응용서버, 응용서버 저작도구, 서비스 등록자 및 클라이언트에 대해 내부 모듈을 설계한다. 본 논문의 III장 1절에서 제시하는 두 방안은 응용서버의 구조만 차이가 있지만 다른 모듈의 구조 같다. GIS 공급자와 GIS 저작도구의 세부구조가 VI장의 1.1절에서 기술하여 있다. 서비스 등록자와 응용서버 저작도구의 세부구조가 V장의 2.3절에서 기술하여 있다.

2.4.1 센서웹 공급자

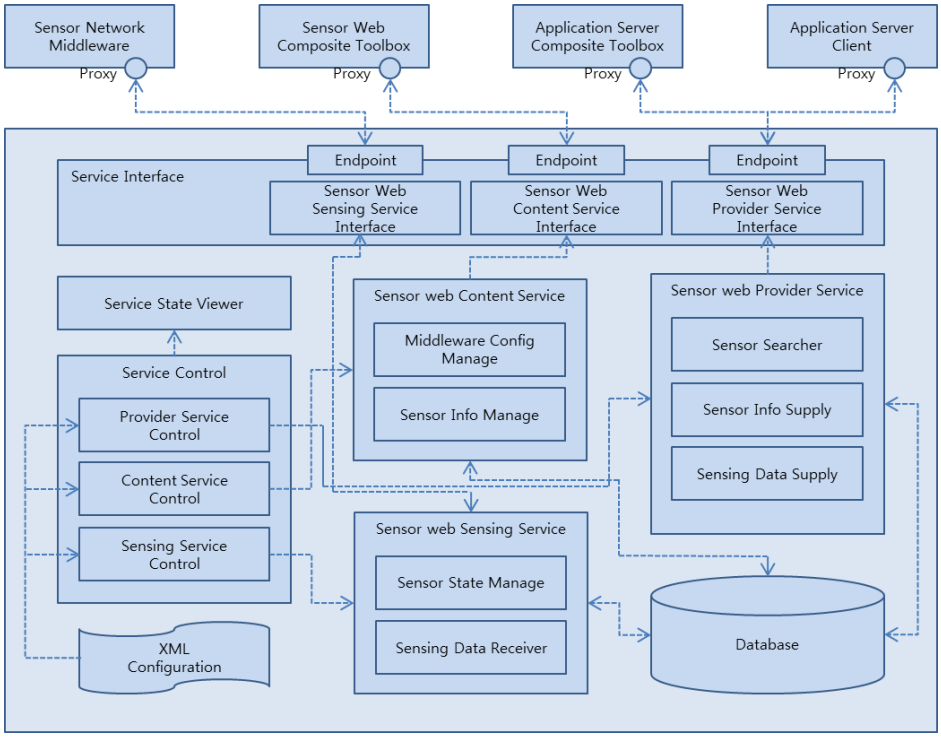

그림 28 센서웹 공급자 세부구조

그림 28은 센서웹 공급자의 세부구조이다. Service Interface는 외부 서비스

에 접속 인터페이스를 제공한다. 센서웹 시스템이 주로 Sensor Web Content Service, Sensor Web Provider Service 및 Sensor Web Sensing Service를 제공한다.

Sensor Web Content Service는 Middleware Config Manage와 Sensor Info Manage로 구성된다. Middleware Config Manage는 미들웨어의 ID, IP 주소와 서비스 접속 권한정보를 생성 및 관리하는 역할을 담당한다. Sensor Info Manage는 센서노드의 이름, ID, 센싱 타입 정보를 생성 및 관리하는 역할을 수행한다. Sensor Web Provider는 Sensor Searcher, Sensor Info Supply와 Sensing Data Supply로 구성된다. Sensor Searcher는 검색 키워드에 의해 센서 ID 리스트를 제공한다. Sensor Info Supply는 센서 ID에 의해 해당 센서노드의 이름, 센싱 타입 정보를 제공한다. Sensing Data Supply는 센서 ID에 의해 해당 센서의 실시간 센싱 데이터를 제공한다(주의: 여기서 나타난 클라이언트가 공급서비스를 사용하는 모두 대상을 의미한다). Sensor Web Sensing Service는 Sensor State Manage 및 Sensing Data Receiver로 구성된다. Sensor State Manage는 센서의 센싱 동작상태를 관리하는 역할을 담당하며, Sensing Data Receiver는 센서 미들웨어에서 수집되는 센싱 데이터를 수신하여 저장하는 역할을 담당한다. Service Control은 Provider Service Control, Sensing Service Control과 Content Service Control로 구성되고 Provider Service, Sensing Service 및 Content Service의 열림, 닫음을 제어한다. Service State Viewer는 서비스의 동작 상태를 도시하는 역할을 수행한다. XML Configuration은 WCF Service의 동작 환경을 배치한다. Database는 센서와 관련 정보(이름, ID, 센싱 데이터 등)를 저장한다.

2.4.2 센서웹 저작도구

그림 29는 센서웹 저작도구의 세부구조이다. 센서웹 저작도구는 센서 정보를 생성, 미들웨어 접속관리 및 서비스 정보등록 역할을 담당한다. Sensor Info Generator는 센서 정보(이름, ID, 센싱 타입 등)를 생성하여 서비스 공급자에 저장한다. Middleware Info Generator는 Middleware 정보(ID, IP주소)를 등록하며, Middleware의 서비스 공급자 접속 권한을 관리해 준다. Sensor Web Service Info Publish는 Service Info Generator를 통해 서비스 정보(검색 키워드, 이름,

주소 등)를 생성하여 XML임시 파일에 저장하여 서비스 등록자에게 등록한다.

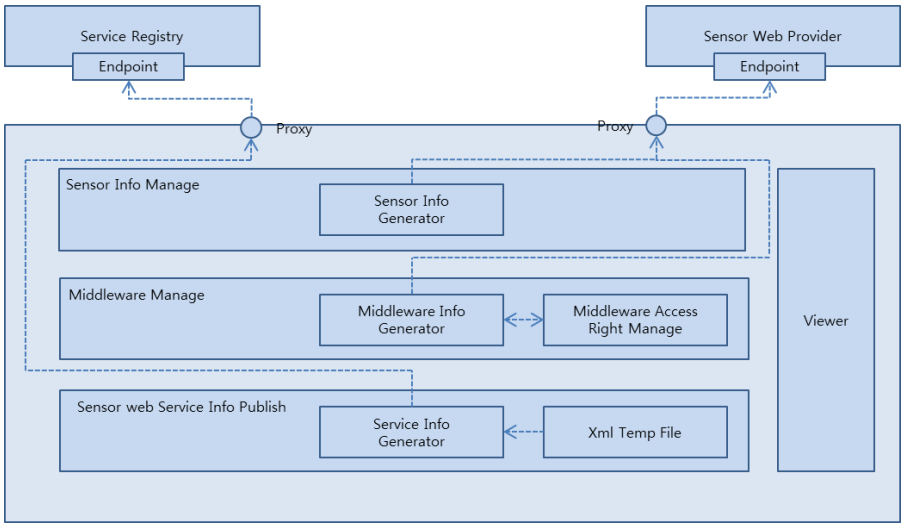

그림 29 센서웹 저작도구 세부구조

2.4.3 센서 미들웨어

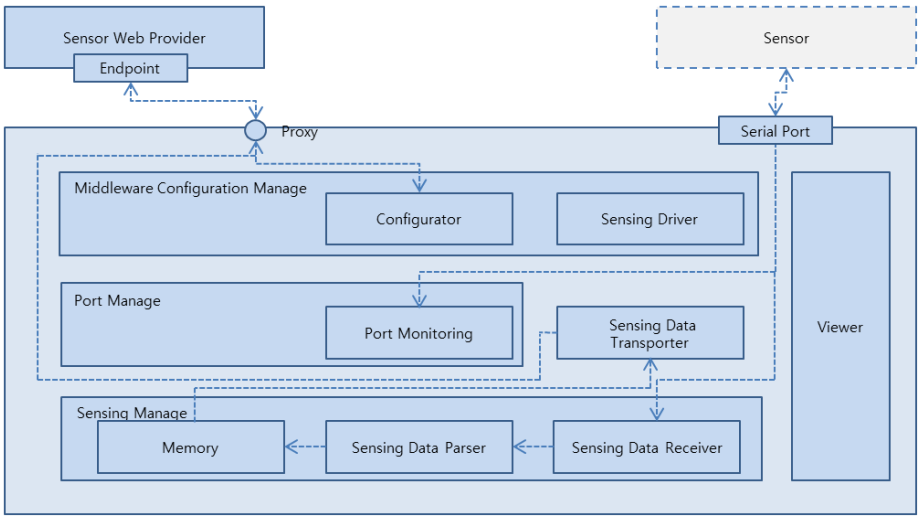

그림 30 센서 미들웨어 세부구조

그림 30은 센서 미들웨어의 세부구조이다. 여기에서 센서 미들웨어는 센서가 전달해 오는 센싱 데이터를 수집하여 센서웹 공급자에게 전송하여 저장하는 역할

을 담당한다. Configurator는 센서웹 공급자에게 구성 정보(IP주소, 접속 권한)를 요청하여 IP 주소의 해당과 서비스 공급자의 접속 권한을 검증한다. Sensing Driver가 다양한 센서의 센싱 데이터 포맷 정보를 갖고 수신되는 센싱 데이터를 이를 통해 파싱 처리한다. Port Monitoring은 미들웨어와 연결하는 포트의 상태를 실시간 감시하는 역할을 수행하고 Sensing Data Receiver는 센서 노드에서 수신되는 센싱 데이터를 접속하여 센싱 데이터 Parser를 통해 처리하여 메모리에 저장한다. Sensing Data Transporter가 Memory에 저장되는 센싱 데이터를 읽어와서 센서웹 공급자에게 전송한다.

2.4.4 센서웹 응용서버

응용서버 기반의 센서웹과 서비스 공급자 기반의 센서웹 시스템 구조가 차이가 있어서 본 절에서 각각 구조에 대해 기술한다. 센서웹의 응용서버는 센서웹중의 GIS부분(GIS 공급자, GIS 저작도구)하고 센서 정보 서비스(센서웹 공급자, 센서웹 저작도구, 센서 미들웨어, 센서) 부분을 연결하는 것이다.

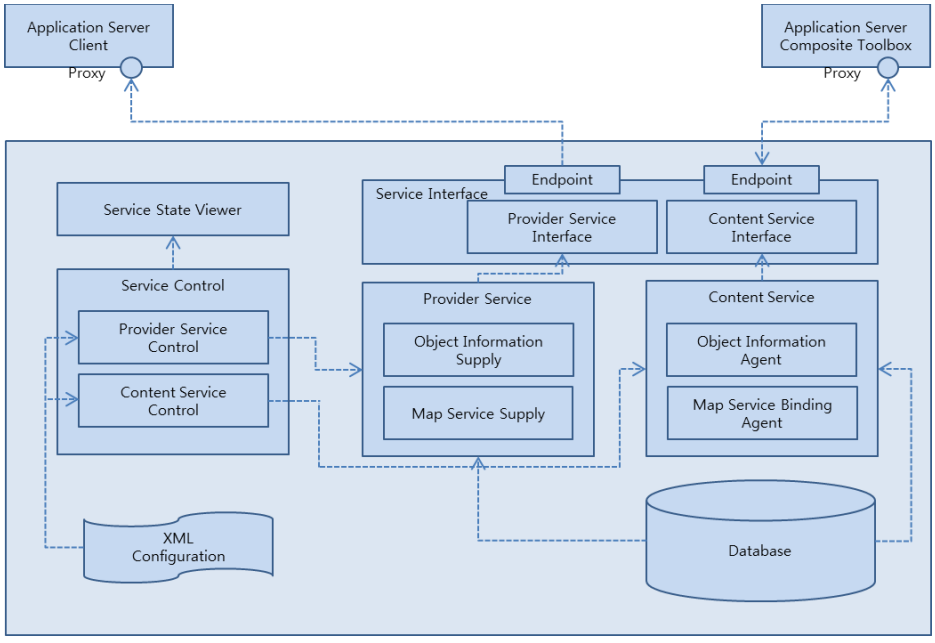

그림 31 서비스 공급자 기반 센서웹 응용서버 세부구조

그림 31은 서비스 공급자 기반 센서웹의 응용서버 세부구조이다. 응용서버는 Provider Service, Content Service로 구성된다. Provider Service는 Object Info Supply 및 Map Info Supply로 구성한다. Object Info Supply는 클라이언트에게 센서 오브젝트 정보(위치 정보, 타입, 공급 서비스 주소)를 제공한다. Map Info Supply는 지도 정보(실외 지도, 실내 지도, 빌딩, 층, 방 정보)를 제공한다. Content Service는 Object Info Agent 및 Map Info Supply로 구성한다. Object Info Agent는 응용서버 저작도구로서 센서 오브젝트 위치 정보관리 및 지도 서비스 바인딩 관리 서비스를 제공한다. Map Info Supply는 지도 정보를 제공한다. XML Configuration은 WCF Service의 동작 환경을 제공한다. Database는 지도 서비스 바인딩 정보와 오브젝트 정보를 저장한다.

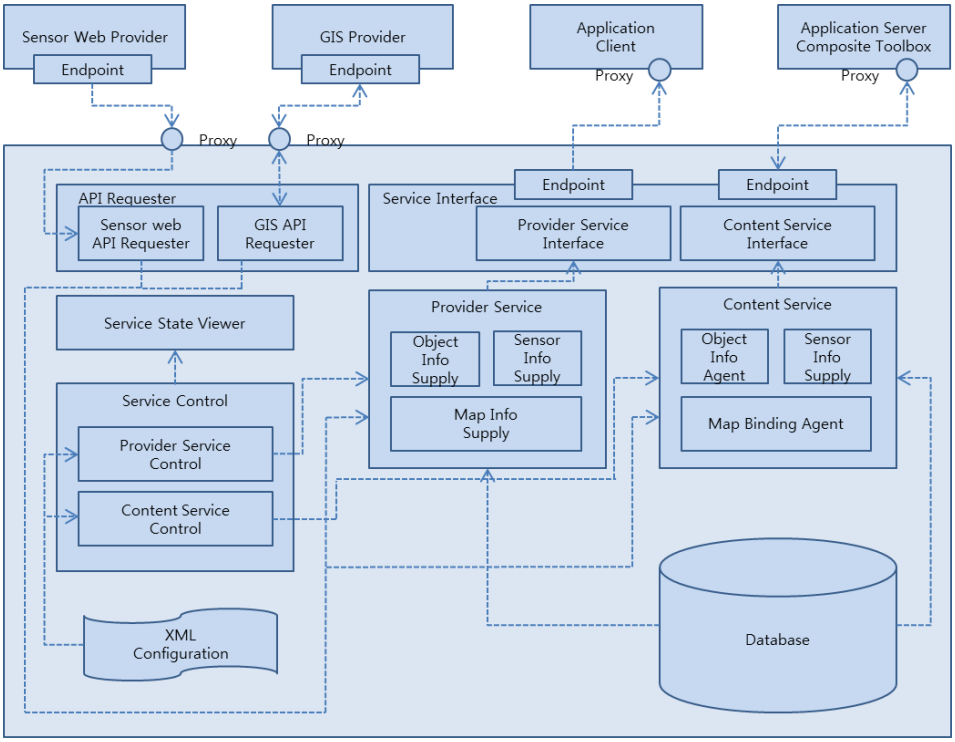

그림 32 응용서버 기반 센서웹 응용서버 세부구조

그림 32는 응용서버 기반 센서웹의 응용서버 세부구조이다. 응용서버는

Provider Service, Content Service로 구성된다. Provider Service는 Object Info Supply, Sensor Info Supply 및 Map Info Supply로 구성한다. Object Info Supply는 클라이언트에게 센서 오브젝트 정보(위치 정보, 타입, 공급 서비스 주소)를 제공한다. Sensor Info Supply는 센서의 세부 정보(센서 이름, 센싱 데이터, 센싱 타입, 센서 ID)를 제공한다. Map Info Supply는 지도 정보(실외 지도, 실내 지도, 빌딩, 층, 방 정보)를 제공한다. Content Service는 Object Info Agent, Sensor Info Supply 및 Map Info Supply로 구성한다. Object Info Agent는 응용서버 저작도구로서 센서 오브젝트 위치 정보관리 및 지도 서비스 바인딩 관리 서비스를 제공한다. Sensor Info Supply는 센서의 세부 정보(센서 이름, 센싱 데이터, 센싱 타입, 센서 ID)를 제공한다. Map Info Supply는 지도 정보(실외 지도, 실내 지도, 빌딩, 층, 방 정보)를 제공한다. XML Configuration은 WCF Service의 동작 환경을 제공한다. Database는 Map Service 바인딩 정보와 오브젝트 정보를 저장한다.

2.4.5 센서웹 클라이언트

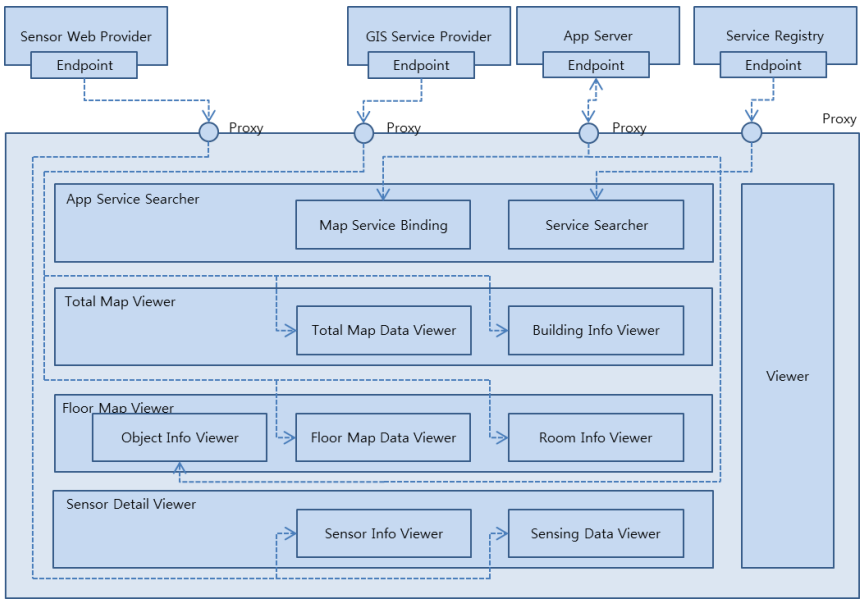

그림 33 서비스 공급자 기반 센서웹 클라이언트 세부구조

센서웹의 시스템 구조방안에 따라서 센서웹 클라이언트의 내부 구조가 조금 차이가 있다. 그림 33은 서비스 공급자 기반 센서웹 클라이언트의 세부구조이다. 클라이언트가 단순한 가시화를 제공한다. 처음에 App Service Searcher의 Service Searcher가 서비스 등록자에게 센서웹 응용서버를 검색하여 지정한 서비스를 접속하고 Map Service Binding이 응용서버에서 Map 서비스 정보를 요청하여 바인딩 한다. 다음에 Total Map Viewer는 GIS 공급자에서 전체지도 데이터 및 빌딩 정보를 요청한다. 관리자가 전체지도 뷰어에서 특정 빌딩의 층을 선택하여 Floor Map Viewer가 층 지도 데이터, 방 정보 및 오브젝트를 도시한다. 사용자가 센서 오브젝트를 선택하면 Sensor Info Viewer가 센서 이름, 아이디, 센싱 타입 등 속성을 출력하며 Sensing Data Viewer가 실시간 센싱 데이터를 출력한다.

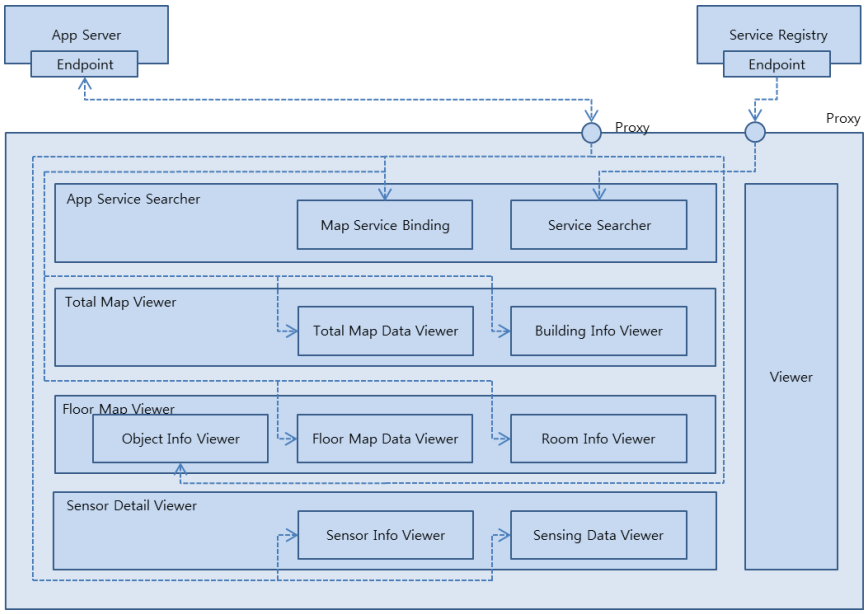

그림 34 응용서버 기반 센서웹 클라이언트 세부구조

그림 35는 응용서버 기반 센서웹 클라이언트의 세부구조이다. 클라이언트의 내부구조가 그림 34와 같지만 센서 정보와 지도 정보의 요청대상만 다르다. 응용서버 기반의 구동체웹 클라이언트가 응용서버를 통해 지도 정보와 구동체 세부정보를 공급자에 요청한다.

3. 실내 센서웹 시스템 구현

실내 센서웹 시스템은 센서를 관련하는 정보를 공급 및 관리하는 역할을 수행한다. 실내 센서웹 시스템은 센서웹 공급자, 센서웹 저작도구, 센서 미들웨어, GIS 공급자, GIS 저작도구, 응용서버, 응용서버 저작도구, 서비스 등록자와 클라이언트로 구성된다. GIS 공급자와 GIS 저작도구의 구현결과는 VI장의 2절에서 기술하여 있다. 응용서버, 응용서버 저작도구, 서비스 등록자와 클라이언트의 구현결과는 V장의 3절에서 기술하여 있다.

3.1 센서웹 공급자

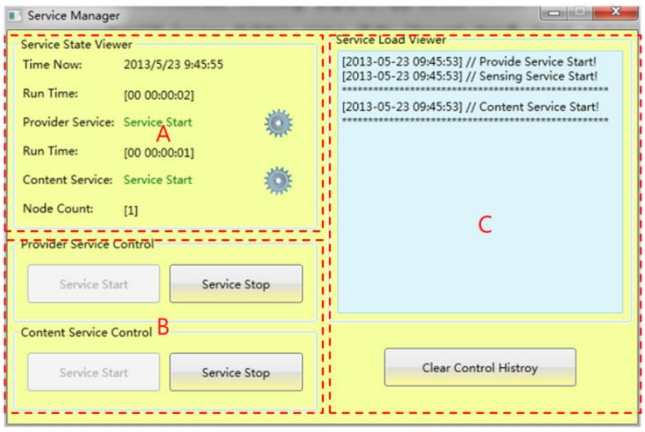

그림 35 센서웹 공급자 실행화면

그림 35는 센서웹 공급자 실행화면이다. 이는 주로 서비스 관리자에게 서비스의 켜짐, 닫음을 제어하는 역할을 제공한다. A구역은 서비스의 상태를 도시하는 것이다. Time Now는 시스템의 현재 시간을 알려준다. Run Time는 서비스 정상 동작하는 시간을 기록한다. Provider Service는 공급서비스의 동작상태를 표시한다. Content Service는 콘텐츠 서비스의 동작 상태를 표시한다. B구역은 서비스를 제어하는 역할을 담당한다. Provider Service Control는 공급서비스 접속 인터페이스를 꺼짐/켜짐을 제어한다. Content Service Control는 콘텐츠서비스 접속 인터페이스를 꺼짐/켜짐을 제어한다. C구역은 서비스 제어 기록 정보를 도시한다.

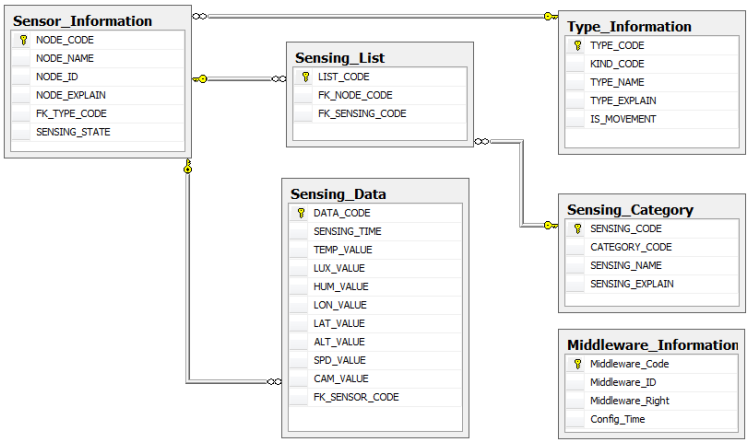

그림 36 센서웹 데이터베이스 관계도

그림 36은 센서웹의 데이터베이스 관계도 이다. Sensor_Information는 센서 정보를 저장하는 테이블이다. Sensing_List는 센서의 센싱 타입을 저장하는 테이블이다. Sensing_Category는 센싱의 타입의 세부 정보를 저장하는 테이블이다. Type_Information는 센서의 타입을 저장하는 테이블이다. Sensing_Data는 센싱 데이터를 저장하는 테이블이다.

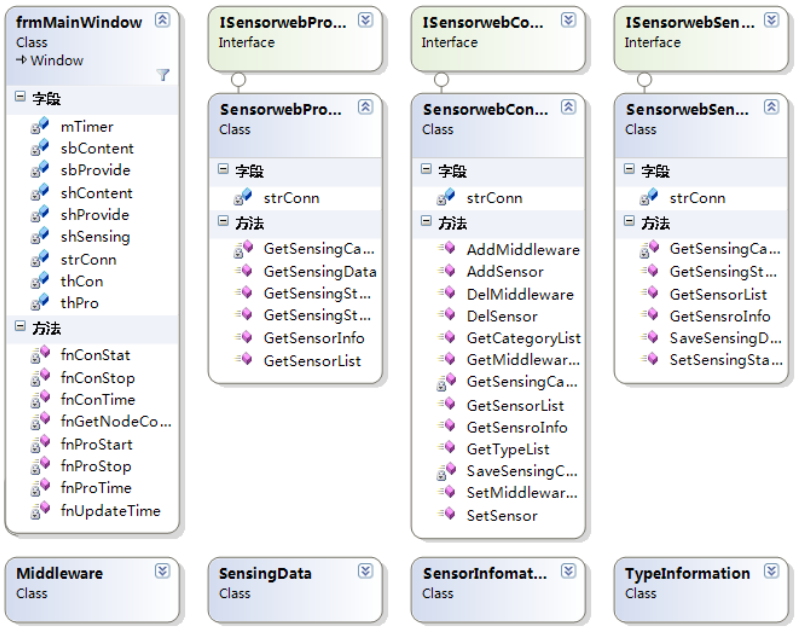

그림 37 센서웹 공급자의 클래스 다이어그램

그림 37은 센서웹 공급자의 클래스 다이어그램이다. frmMianWindow 클래스 서비스를 제어, 상태 정보 뷰어를 제공한다. ISensorwebProvider, ISensorwebContent, ISensorwebSensingService는 제공하는 서비스 인터페이스이며 외부 모듈들이 이를 통해 서비스에 접속한다. SensorwebProviderService는 센서 정보를 DB에서 읽어와서 공급하는 역할을 담당한다. SensorwebContentService는 외부에서 센서정보를 받아서 DB에 저장하는 역할을 담당한다. SensorwebSensingService는 외부에서 센싱 데이터를 수신하여 DB에 저장하는 역할을 수행한다. Middleware는 미들웨어 정보를 저장한다. SensingData는 센싱 데이터를 저장한다. SensorInforamtion는 센서의 정보를 저장한다. TypeInformation는 센서의 타입정보를 저장한다.

3.2 센서웹 저작도구

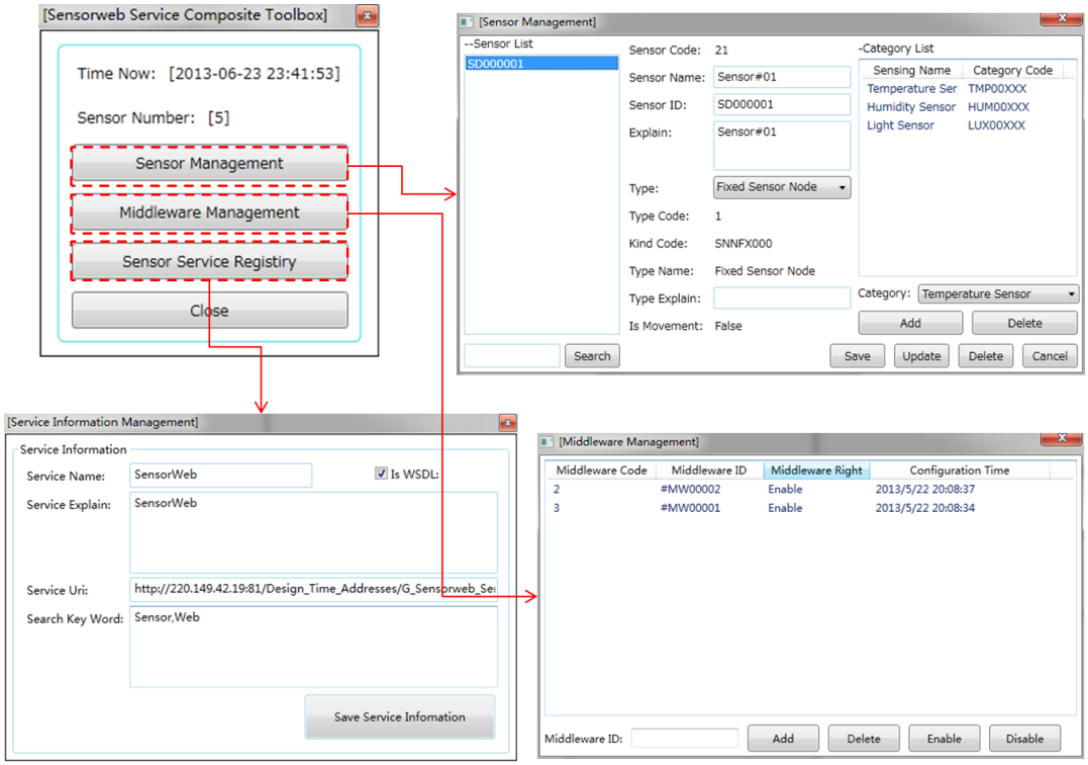

그림 38 센서웹 저작도구 실행화면

그림 38은 센서웹 저작도구의 실행화면이다. 그림 대로 센서웹 저작도구는 센서 정보 생성 및 관리, 미들웨어 정보 생성 및 관리, 서비스 정보 생성 및 관리 역할을 제공한다. Time Now는 시스템의 현재 시간을 표시한다. 센서 Number는 현재 서비스 공급자에서 등록되는 센서노드의 개수를 표시한다.

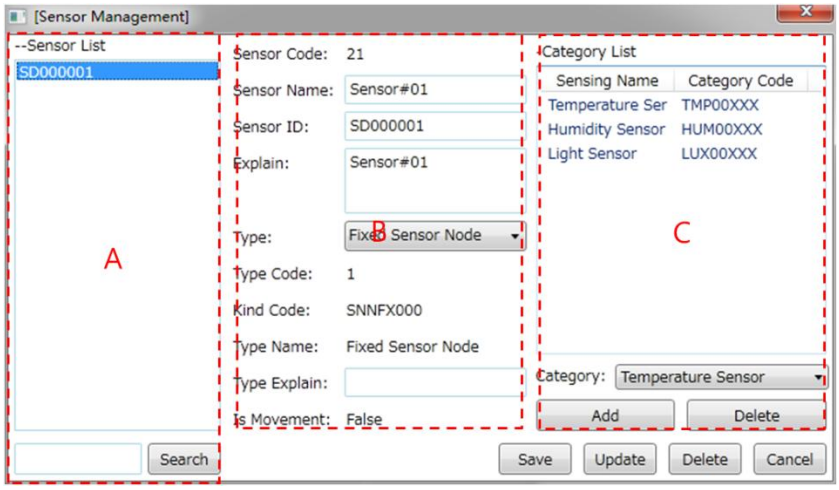

그림 39 센서 정보 관리 실행화면

그림 39는 센서 정보 관리 실행화면이다. A구역은 센서 리스트를 검색하는 역할을 담당한다. B구역은 센서 정보를 생성하는 역할을 담당한다. Sensor Code는 센서가 DB에서 저장하는 유일코드를 의미한다. Sensor Name는 센서노드의 이름을 표시한다. Sensor ID는 센서의 유일 식별자이다. Explain는 센서노도에 대한 기능 설명이다. Type는 센서의 이동/고정형을 지정한다. Type Code는 DB에서 Type정보에 해당하는 유일 코드이다. Type Name는 Type의 이름을 표시한다. Type Explain는 해당타입의 의미를 설명한다. Is Movement는 해당 Type의 이동이나 고정을 표시한다. C구역은 센서 카테고리 정보를 생성하는 역할을 담당한다. 센서 카테고리는 센서의 센싱 타입(온도, 습도, 조도 등의 다양한 센싱기능)을 지정한다.

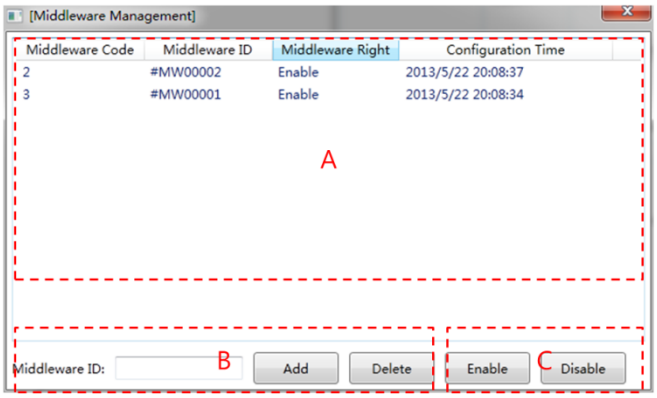

그림 40 미들웨어 관리 실행화면

그림 40은 미들웨어 관리 실행화면이다. A구역은 미들웨어 리스트를 도시하는 역할을 담당한다. 여기서 서비스 공급자의 DB에 등록되는 미들웨어 정보(미들웨어 코드, ID, 접속권한, 등록하는 시간)를 나열한다. B구역은 미들웨어 정보를 추가 및 삭제기능을 제공한다. C구역은 미들웨어의 서버 접속 권한을 관리하는 역할을 담당한다.

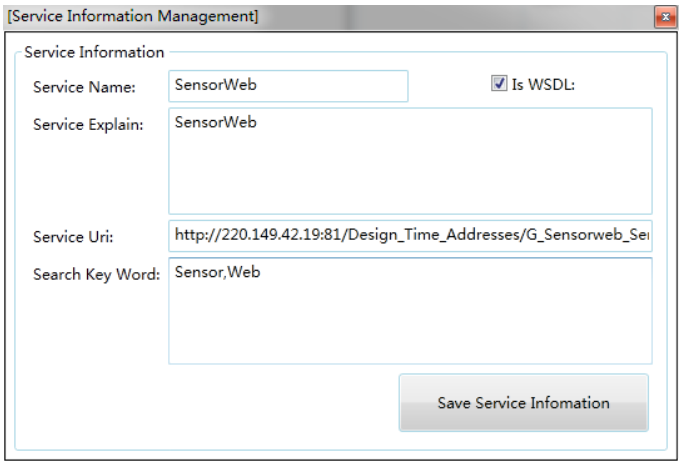

그림 41 서비스 정보 관리 실행화면

그림 41은 서비스 정보를 관리하는 실행화면이다. 이는 서비스 정보를 생성하여 서비스 등록자에게 등록하는 역할을 수행한다. 서비스 정보는 서비스 이름, WSDL제공 여부, 서비스 설명, 서비스 접속 주소, 서비스 검색 키워드를 포함한

다. Service Name는 서비스 이름을 입력하는 구역이다. Is WSDL는 서비스 공급자의 WSDL(Web Services Description Language)의 제공여부를 관리한다. Service Explain는 서비스에 대한 설명이다. Service Uri는 서비스 공급자의 접속 주소이다. Search Key Word는 서비스 공급자를 검색하는 키워드를 지정한다. Save Service Information는 서비스 공급자의 정보를 서비스 등록자에게 등록시키는 버튼이다.

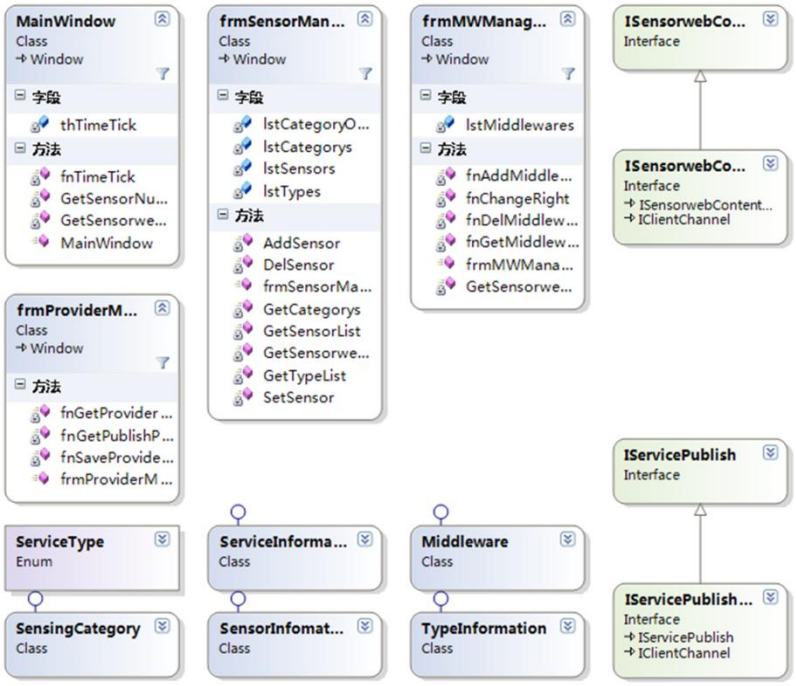

그림 42 센서웹 저작도구 클래스 다이어그램

그림 42는 센서웹 저작도구의 클래스 다이어그램이다. MainWindow, frmSesnorManagement, frmMiddlewareManagement, frmProviderManagement는 실제 화면에 표시되는 WPF Form을 상속받는 클래스를 나타낸다. MainWindow는 다른 창으로 넘어가는 가시화를 제공한다. frmSesnorManagement는 센서 정보를 관리하는 뷰어를 제공한다. frmMiddlewareManagement는 미들웨어를 관리 및 전속 권한을 지정하는 기능을 담당한다. frmProviderManagement는 서비스 공급자 정보 생성 및 관리하는 기능을 담당한다. ISensorwebContentService 및 IPublishService는 WCF Service

를 참조하기 위해 필요한 인터페이스이다. Service Type은 서비스 타입(응용서버, 센서웹, 구동체웹, GIS)을 정의한다. Service Information는 서비스 정보를 저장하는 클래스이다.

3.3 센서 미들웨어

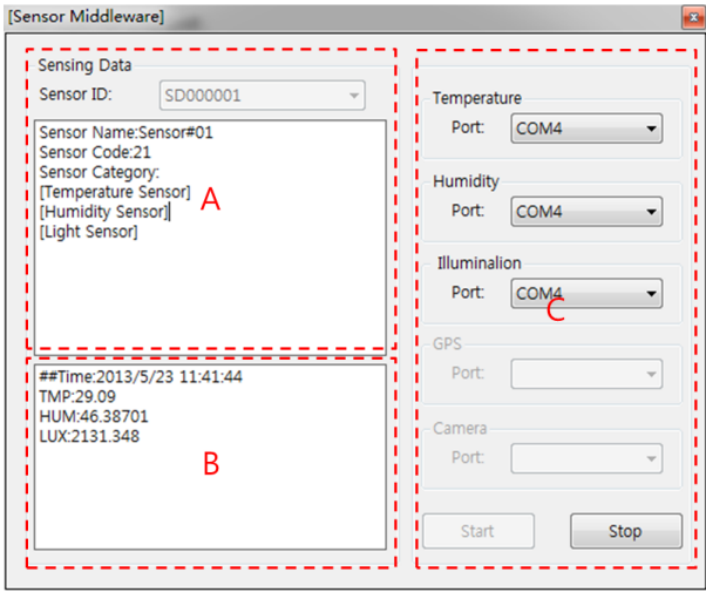

그림 43 센서 미들웨어 실행화면

그림 43은 센서 미들웨어의 실행화면이다. 센서 미들웨어는 센서와 연결하고 센싱 데이터를 수집하는 역할을 수행한다. A구역은 연결된 센서의 정보를 도시하는 기능 제공한다. 센서 ID는 수신하는 센서의 ID를 지정하여 해당센서의 정보(센서 이름, 코드, 센싱타입)를 표시한다. B구역은 센싱 데이터를 실시간 표시하는 기능을 수행한다. 센싱 데이터는 센싱 시간과 해당 센서의 측정값을 포함한다. C는 미들웨어의 센서 연결 포트를 설정 및 센싱 수신 여부를 제어하는 역할을 담당한다.

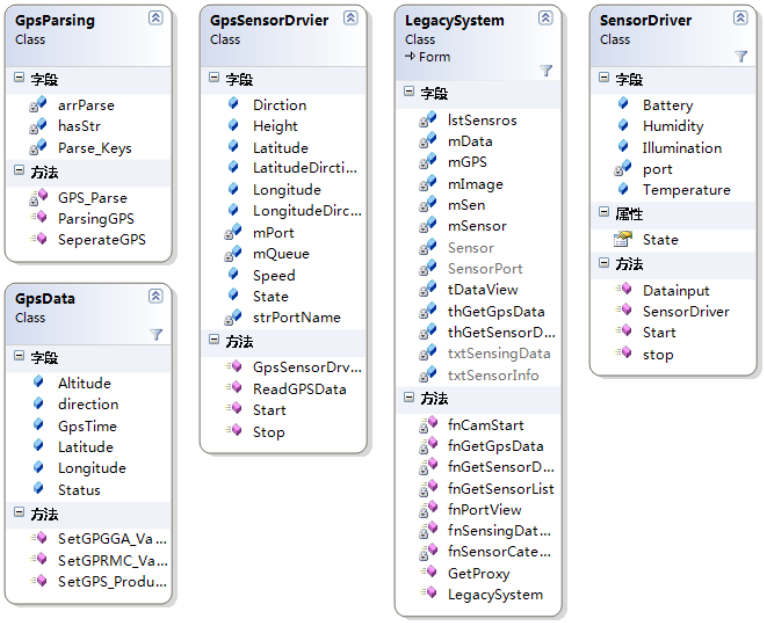

그림 44 센서 미들웨어 클래스 다이어그램

그림 44는 센서 미들웨어 클래스 다이어그램이다. GpsParsing은 GPS센서에서 수신된 GPS 센싱 Stream은 특정한 포맷으로 변환하는 기능을 제공한다. GpsData는 GPS 센싱 데이터를 저장하는 클래스이다. GpsSensorDriver 및 SensorDriver는 GPS 센서 및 종합센서를 구동하는 메서드를 제공한다. LegacySystem은 센서를 구동, 센서 정보, 센싱 데이터의 뷰어를 제공한다.

4. 성능 분석 및 평가

4.1 실험환경

실험환경은 표 8과 같다. 실험은 실내 센서웹 시스템의 주요 기능에서 성능 분석을 한다. 모든 성능분석은 해당 주요모듈 별로 20회에 걸쳐 수행시간을 측정하고 분석한다.

표 8 실험환경

Division	Detail
Operating System	Microsoft Windows 7(X64)
Development Environment	.Net Framework 3.5, 4.0
Development Tool	Visual Studio .Net 2010
Programming Language	C#, XMAL, XML
DBMS	Microsoft SQL Server 2008 R2
Hardware	CPU: Intel® Core™ i3-3220 @ 3.30GHz RAM: 4GB Graphics: NVIDIA GeForce GT 440

실내 센서웹 시스템은 주로 센서 세부 정보 및 센싱 데이터를 요청하여 응답하는 수행시간을 측정한다. 그리고 센서웹 공급자의 데이터베이스에 저장된 센서 노드 세부 정보 및 실시간 센싱 데이터를 요청하여 응답하는 수행시간을 측정한다.

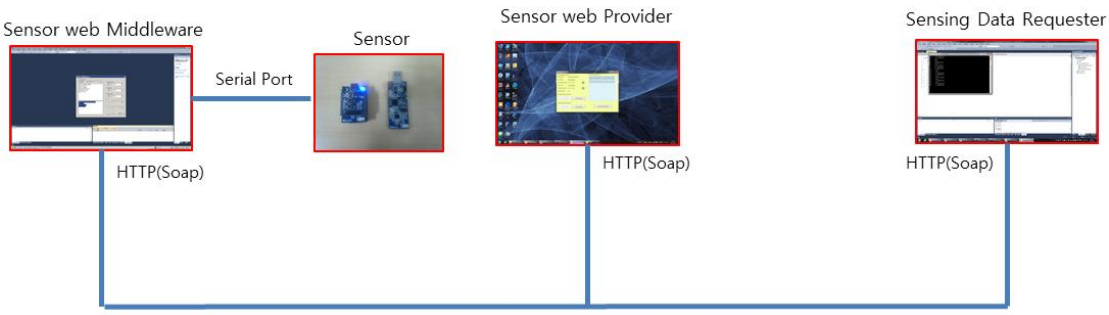

그림 45 센서 웹의 실험 네트워크 구성

센서 웹의 실험환경의 네트워크 구성은 그림 45와 같다. 실험을 위해 센서웹 공급자를 위한 데스크탑 서버 1대, 센서 미들웨어를 구동하기 위한 데스크탑 1대, 센싱 데이터 요청자를 구동하기 위한 노트북을 1대를 사용한다.

표 9 센서 웹의 실험 네트워크 환경

Module Name	Address
센서웹 서비스 공급자	http://220.149.42.19:81/Design_Time_Addresses/G_Sensorweb_Service_Provider/SensorwebProviderService/
센서웹 미들웨어	117.17.102.28
센싱 데이터 요청자	117.17.102.197

4.2 센싱 데이터 질의 성능평가

본 논문에서 센서 웹 시스템은 실내 환경 정보를 수집과 공급하는 역할을 담당한다. 그래서 센서 노드가 센싱하고 미들웨어를 통해 서비스 공급자의 DB에서 저장하는 과정과 클라이언트가 서비스 공급자에서 센싱 데이터를 요청하는 과정으로 구성한다. 다음에 두 과정을 분리하여 성능 분석한다.

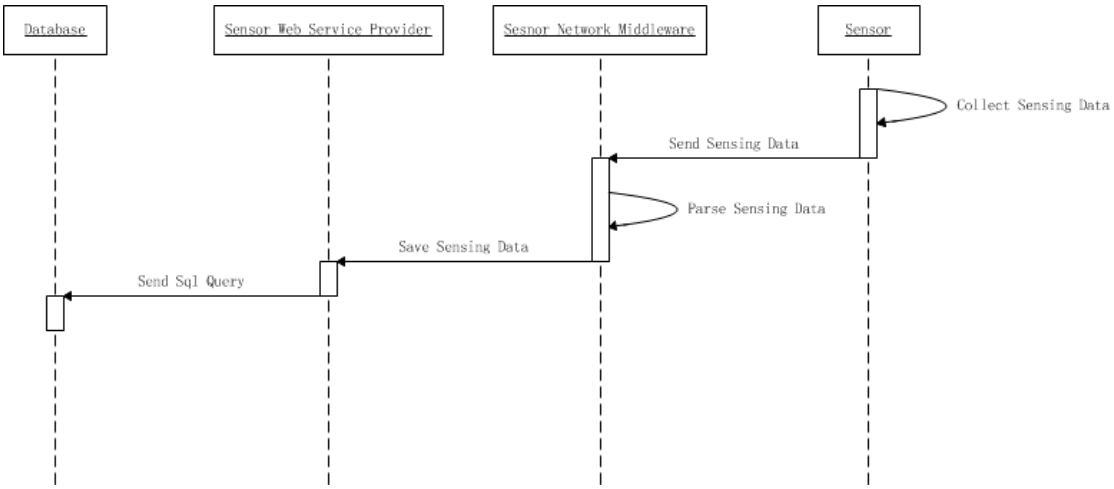

그림 46 센싱 데이터 생성하고 저장과정

그림 46은 센싱 데이터를 생성하고 저장과정이다. 본 실험은 1개, 5개, 50개 센서 노드 환경을 별도 설치하여 동시에 센싱하고 미들웨어를 통해 서비스 공급자의 DB에서 저장을 수행하는 시간을 측정하여 비교한다. 그리고 센서의 다중 센싱 타입(온도, 습도, 조도 등)을 고려하여 분석한다.

표 10 실험 결과(센싱과정)

Sensor / Record	1 Sensor	5 Sensor	50 Sensor
1 Record	19.4ms	50.3ms	245.05ms
2 Record	20.4ms	54.79ms	272.25ms
3 Record	23.75ms	48.1ms	195.15ms
4 Record	25.25ms	57.29ms	155.64ms

표 10은 센싱 데이터를 생성하고 센서웹 공급자에 저장하는 과정에 대해 실험되는 결과이다. 가로는 동작하는 센서 개수를 의미하고 세로는 센싱 타입의 개수를 의미한다.

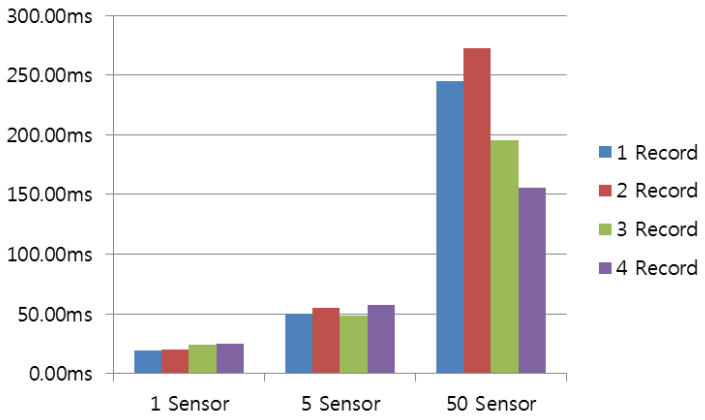

그림 47 실험 결과 비교(센싱과정)

그림 47은 표 10에 대한 실험결과 비교 그래프이다. 이 그림을 보면 센서의 개수 증가에 따라서 센서웹 공급자가 처리하는 시간이 늘어진다. 1 센서가 1센싱 데이터를 생성하여 센서웹의 DB에 저장하는 소요시간은 19.4ms 이며, 2센싱 데이터 같은 경우에는 20.4ms 이며, 3센싱 데이터 같은 경우에는 23.75ms이며, 4센싱 데이터 같은 경우에는 25.25ms이다. 5 센서가 1센싱 데이터를 생성하여 센서웹의 DB에 저장하는 소요시간은 50.3ms 이며, 2센싱 데이터 같은 경우에는

54.79ms 이며, 3센싱 데이터 같은 경우에는 48.1ms이며, 4센싱 데이터 같은 경우에는 57.29ms이다. 50 센서가 1센싱 데이터를 생성하여 센서웹의 DB에 저장하는 소요시간은 245.05ms 이며, 2센싱 데이터 같은 경우에는 272.25ms 이며, 3센싱 데이터 같은 경우에는 195.15ms이며, 4센싱 데이터 같은 경우에는 155.64ms이다. 그러나 센싱 타입의 개수는 실제 네트워크의 환경은 실험결과에 대해 거의 영향을 미치지 않다.

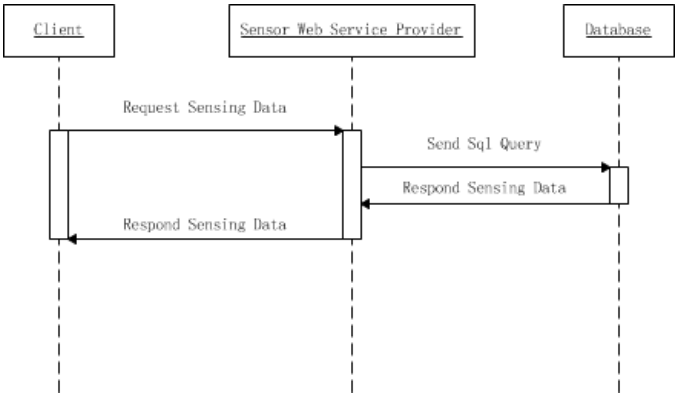

그림 48 센싱 데이터 요청과정

그림 48은 센서웹에서 센싱 데이터 요청하는 과정이다. 본 실험은 1개, 5개, 50개 클라이언트 별도 설치하여 동시에 서비스 공급자에게 센싱 데이터를 요청하는 시간을 측정하여 비교한다. 그리고 센서의 다중 센싱 타입(온도, 습도, 조도 등)을 고려하여 분석한다.

표 11 실험 결과(센싱 데이터 요청과정)

Client / Record	1 Client	5 Client	50 Client
1 Record	369.46ms	389.04ms	685.26ms
2 Record	367.13ms	396.27ms	889.94ms
3 Record	400.61ms	418.37ms	674.39ms
4 Record	410.04ms	415.56ms	698.54ms

표 11은 클라이언트가 센싱 데이터 요청과정에 대해 실험한 결과이다. 가로는 동작하는 클라이언트 개수를 의미하고 세로는 센싱 타입의 개수를 의미한다.

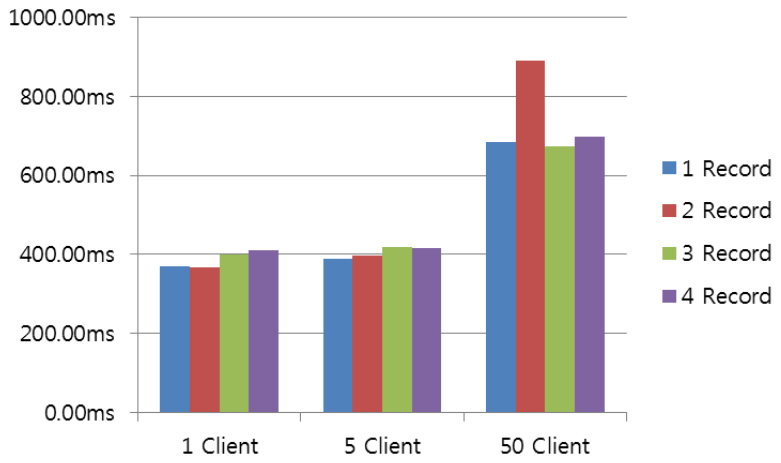

그림 49 실험 결과 비교(센싱 데이터 요청과정)

그림 49는 표 11에 대한 실험결과 비교 그래프이다. 1개 클라이언트가 1센싱 데이터를 센서웹의 DB에 요청하여 응답하는 소요시간은 369.46ms 이며, 2센싱 데이터 같은 경우에는 367.13ms 이며, 3센싱 데이터 같은 경우에는 400.61ms이며, 4센싱 데이터 같은 경우에는 410.04ms이다. 5개 클라이언트가 1센싱 데이터를 센서웹의 DB에 요청하여 응답하는 소요시간은 389.04ms 이며, 2센싱 데이터 같은 경우에는 396.27ms 이며, 3센싱 데이터 같은 경우에는 418.37ms이며, 4센싱 데이터 같은 경우에는 415.56ms이다. 50개 클라이언트가 1센싱 데이터를 센서웹의 DB에 요청하여 응답하는 소요시간은 685.26ms 이며, 2센싱 데이터 같은 경우에는 889.94ms 이며, 3센싱 데이터 같은 경우에는 674.39ms이며, 4센싱 데이터 같은 경우에는 698.54ms이다. 그래프에 의해 센싱 데이터를 요청하는 클라이언트의 개수 증가에 따라서 센싱 데이터를 요청하는 소요시간이 늘어지고 센싱 타입의 개수는 실제 네트워크 환경은 실험결과에 대해 거의 영향을 미치지 않다.

4.3 서비스 공급자 및 응용서버 기반 센서웹 비교 성능평가

본 절에서 서비스 공급자 기반 IOT 시스템의 클라이언트가 해당 센서웹 공급자에서 센싱 데이터를 요청하여 수신하는 시간이 응용서버 기반 IOT 시스템의 클라이언트가 해당 센서웹 공급자에서 센싱 데이터를 요청하여 수신하는 시간과 비교한다. 테스트를 위해 제주대학교 공대4호관의 D423호실에 설치된 센서ID는 'SD000001'이라는 종합 센서를 대상으로 성능 분석 진행한다. 그리고 여러 대 클라이언트를 준비하여 1대, 5대와 25대의 클라이언트가 별도 동시에 센싱 데이터를 요청하는 시간을 측정하여 분석한다.

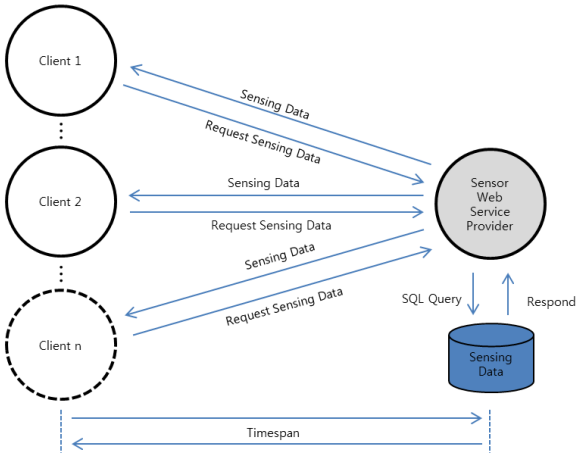

그림 50 서비스 공급자 기반 센싱 데이터 요청

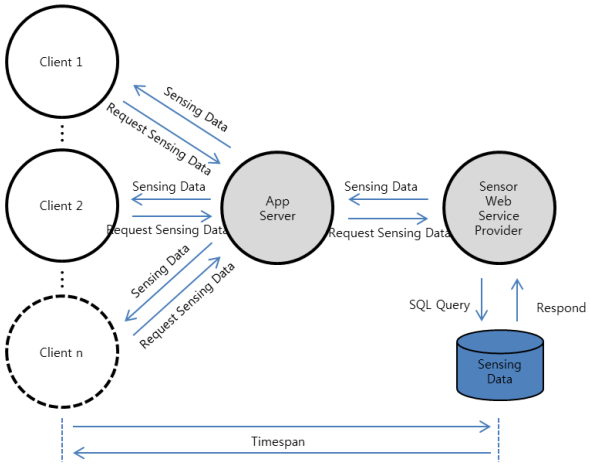

그림 51 응용서버 기반 센싱 데이터 요청

그림 50은 서비스 공급자 기반의 IOT 시스템에서 클라이언트가 센싱 데이터를 요청하는 과정이다. 1대, 5대와 25대 클라이언트가 동시 접속경우는 별도 20회 실험을 진행한다.

그림 51은 응용서버 기반의 IOT 시스템에서 클라이언트가 센싱 데이터를 요청하는 과정이다. 1대, 5대와 25대 클라이언트가 동시 접속경우는 별도 20회 실험을 진행한다.

표 12 센싱 데이터 요청시간

	Base on App Server			Base on Service Provider		
	1 Client	5 Client	25 Client	1 Client	5 Client	25 Client
1	22.00ms	77.81ms	279.60ms	18.99ms	53.20ms	191.50ms
2	24.00ms	75.40ms	283.40ms	17.99ms	60.21ms	187.62ms
3	24.02ms	92.01ms	262.00ms	22.01ms	57.61ms	167.62ms
4	29.00ms	75.21ms	294.48ms	16.00ms	60.81ms	181.70ms
5	21.01ms	69.01ms	272.04ms	16.00ms	54.21ms	167.26ms
6	20.99ms	72.80ms	308.40ms	20.01ms	55.21ms	181.46ms
7	20.00ms	71.01ms	263.24ms	15.00ms	58.60ms	188.62ms
8	25.00ms	76.41ms	347.76ms	18.01ms	56.00ms	212.30ms
9	39.00ms	78.00ms	271.60ms	18.00ms	57.01ms	184.14ms
10	22.01ms	77.20ms	278.20ms	16.00ms	54.01ms	176.94ms
11	22.00ms	108.21ms	253.12ms	17.00ms	59.20ms	189.58ms
12	22.00ms	79.61ms	247.24ms	15.00ms	56.40ms	174.42ms
13	21.01ms	81.00ms	278.08ms	19.01ms	53.00ms	188.78ms
14	20.99ms	88.41ms	276.12ms	17.00ms	53.41ms	208.02ms
15	21.01ms	75.01ms	290.04ms	18.01ms	54.41ms	198.86ms
16	24.02ms	72.81ms	272.60ms	15.00ms	56.00ms	183.22ms
17	20.00ms	75.20ms	280.80ms	18.00ms	67.81ms	177.10ms
18	20.99ms	0.21ms	331.16ms	18.01ms	60.20ms	182.94ms
19	23.00ms	85.21ms	279.52ms	18.00ms	53.00ms	196.50ms
20	21.01ms	76.81ms	273.20ms	15.00ms	55.21ms	169.38ms

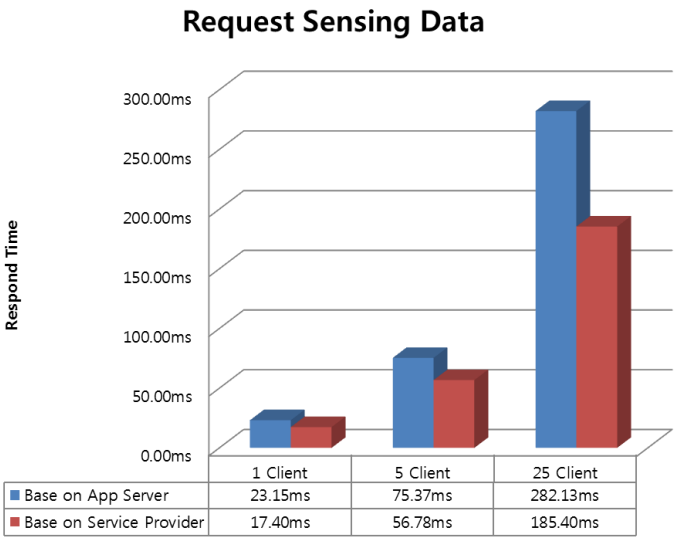

그림 52 센싱 데이터 평균 요청시간 비교

표 12는 그림 50과 그림 51에 대해 성능분석 결과이다. 응용서버 기반의 IOT 시스템에서 클라이언트가 간접방식으로 센서웹 공급자와 접속하기 때문에 직접 접속방식을 사용하는 서비스 공급자 기반의 IOT 시스템보다는 센싱 데이터를 요청하는 속도가 느리다.

그림 52은 센싱 데이터 평균 요청하는 시간 비교 그래프이다. 1개 클라이언트가 센싱 데이터 요청 같은 경우에는 응용서버 기반의 시스템 구조를 사용하는 시스템은 평균 23.14ms 걸리며, 서비스 공급자 기반의 시스템 구조를 사용하는 시스템은 평균 17.40ms 걸린다. 5개 클라이언트가 동시에 센싱 데이터를 요청을 같은 경우에 응용서버 기반의 시스템 구조를 사용하는 시스템은 75.37ms 걸리며, 서비스 공급자 기반의 시스템 구조를 사용하는 시스템은 56.78ms 걸린다. 25개 클라이언트가 동시에 센싱 데이터를 요청을 같은 경우에 응용서버 기반의 시스템 구조를 사용하는 시스템은 282.13ms 걸리며, 서비스 공급자 기반의 시스템 구조를 사용하는 시스템은 185.40ms 걸린다.

IV. 개방형 API 기반의 실내 구동체웹

1 실내 구동체웹 시스템 설계

1.1 서비스 공급자 기반 실내 구동체웹

본 논문의 III장 1절에서 센서웹의 시스템구조가 서비스 공급자 기반과 응용서버 기반의 두 가지 방안을 제시한다. 따라서 본 절에서 센서웹 시스템과 같은 구조를 사용하여 구동체웹 시스템을 설계한다.

1.1.1 서비스 공급자 기반 실내 구동체웹 시스템의 인터페이스 프로토콜

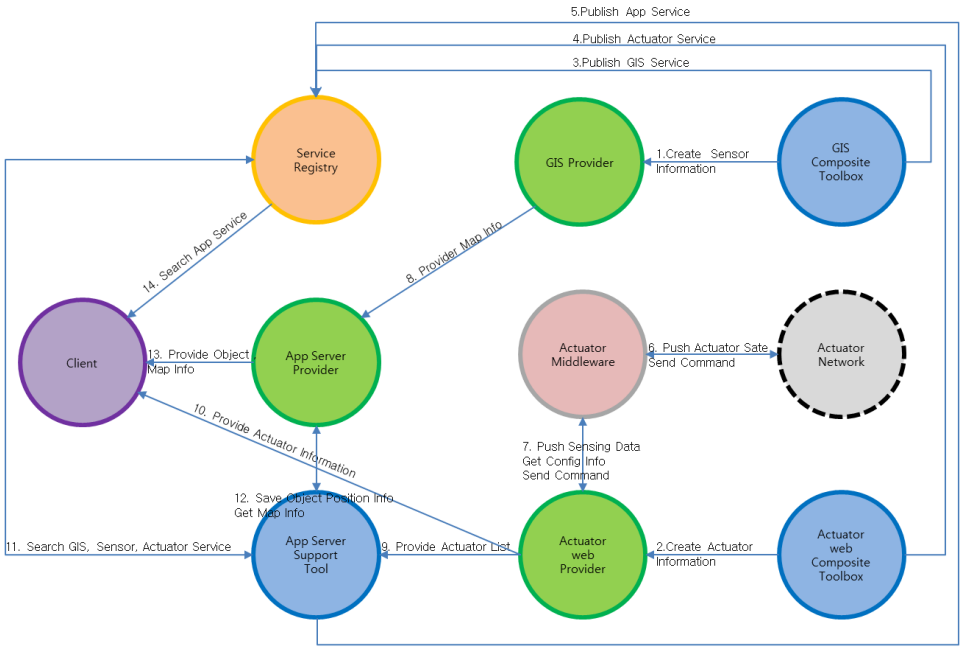

그림 53 서비스 공급자 기반의 구동체웹 구조

그림 53는 서비스 공급자 기반 실내 구동체웹 시스템 내부 모듈간의 API 구성도이다. 구동체 웹 시스템은 주로 구동체 노드 정보관리, 공급 그리고 구동체 제어 및 동작상태를 감시하는 역할을 담당한다.

표 13 서비스 공급자 기반 구동체웹 시스템의 API

Module Name	API Name	Description		Applied At
		Type	Operation	
Service Registry	Service Publish Interface Protocol	Service Information	Write	3, 4, 5
	Service Search Interface Protocol	Service Information	Search, Read	11, 15
GIS Service Provider	GIS Provider Interface Protocol	Map Information	Read	8
	GIS Content Interface Protocol	Map Information	Read, Write	1
Actuator web Service Provider	Actuator web Service Provider Interface Protocol	Actuator List	Read, Search	9, 10
		Actuator Information	Read	9
		Control	Write	
	Actuator web Service Content Interface Protocol	Actuator List	Read, Search	2
		Actuator Information	Write	2
App Server	App Service Provider Interface Protocol	Map Information	Read	13
		Object Information	Read	13
	App Service Content Interface Protocol	Map Information	Read	12
		Object Information	Read, Write	12

표 13은 구동체웹 시스템에 내부 존재하는 API들이다. 위의 표와 그림 53를 같이 참조 하면 각 모듈간 어떤 API를 통해 연동하는 것을 쉽게 이해할 수 이다. Service Publish API는 서비스 등록하는 역할을 수행하고, Service Search API는 서비스 정보 검색하는 역할을 수행하고, GIS Service Provider API는 지도정보 공급하는 역할을 수행하고, GIS Content API는 지도정보 관리하는 역할을 수행, Actuator Web Provider API는 구동체 정보 및 구동체 제어하는 역할을 수행하고, Actuator Web Content API는 구동체 정보 관리하는 역할을 수행하고, App Service Provider API는 지도정보 및 오브젝트 정보 공급하는 역할을 수행하고, App Service Content API는 지도 서비스 공급, 오브젝트 정보를 관리하는 역할을 수행한다.

1.1.2 환경 설정 단계

그림 54는 서비스 공급자 기반 실내 구동체웹 시스템의 환경 설정 단계이다. 실내 구동체웹 시스템의 환경 설정 단계는 서비스의 콘텐츠를 제대로 클라이언트에게 공급을 수행하기 위해 생성한다.

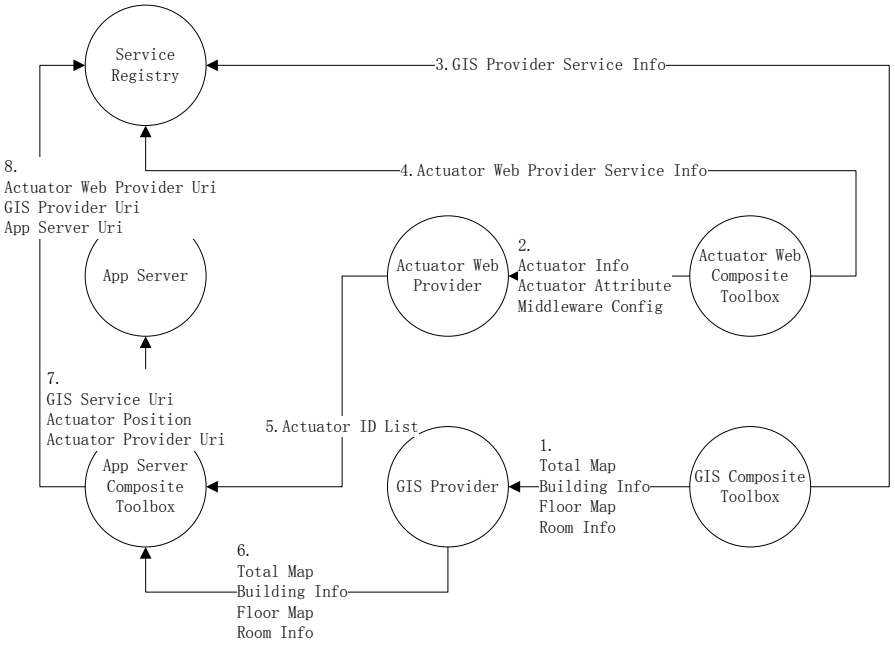

그림 54 서비스 공급자 기반 실내 구동체웹 시스템 환경 설정 단계

Total Map은 실외 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Building Info는 빌딩의 이름, 위치 정보, 층 이름을 포함한다. Floor Map은 실내 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Room Info는 방의 이름, 방 위치를 포함한다. Actuator Info는 구동체 노드의 이름, 구동체 ID를 포함한다. Actuator Attribute는 구동체의 속성 타입 정보를 의미한다. Middleware Config는 구동체 미들웨어의 ID, IP주소과 접속 권한 정보를 포함한다. Service Info는 각각 서비스 공급자의 서비스 이름, 검색 키워드, 접속 주소, 서비스 유형을 포함한다. Actuator List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 구동체 ID들을 의미한다. GIS Service Uri는 GIS 공급자의 접속주소를 의미한다. Actuator Position는 구동체의 지도상의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다.

GIS 서비스 저작도구는 지도 관련 데이터를 외부에서 GIS Content API를 통해 GIS 공급자의 DB에 저장한다. 구동체웹 저작도구는 각각 구동체 노드의 정보를 Actuator Web Content API를 통해 생성하여 DB에 저장하여 미들웨어의 배치 정보를 생성한다. GIS 서비스 저작도구가 GIS 공급 서비스의 정보를 서비스 등록

자에게 등록한다. 구동체웹 저작도구가 Actuator Web Provider Service의 정보를 서비스 등록자에게 등록한다. 구동체웹 공급자가 제공하는 센서 리스트 정보를 응용서버 저작도구에게 제공한다. GIS 공급자가 제공하는 지도 데이터를 응용서버 저작도구에게 제공한다. 응용서버 저작도구가 각각 센서 노드와 지도의 위치 정보를 생성하여 저장한다. 응용서버 저작도구가 응용서버의 서비스 정보를 등록한다.

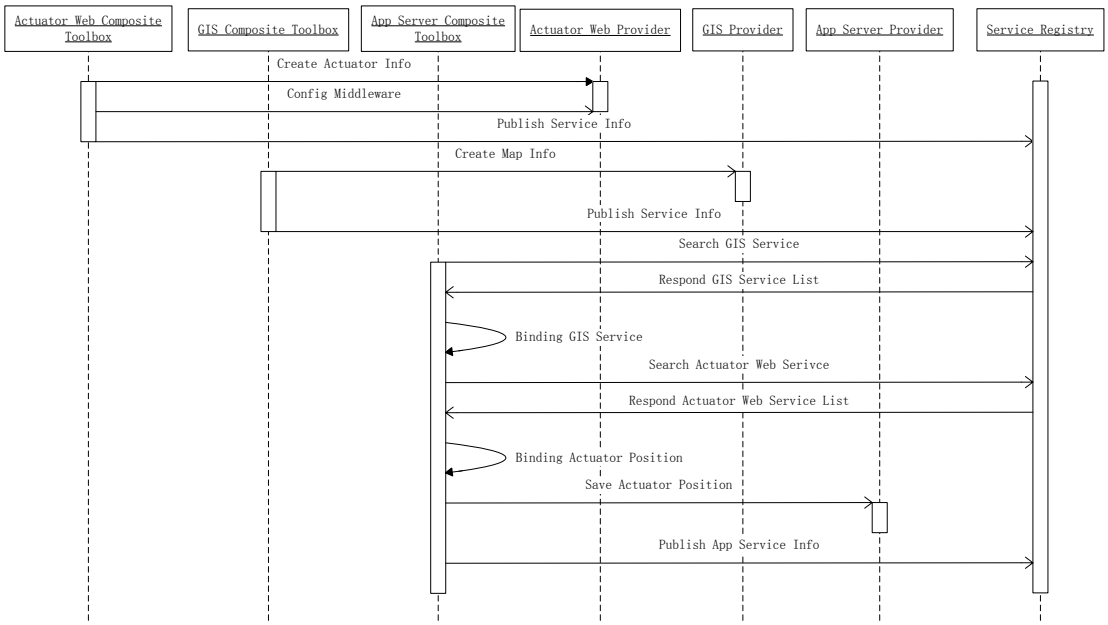

그림 55 서비스 공급자 기반 실내 구동체웹 시스템 환경 설정 단계 시퀀스

그림 55는 서비스 공급자 기반 실내 구동체웹 시스템의 환경 설정 단계 시퀀스 다이어그램이다. 위의 과정은 구동체웹 저작도구, GIS 서비스 저작도구, 응용서버 저작도구, 구동체웹 공급자, GIS 공급자, 응용서버 서비스 공급자와 서비스 등록자를 구성한다.

이 단계가 실행할 때 구동체웹 저작도구는 구동체 오브젝트의 정보를 생성하여 구동체웹 공급자에 저장하며 구동체 미들웨어의 배치 정보를 생성하여 구동체웹 공급자에게 등록한다. 다음에 구동체웹 공급자 서비스 정보를 서비스 등록자

에게 등록한다. 그리고 GIS 서비스 저작도구가 지도정보를 생성하여 GIS 공급자에 전송하고 GIS 공급 서비스 정보를 서비스 등록자에게 등록한다. 응용서버 저작도구는 서비스 등록자에 GIS 서비스 정보를 검색하여 관리자가 자기가 원하는 지도 엔진 서비스를 선택하여 바인딩한다. 지도 서비스를 바인딩하여 서비스 등록자가 제공하는 구동체웹 공급자 서비스 정보를 뽑아서 이 구동체웹 공급자가 제공하는 구동체 노드 정보를 보여준다. 관리자가 선택된 구동체 노드를 지도에서 하나씩 위치를 바인딩하고 응용서버 서비스 정보를 서비스 등록자에 저장한다.

1.1.3 검색 단계

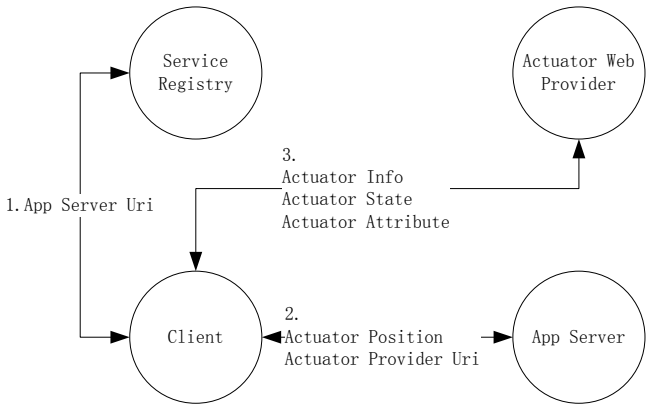

그림 56 서비스 공급자 기반 실내 구동체웹 시스템 검색 단계

그림 56은 서비스 공급자 기반 실내 구동체웹 시스템 검색 단계이다. 구동체웹 시스템의 검색 단계는 클라이언트가 서비스 등록자에게 쉽게 응용서버 및 센서 노드 및 구동체 노드를 찾을 수 있도록 한다.

Actuator Info는 구동체 노드의 이름, 구동체 ID를 포함한다. Actuator State는 구동체의 동작상태 정보를 의미한다. Actuator Attribute는 구동체의 작업 속성을 의미한다. Actuator Position는 구동체가 지도상의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

클라이언트는 사용자가 입력하는 서비스검색 키워드를 통해 서비스 등록자에게 서비스 리스트를 요청한다. 클라이언트가 응용서버를 접속하여 센서의 위치 정보를 요청한다. 클라이언트가 구동체의 ID를 통해 구동체 정보를 요청한다. 클라이언트가 구동체의 ID를 통해 구동체 정보를 요청한다.

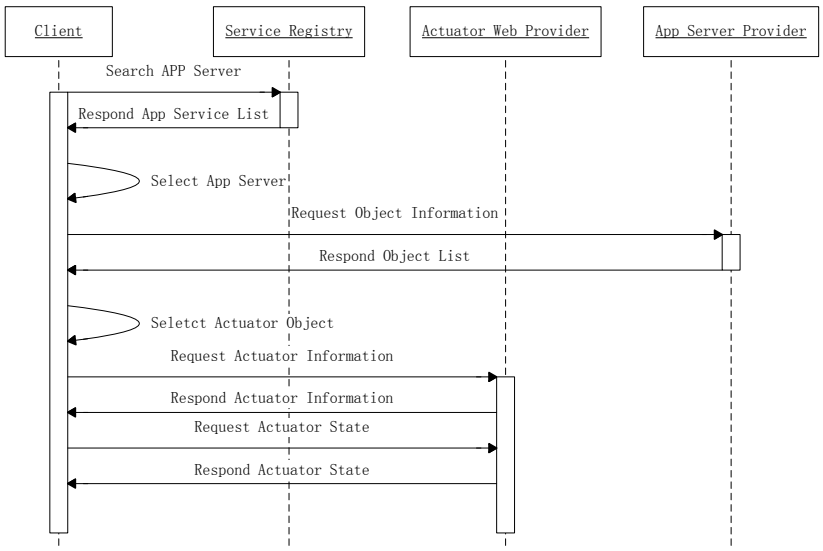

그림 57 서비스 공급자 기반 실내 구동체웹 시스템 검색 단계 시퀀스

그림 57은 서비스 공급자 기반 실내 구동체웹 시스템의 검색 단계 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 서비스 등록자, 센서웹 공급자, 응용서버로 구성한다.

처음에 클라이언트는 서비스 등록자에게 응용서버의 서비스 정보를 검색하고 사용자가 원하는 응용서버를 접속하며 지도에 따라서 응용서버에서 구동체 오브젝트 정보를 지도상에 출력한다. 사용자가 지정하는 구동체 오브젝트를 구동체 정보에 속하는 구동체웹 공급자에게 세부 정보와 구동체 동작상태 정보를 요청한다.

1.1.4 구동체 제어 단계

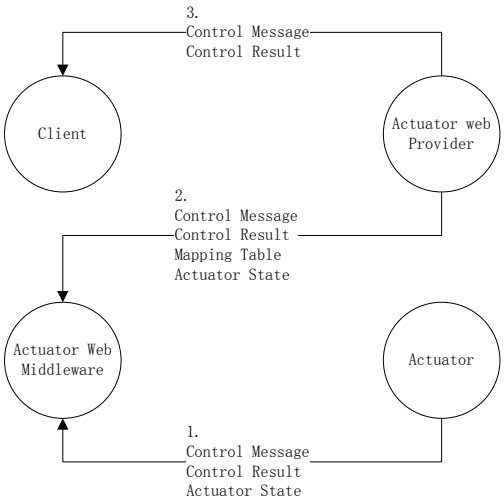

그림 58 서비스 공급자 기반 실내 구동체웹 시스템 제어 단계

그림 58은 서비스 공급자 기반 실내 구동체웹 시스템의 제어 단계이다. 구동체 제어 단계는 응용서버 서비스 공급자가 환경 상태에 따라서 작동 구동체 제어 명령을 내리고 실행한다.

Control Message는 구동체를 제어하기 위한 명령 메시지를 의미한다. Control Result는 구동체를 제어하고 응답 메시지를 의미한다. Mapping Table는 구동체의 네트워크의 연결 환경정보(구동체 및 미들웨어의 IP, ID영사 관계)를 의미한다. Actuator State는 구동체의 동작 상태정보(가전의 파워상태, 에어컨의 조정온도 등)를 의미한다.

구동체가 연결, 동작상태 및 제어 결과를 미들웨어에게 전송. 미들웨어가 구동체에게 제어 명령을 전송한다. 미들웨어가 구동체웹 공급자에게 라우팅 테이블을 전송하고 제어 명령의 제어 결과를 전송한다. 구동체웹 공급자가 미들웨어에게 제어 명령을 전송한다. 클라이언트가 구동체웹 공급자에게 제어 명령을 전송한다.

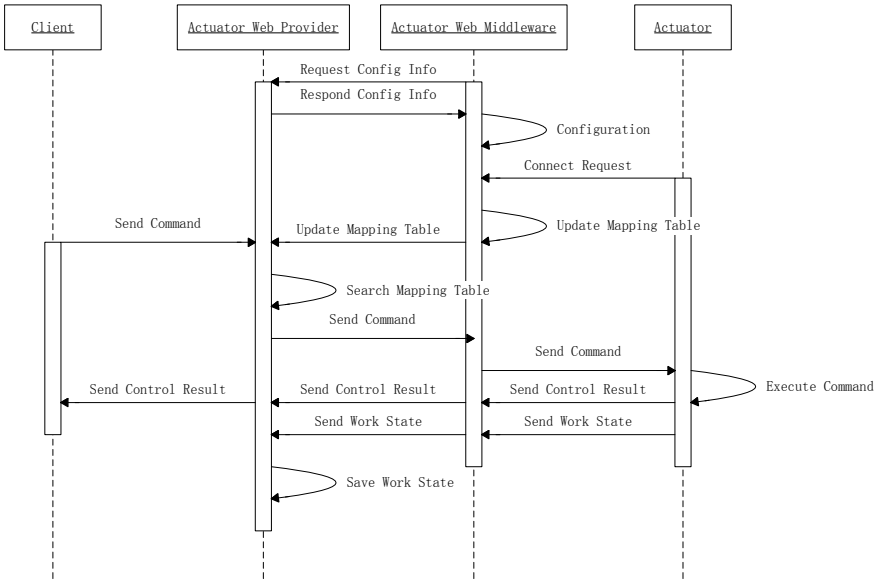

그림 59 서비스 공급자 기반 실내 구동체웹 시스템 제어 단계 시퀀스

그림 59는 서비스 공급자 기반 실내 구동체웹 시스템의 구동체 제어 단계 시퀀스다이어그램이다. 위의 과정은 클라이언트, 구동체웹 공급자, 구동체 미들웨어와 구동체를 구성된다. 위의 과정은 시작할 때 구동체 네트워크 미들웨어가 구동체웹 공급자에게 자기 Configuration 관련정보를 요청 받아서 배치한다. 다음에 구동체가 미들웨어에게 연결 요청 메시지를 전송한다. 미들웨어가 연결 메시지를 수신하여 분석하고 자기가 갖고 있는 매핑 테이블을 수정한다. 그리고 구동체웹 공급자에게 자기 갱신되는 매핑 테이블 서비스 공급자에게 전송하고 서비스 공급자가 새 매핑 테이블에 따라서 서비스 공급자 매핑 테이블에 대해 관리한다. 이제 클라이언트가 제어 메시지를 생성하여 구동체웹 공급자에게 보내고 서비스 공급자가 매핑 테이블에 따라서 목적 구동체와 연결하는 미들웨어에게 제어 메시지를 전달한다. 최종 미들웨어가 제어 메시지를 목적 구동체에게 보낸다. 구동체가 이 제어 메시지를 수행하면 상태 정보를 반드시 변경하기 때문에 제어 응답 메시지와 구동체 동작상태 정보를 미들웨어를 통해 서비스 공급자에게 전송한다. 서비스 공급자가 제어 응답 메시지를 클라이언트에게 알려주며 구동체 동작상태 정보를 DB에 저장한다.

1.2 응용서버 기반 실내 구동체웹

1.2.1 응용서버 기반 실내 구동체웹 시스템의 인터페이스 프로토콜

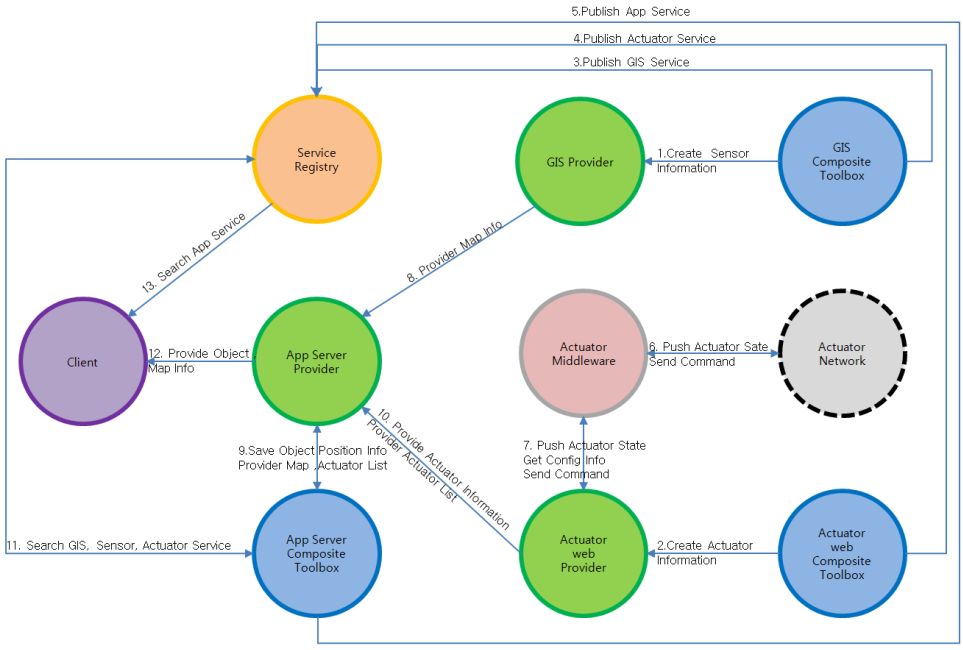

그림 60 응용서버 기반의 구동체웹 구조

표 14 응용서버 기반 구동체웹 시스템의 API

Module Name	API Name	Description		Applied At
		Type	Operation	
Service Registry	Service Publish Interface Protocol	Service Information	Write	3, 4, 5
	Service Search Interface Protocol	Service Information	Search, Read	11, 13
GIS Service Provider	GIS Provider Interface Protocol	Map Information	Read	8
	GIS Content Interface Protocol	Map Information	Read, Write	1
Actuator web Service Provider	Actuator web Service Provider Interface Protocol	Actuator List	Read, Search	10
		Actuator Information	Read	10
		Control	Write	10
	Actuator web Service Content Interface Protocol	Actuator List	Read, Search	2
		Actuator Information	Write	2
App Server	App Service Provider Interface Protocol	Map Information	Read	12
		Actuator Information	Read	12
		Object Information	Read	12
	App Service Content Interface Protocol	Map Information	Read	9
		Actuator Information	Read	9
		Actuator List	Read	9
		Object Information	Read, Write	9

그림 60은 서비스 공급자 기반 실내 구동체웹 시스템 내부 모듈간의 API 구

성도이다. 구동체 웹 시스템은 주로 구동체 노드 정보관리, 공급 그리고 구동체를 제어 및 동작상태 감시하는 역할을 담당한다.

표 14는 구동체웹 시스템의 내부에 존재하는 API들이다. 위의 표와 그림 60을 같이 참조 하면 각 모듈간 어떤 API를 통해 연동하는 것 쉽게 이해할 수 이다. Service Publish API는 서비스 등록하는 역할을 수행하고, Service Search API는 서비스 정보 검색하는 역할을 수행하고, GIS Service Provider API는 지도정보를 공급하는 역할을 수행하고, GIS Content API는 지도정보 관리하는 역할을 수행, Actuator Web Provider API는 구동체 정보 및 구동체 제어하는 역할을 수행하고, Actuator Web Content API는 구동체 정보 관리하는 역할을 수행하고, App Service Provider API는 지도정보, 구동체 정보 및 오브젝트 정보 공급하는 역할을 수행하고, App Service Content API는 지도 서비스 공급, 구동체 리스트 및 오브젝트 정보 관리하는 역할을 수행한다.

1.2.2 환경 설정 단계

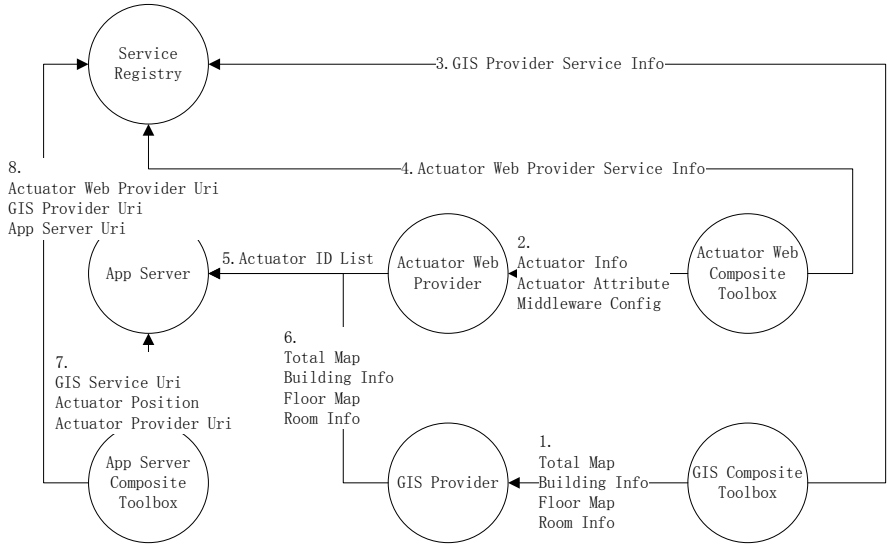

그림 61 응용서버 기반 실내 구동체웹 시스템 환경 설정 단계

그림 61은 응용서버 기반 실내 구동체웹 시스템의 환경 설정 단계이다. 실내

구동체웹 시스템의 환경 설정 단계는 서비스의 콘텐츠를 제대로 클라이언트에게 공급을 수행하기 위해 생성한다.

Total Map은 실외 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Building Info는 빌딩의 이름, 위치 정보, 층 이름을 포함한다. Floor Map은 실내 지도의 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Room Info는 방의 이름, 방 위치를 포함한다. Actuator Info는 구동체 노드의 이름, 구동체 ID를 포함한다. Actuator Attribute는 구동체의 속성 타입 정보를 의미한다. Middleware Config는 구동체 미들웨어의 ID, IP 주소와 접속권한 정보를 포함한다. Service Info는 각각 서비스 공급자의 서비스 이름, 검색 키워드, 접속 주소, 서비스 유형을 포함한다. Actuator List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 구동체 ID들을 의미한다. GIS Service Uri는 GIS 공급자의 접속주소를 의미한다. Actuator Position는 구동체의 지도상의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다.

GIS 서비스 저작도구는 지도 관련 데이터를 외부에서 GIS Content API를 통해 GIS 공급자의 DB에 저장한다. 구동체웹 저작도구는 각각 구동체 노드의 정보를 Actuator Web Content API를 통해 생성하여 DB에 저장하여 서비스 정보를 서비스 등록자에게 등록한다. 구동체웹 저작도구가 구동체웹 공급자 미들웨어의 배치 정보를 생성한다. GIS 서비스 저작도구가 GIS 공급 서비스의 정보를 서비스 등록자에게 등록한다. 구동체웹 공급자가 제공하는 센서 리스트 정보를 응용서버를 통해 응용서버 저작도구에게 제공한다. GIS 공급자가 제공하는 지도 데이터를 응용서버를 통해 응용서버 저작도구에게 제공한다. 응용서버 저작도구가 각각 센서 노드와 지도의 위치정보를 생성하여 저장한다. 응용서버 저작도구가 응용서버의 서비스 정보를 등록한다.

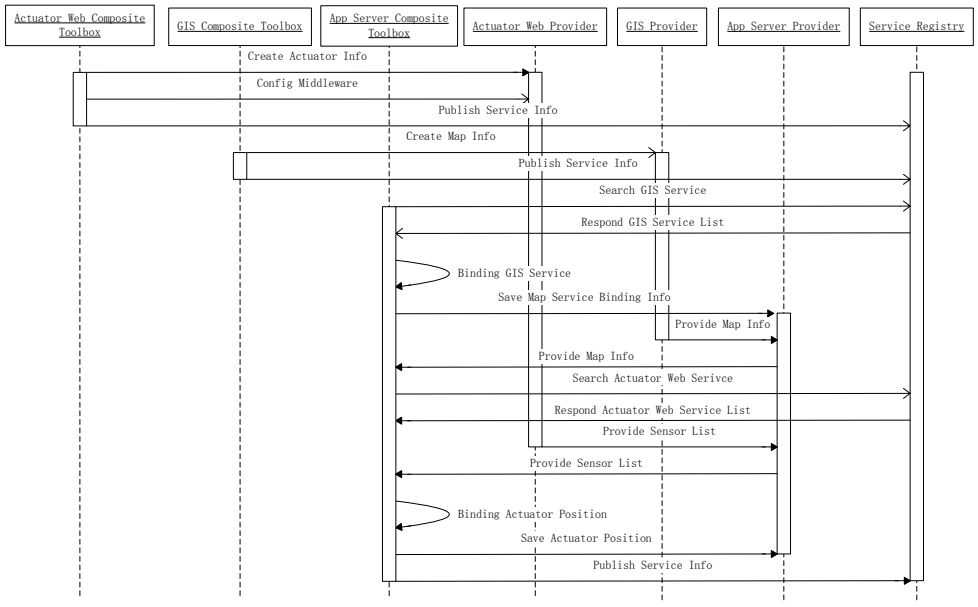

그림 62 응용서버 기반 실내 구동체웹 시스템 환경 설정 단계 시퀀스

그림 62는 서비스 공급자 기반 실내 구동체웹 시스템의 환경 설정 단계 시퀀스 다이어그램이다. 위의 과정은 구동체웹 저작도구, GIS 서비스 저작도구, 응용서버 저작도구, 구동체웹 공급자, GIS 공급자, 응용서버 서비스 공급자와 서비스 등록자를 구성한다.

이 단계를 실행할 때 구동체웹 저작도구는 구동체 오브젝트의 정보를 생성하여 구동체웹 공급자에 저장하며 구동체 미들웨어의 배치 정보를 생성하여 구동체웹 공급자에게 등록한다. 다음에 구동체웹 공급자 서비스 정보를 서비스 등록자에게 등록한다. 그리고 GIS 서비스 저작도구가 지도정보를 생성하여 GIS 공급자에 전송하고 GIS 공급자의 서비스 정보를 서비스 등록자에게 등록한다. 응용서버 저작도구는 서비스 등록자에서 GIS 서비스 정보를 검색하여 관리자가 자기 원하는 지도 엔진 서비스를 골라서 바인딩하고 응용서버에 저장한다. GIS 공급자가 응용서버의 통합 서비스를 통해 지도 정보를 응용서버 저작도구에게 제공한다. 서비스 등록자가 제공하는 구동체웹 공급자 서비스 정보를 뽑아서 응용서버를 통해서 이 구동체웹 공급자가 제공하는 구동체 노드 정보를 보여준다. 관리자가 선택된 구동체 노드를 지도에서 하나씩 위치를 바인딩하고 저장한다. 마지막 App

Serve의 서비스 정보를 서비스 등록자에 저장한다.

1.2.3 검색 단계

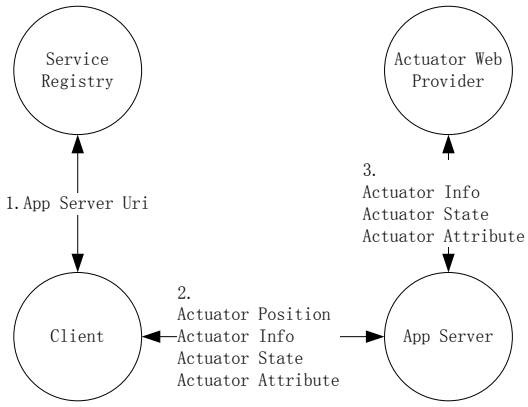

그림 63 응용서버 기반 구동체웹 시스템 검색 단계

그림 63은 응용서버 기반 실내 구동체웹 시스템 검색 단계이다. 구동체웹 시스템의 검색 단계는 클라이언트가 서비스 등록자에 쉽게 구동체 노드를 찾을 수 있도록 한다.

Actuator Info는 구동체 노드의 이름, 구동체 ID를 포함한다. Actuator State는 구동체의 동작상태 정보를 의미한다. Actuator Attribute는 구동체의 작업 속성을 의미한다. Actuator Position는 구동체의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

클라이언트는 사용자가 입력하는 서비스검색 키워드를 통해 서비스 등록자에게 서비스 리스트를 요청한다. 클라이언트가 응용서버를 접속하여 구동체 노드의 위치 정보를 요청한다. 클라이언트가 구동체의 ID를 이용하여 응용서버를 통해 구동체웹 공급자에 구동체 정보(구동체 세부정보, 동작상태, 속성)를 요청한다.

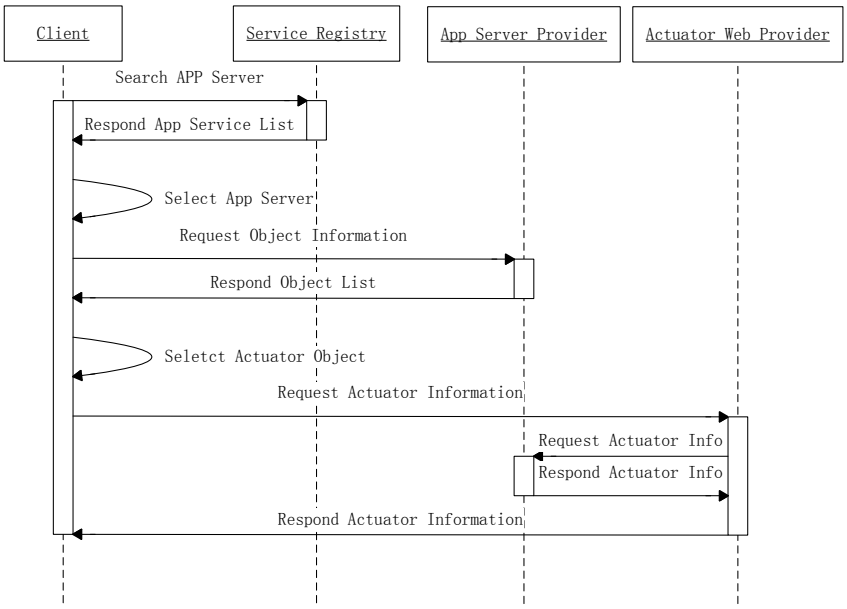

그림 64 응용서버 기반 구동체웹 시스템 검색 단계 시퀀스

그림 64는 응용서버 기반 실내 구동체웹 시스템의 검색 단계 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 서비스 등록자, 센서웹 공급자, 응용서버를 구성한다.

처음에 클라이언트는 서비스 등록자에게 응용서버의 서비스 정보를 검색하고 사용자가 원하는 응용서버를 접속하며 지도에 따라서 응용서버에서 구동체 오브젝트 정보를 지도상에 출력한다. 사용자가 지정하는 구동체 오브젝트의 구동체 정보를 응용서버를 통해 해당 구동체웹 공급자에 구동체 세부 정보, 동작 상태 및 속성 정보를 요청한다.

1.2.4 구동체 제어 단계

그림 65는 응용서버기반 실내 구동체웹 시스템의 제어 단계이다. 구동체 제어단계는 응용서버 서비스 공급자가 환경 상태에 따라서 작동 구동체 제어 명령을 내리고 실행한다. Control Message는 구동체를 제어하기 위한 명령 메시지를 의미한다. Control Result는 구동체를 제어하고 응답 메시지를 의미한다. Mapping

Table는 구동체의 네트워크의 연결 환경정보(구동체 및 미들웨어의 IP, ID영사 관계)를 의미한다. Actuator State는 구동체의 동작 상태정보(가전의 파워상태, 에어컨의 조정온도 등)를 의미한다. 구동체가 연결, 동작상태 및 제어 결과를 미들웨어에게 전송. 미들웨어가 구동체에게 제어 명령을 전송한다. 미들웨어가 구동체웹 공급자에게 라우팅 테이블을 전송하고 제어 명령의 제어 결과를 전송한다. 구동체웹 공급자가 미들웨어에게 제어 명령을 전송한다. 클라이언트가 응용서버를 통해 구동체웹 공급자에게 제어 명령을 전송한다.

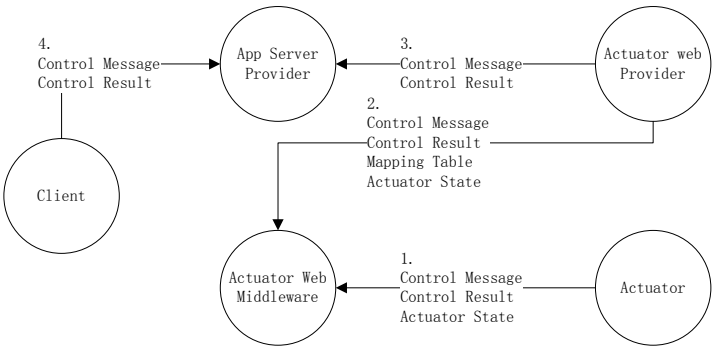

그림 65 응용서버 기반 구동체웹 시스템 제어 단계

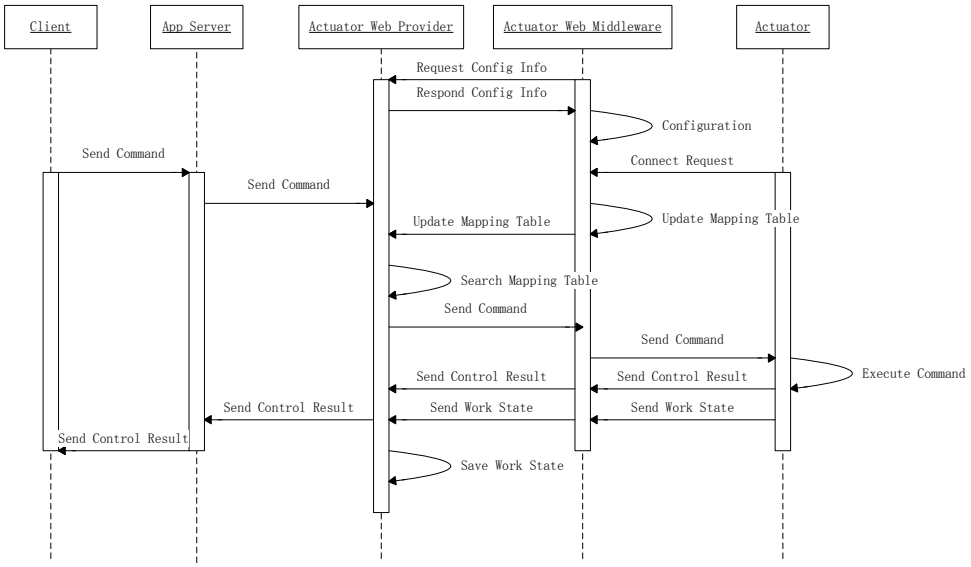

그림 66 응용서버 기반 구동체웹 시스템 제어 단계 시퀀스

그림 66는 응용서버 기반 구동체웹 시스템의 제어 단계 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 구동체웹 공급자, 구동체 미들웨어와 구동체를 구성된다.

위의 과정은 시작할 때 구동체 네트워크 미들웨어가 구동체웹 공급자에게 자기 환경 배치 관련정보를 요청 받아서 배치한다. 다음에 구동체가 미들웨어에게 연결 요청 메시지를 전송한다. 미들웨어가 연결 메시지를 수신하여 분석하고 자기 갖고 있는 매핑 테이블을 수정한다. 그리고 구동체웹 공급자에게 자기가 갱신되는 매핑 테이블을 서비스를 공급자에게 전송하고 서비스 공급자가 새 매핑 테이블에 따라서 공급자 매핑 테이블에 대해 관리한다. 이제 클라이언트가 제어 메시지를 생성하여 응용서버를 통해서 서비스 공급자에게 보내고 서비스 공급자가 매핑 테이블에 따라서 목적 구동체와 연결하는 미들웨어에게 제어 메시지를 전달한다. 최종 미들웨어가 제어 메시지를 목적 구동체에게 보낸다. 구동체가 이 제어 메시지를 수행하면 상태 정보를 반드시 변경하기 때문에 제어 응답 메시지와 구동체 동작상태 정보를 미들웨어를 통해 서비스 공급자에게 전송한다. 서비스 공급자가 제어 응답 메시지를 응용서버를 통해 클라이언트에게 전달하며, 마지막 구동체웹 공급자가 수집한 구동체 동작상태 정보를 DB에서 저장한다.

1.3 실내 구동체웹 시스템 세부설계

실내 구동체웹 시스템중의 구동체웹 공급자, 구동체웹 저작도구, 구동체 미들웨어, 구동체 에뮬레이터, GIS 공급자, GIS 저작도구, 응용서버, 응용서버 저작도구, 서비스 등록자 및 클라이언트에 대해 내부 모듈을 설계한다. 본 논문의 III장 1절에서 제시하는 두 방안은 응용서버의 구조만 차이가 있지만 다른 모듈의 구조 같다. GIS 공급자와 GIS 저작도구의 세부구조가 VI장의 1.1절에서 기술하여 있다. 서비스 등록자와 응용서버 저작도구의 세부구조가 V장의 2.3절에서 기술하여 있다.

1.3.1 구동체웹 공급자

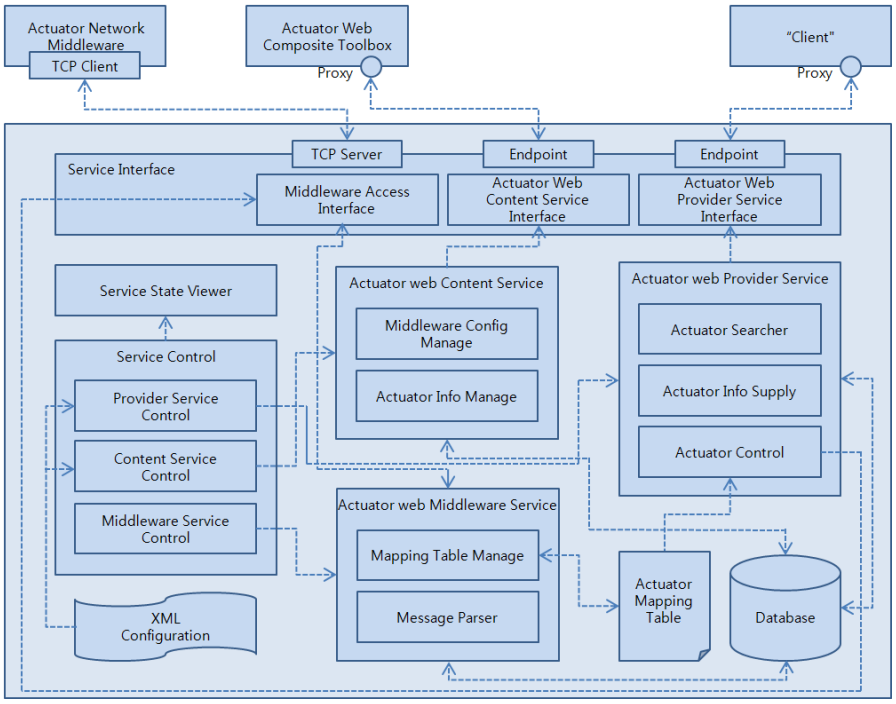

그림 67 구동체웹 공급자 세부구조

그림 67은 구동체웹 공급자의 세부구조이다. 구동체웹 공급자는 Actuator Web Content Service, Actuator Web Provider Service, Actuator Middleware Service로 구성된다. Actuator Web Content Service는 구동체 미들웨어 Config Manage와 Actuator Info Manage로 구성된다. Middleware Config Manage는 미들웨어의 ID, IP 주소와 서비스 접속 권한정보를 생성 및 관리하는 역할을 담당한다. Actuator Info Manage는 구동체 노드의 이름, ID, 센싱 타입 정보를 생성 및 관리하는 역할을 수행한다. Actuator Web Provider Service는 구동체의 정보 공급, 정보 검색, 제어 서비스를 제공한다. Actuator Web Provider Service는 센서 Searcher, Actuator Info Supply와 구동체 제어로 구성된다. Actuator Searcher는 검색 키워드에 의해 구동체 ID 리스트를 제공한다. Actuator Info Supply는 구동체 ID에 의해 해당 구동체 노드의 이름, 구동체 동작상태 정보를 제공한다. 구동체 제어는 구동체 원격 제어 인터페이스를 제공한다. 구동체 미들웨어 서비스는 미들웨어가 보내는 메시지 파싱과 매핑 테이블을 관리하는 역할을 수행한다. Mapping Table Management는 메모리에서 저장되는 구동체 ID와 구동체 미들웨어 주소간의 영상관계정보를 관리하는 역할을 담당한다. Message Parser는 구동

체 미들웨어에서 수신하는 메시지에 대해 미리 정의된 포맷형식에 의해 풀어준다. Service Control은 Provider Service Control, Middleware Service Control과 Content Service Control로 구성되고 Provider Service, Middleware Service 및 Content Service의 열림, 닫음을 제어한다. Service State Viewer는 서비스의 동작 상태(서비스 인터페이스를 꺼짐/켜짐 정보)를 표시한다. XML Configuration은 WCF Service의 동작 환경을 배치한다. Database는 서비스가 제공하는 구동체 노드정보(구동체 이름, ID, 모델, 속성)를 저장하는 것이다. Actuator Mapping Table은 구동체의 연결 구동체 ID와 구동체 미들웨어 주소간의 영상관계 정보를 저장한다.

1.3.2 구동체웹 저작도구

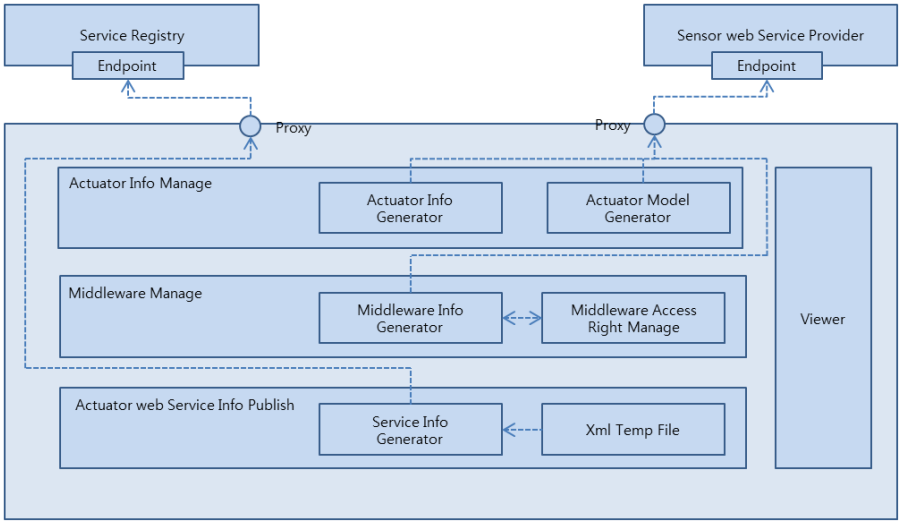

그림 68 구동체웹 저작도구 세부구조

그림 68은 구동체웹 저작도구 세부구조이다. 구동체웹 저작도구는 구동체 정보를 생성, 미들웨어 접속관리 및 서비스 정보등록 역할을 담당한다. Actuator Info Generator는 구동체 정보(이름, ID, 타입, 속성 등)를 생성하여 Provider에 저장한다. Actuator Model Generator는 구동체의 모델 정보(구동체 속성정보)를 생성 및 관리하는 역할을 담당한다. Middleware Info Generator는 Middleware

정보(미들웨어 ID, IP주소)를 등록하며, Middleware Access Right Manage는 미들웨어가 서비스 공급자와의 접속 권한을 관리해 준다. Actuator Web Service Info Publish는 Service Info Generator를 통해 서비스 정보(검색 키워드, 이름, 주소 등)를 생성하여 XML임시 파일에 저장하여 서비스 등록자에게 등록한다.

1.3.3 구동체 미들웨어

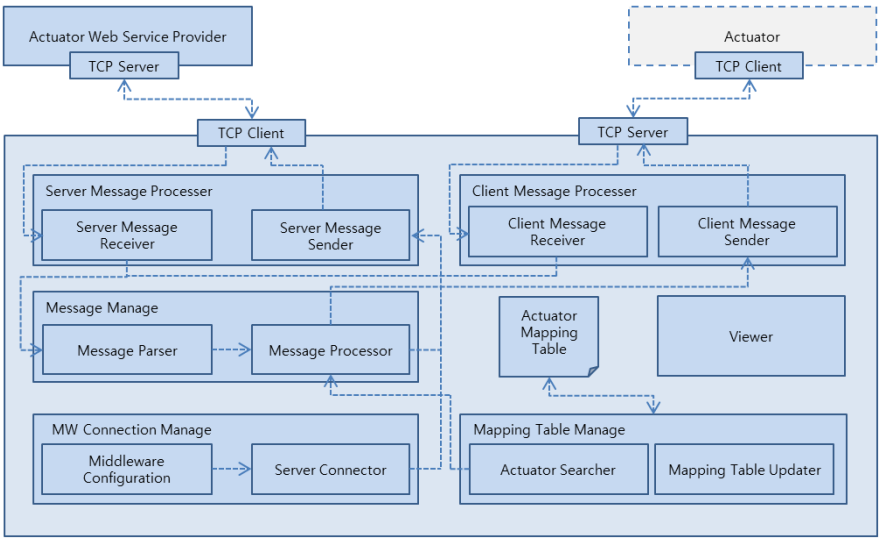

그림 69 구동체 미들웨어 세부구조

그림 69는 구동체 미들웨어의 세부구조이다. 구동체 미들웨어는 구동체웹 공급자와 구동체와의 중계 모듈이다. 즉 구동체웹 공급자가 보내는 메시지를 제대로 구동체에게 전달하고 구동체가 보내는 메시지를 서비스 공급자에게 전송한다. 이 모듈의 양방향 통신기능을 Socket TCP를 사용하여 구현한다. Server Message Receiver는 구동체웹 공급자가 송신하는 제어 메시지를 수신하는 역할을 담당한다. Sever Message Sender는 구동체 미들웨어부터 구동체웹 공급자에게 메시지를 전달하는 역할을 수행한다. Message Parser는 구동체 서비스 공급자와 구동체에서 수신하는 메시지를 메시지 포맷으로 풀어준다. Message Processor는 파싱된 메시지 타입에 따라서 처리방식(연결요청, 제어요청, 매핑테이블 갱신 요청, 제어 응답, 연결 응답)을 결정한다. Middleware Configuration

는 서비스 공급자가 제공하는 ID, IP정보에 따라 환경을 구성한다. Server Connector는 미들웨어의 접속권한을 검증하여 서비스 공급자와 연결 여부를 결정한다. Client Message Receiver는 구동체에서 전달하는 메시지를 수신한다. Client Message Sender는 미들웨어부터 구동체까지 메시지를 전송하는 역할을 담당한다. Actuator Mapping Table는 구동체의 IP주소와 구동체의 ID간의 관계정보를 저장하는 곳이다. Viewer는 사용자 접속을 위해 가시화를 제공한다. Actuator Searcher는 구동체의 ID에 의해 구동체의 IP주소를 검색하는 역할을 담당한다. Mapping Table Updater는 구동체 미들웨어 메모리에서 저장하는 구동체 영사관계 정보를 실시간 갱신하는 역할을 담당한다.

Server Message Receiver는 서비스 공급자가 보내는 제어 메시지를 수신하여 Message Parser에게 전달하고 파싱 처리된 메시지를 Message Processor에게 맡겨 준다. Message Processor가 메시지 내용에 의해 Actuator Searcher를 통해 구동체의 연결상태 정보를 요청하여 클라이언트 Message Sender를 통해 목적 Actuator에게 전송한다. Server Message Sender는 미들웨어가 생성하는 연결 요청 메시지, 클라이언트 Message Receiver가 수신하는 제어 응답 메시지, Actuator 동작 상태 메시지를 서비스 공급자에게 보낸다.

1.3.4 구동체 에뮬레이터

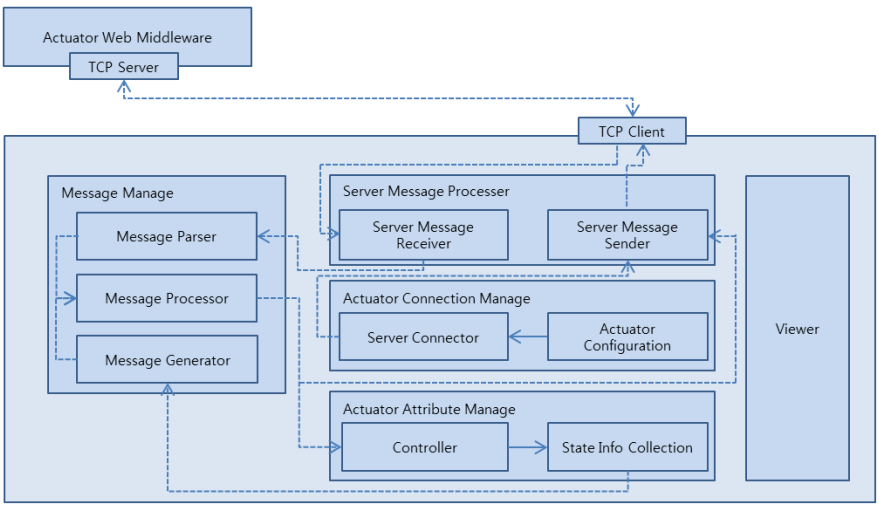

그림 70 구동체 에뮬레이터 세부구조

그림 70은 구동체 에뮬레이터의 세부구조이다. 본 연구는 실제 네트워크와 연결하고 원격제어 기능이 있는 가전제품 대신 구동체 에뮬레이터를 사용한다. Message Parser는 구동체 미들웨어부터 수신하는 메시지를 특정한 포맷으로 해석한다. Massage Processor는 해석된 메시지의 타입유형에 따라 처리 방식((연결 요청, 제어요청, 매핑 테이블 갱신 요청, 제어 응답, 연결 응답)을 결정한다. Message Generator가 구동체의 메모리에서 보낼 메시지(제어 응답 메시지, 동작 상태 메시지, 연결 요청 메시지)를 생성한다. Server Message Receiver는 구동체 미들웨어가 송신하는 메시지(제어 메시지, 연결 응답 메시지)를 수신역할을 담당한다. Sever Message Sender는 구동체부터 구동체 미들웨어에게 메시지(구동체 동작상태 메시지, 연결 요청 메시지)를 전달하는 역할을 수행한다. Server Connector는 구동체의 접속권한을 검증하여 구동체 미들웨어와 연결 여부를 결정한다. Actuator Configuration는 외부에서 네트워크 환경을 구성한다. Controller는 구동체 에뮬레이터의 동작상태를 변경하는 역할을 수행한다. State Info Collection는 구동체의 동작 상태 정보를 수집하는 역할을 담당한다.

구동체 에뮬레이터는 주로 미들웨어와 연동하며 수신하는 메시지를 Message Parser를 통해 파싱하고 Message Processor 가 구동한다. Server Connector는 Actuator Configuration의 연결환경 정보에 따라서 미들웨어와 연결시킨다. Controller는 Message Processor가 생성하는 Command에 따라서 구동체의 동작 상태를 변경한다. 그리고 변경되는 동작 상태정보를 State Info Collection을 통해 미들웨어에게 전송한다.

1.3.5 구동체웹 응용서버

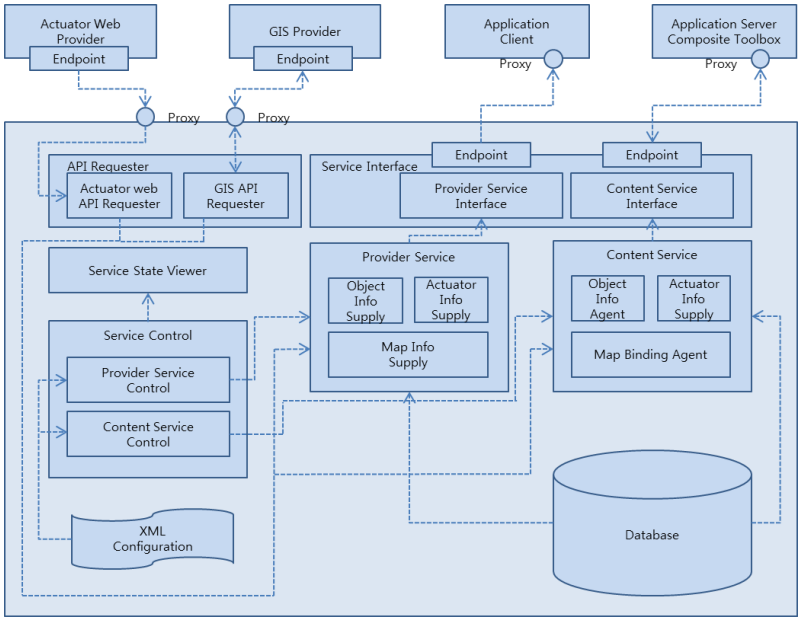

그림 71 응용서버 기반 구동체웹 응용서버 세부구조

서비스 공급자 기반 구동체웹의 응용서버의 세부구조가 III장의 2.4.4절의 서비스 공급자 기반 센서웹의 응용서버의 세부구조와 같다. 그림 71은 응용 서버 기반 구동체웹 응용서버 세부구조이다. 응용서버는 Provider Service, Content Service로 구성된다. Provider Service는 Object Info Supply, Actuator Info Supply 및 Map Info Supply로 구성한다. Object Info Supply는 클라이언트에게 구동체 오브젝트 정보(위치 정보, 타입, 공급 서비스 주소)를 제공한다. Actuator Info Supply는 구동체의 세부 정보(구동체 이름, 구동체ID, 구동체 동작상태, 구동체 동작속성)를 제공한다. Map Info Supply는 지도 정보(실외 지도, 실내 지도, 빌딩, 층, 방 정보)를 제공한다. Content Service는 Object Info Agent, Sensor Info Supply 및 Map Info Supply로 구성한다. Object Info Agent는 응용서버 저작도구로서 구동체 오브젝트 위치 정보관리 및 지도 서비스 바인딩 관리 서비스를 제공한다. Actuator Info Supply는 센서의 세부정보를 제공한다. Map Info Supply는 지도 정보를 제공한다. XML Configuration은 WCF Service의 동작 환경을 제공한다. Database는 Map Service 바인딩 정보와 오브젝트 정보를 저장한다.

1.3.6 구동체웹 클라이언트

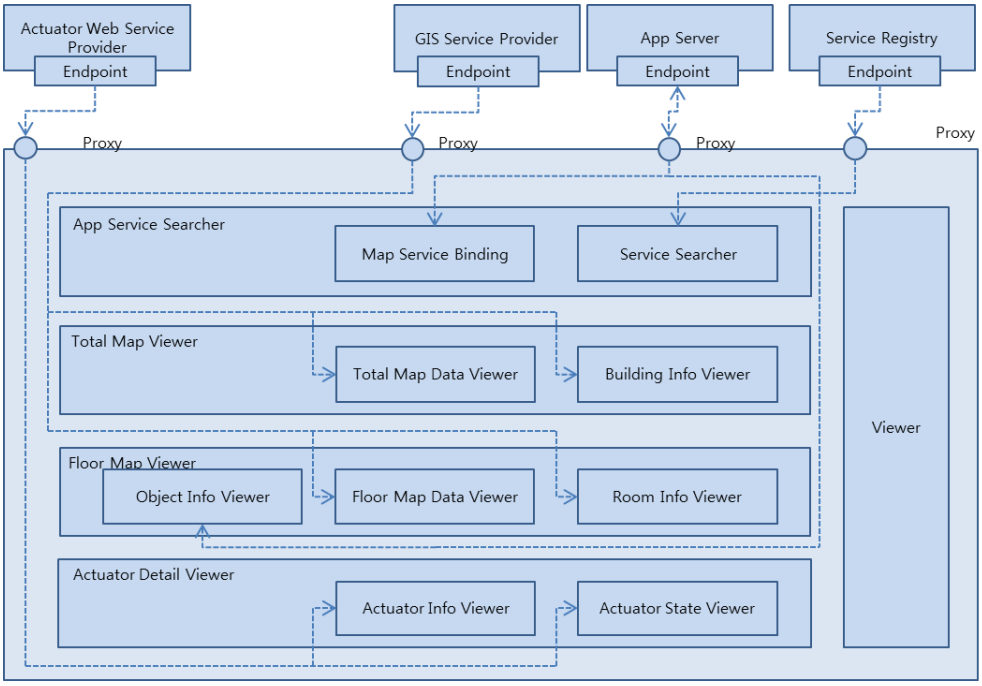

그림 72 서비스 공급자 기반 구동체웹 클라이언트 세부구조

구동체웹의 시스템 구조방안에 따라서 구동체웹 클라이언트의 내부 구조가 조급 차이가 있다. 그림 72는 서비스 공급자 기반 구동체웹 클라이언트의 세부구조이다. 클라이언트가 단순한 가시화를 제공한다. 처음에 App Service Searcher의 Service Searcher가 서비스 등록자에게 구동체웹 응용서버를 검색하여 지정한 응용 서비스를 접속하고 Map Service Binding이 응용서버에서 Map 서비스 정보를 요청하여 바인딩 한다. 다음에 Total Map Viewer는 GIS 공급자에서 전체지도 데이터 및 빌딩 정보를 요청한다. 관리자가 전체지도 뷰어에서 특정 빌딩의 층을 선택하여 Floor Map Viewer가 층 지도 데이터, 방 정보 및 오브젝트를 도시한다. 사용자가 구동체 오브젝트를 선택하면 Actuator Info Viewer가 구동체 이름, 아이디 등 속성을 출력하며 Actuator State Viewer가 실시간 구동체의 동작상태정보를 표시한다.

그림 73은 응용서버 기반 구동체웹 클라이언트의 세부구조이다. 클라이언트의 내부구조가 그림 34와 같지만 센서 정보, 지도 정보의 요청대상만 다르다. 응용서버 기반의 센서웹 클라이언트가 응용서버를 통해 지도 정보와 센서 세부정보를 공급자에 요청한다.

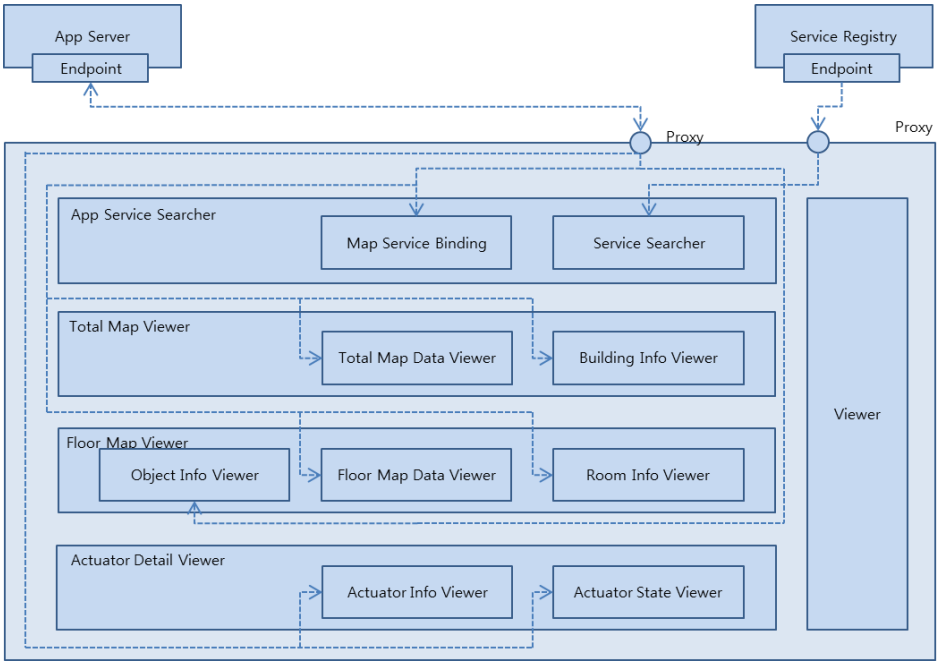

그림 73 응용서버 기반 구동체웹 클라이언트 세부구조

1.4 구동체웹 메시지 포맷

본 논문에서는 Socket TCP 방식을 사용하여 구동체웹 공급자에서 구동체까지의 통신역할이 수행되어야 하므로 주고 받는 메시지의 포맷을 정의해야 한다.

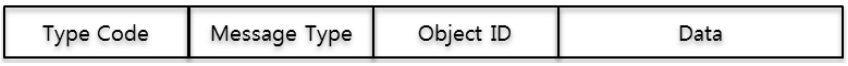

그림 74 메시지 포멧

그림 74는 구동체웹 시스템의 메시지 포맷이다. 구동체웹 시스템의 메시지는 Type Code, Message Type, 오브젝트 ID와 Data 네 부분으로 구성된다. 구동체웹 시스템에 네 가지 타입의 메시지를 사용하면 요구를 만족할 수 있다. 이는 연결 제어 메시지, 매핑 정보 교환 메시지, 제어 메시지하고 상태 정보 메시지로 나눈다.

표 15 메시지 포맷 타입

Number	Type Code	Type	Object ID	Data		Explanation
①	00	Request	Actuator ID/Middleware ID	Key & "connect"	----------	Server Connect Request Message
②	00	Respond	Actuator ID/Middleware ID	Key & "connect"	True/False	Server Connect Respond Message
③	00	Request	Actuator ID/Middleware ID	Key & "disconnect"	----------	Server Disconnect Request Message
④	00	Respond	Actuator ID/Middleware ID	Key & "disconnect"	True/False	Server Disconnect Respond Message
⑤	01	----------	Middleware ID	"Actuator ID" List	----------	Service Provider Mapping Table Message
⑥	10	Request	Actuator ID	Key & Command	----------	Actuator Control Request Message
⑦	10	Respond	Actuator ID	Key	True/False	Actuator Control Respond Message
⑧	11	----------	Actuator ID	State List	----------	Actuator Work State Notice Message

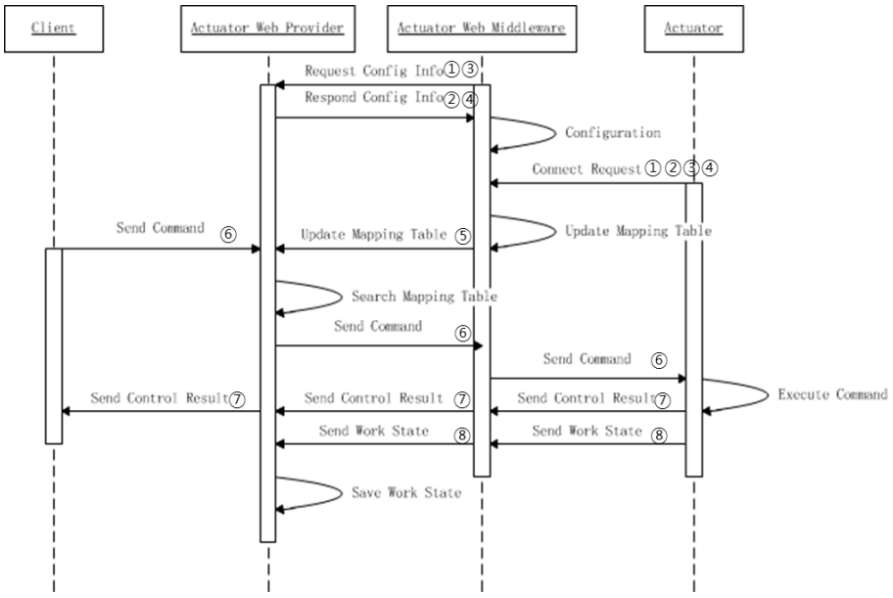

그림 75 매시지 타입 역할

표 15는 구동체웹 시스템 메시지 포맷의 세부 설명이다. Type Code가 "00"이면 연결 메시지, "01"이면 매핑 정보 교환 메시지, "10"이면 제어 메시지, "11"이면 상태 정보 메시지이다. 제어 메시지와 연결 메시지는 단순 Push 방식으로 데이터를 송신하는 것이 아니라 HTTP처럼 응답 매커니즘이 있어야 한다. 그래서 Type이라는 필터를 통해서 Request하고 Respond를 나눌 수 있다. 오브젝트 ID

는 클라이언트입장에서 클라이언트의 유일한 식별자이다. 여기서는 미들웨어를 같은 경우는 Middleware ID이고 구동체는 Actuator ID이다. 이 방식을 이용하여 클라이언트가 메시지를 파싱할 때 메시지를 어느 장비로 보냈는지 쉽게 알 수 있다. Data는 메시지의 실제 내용을 저장한다.

그림 75는 표 15에서 정의되는 메시지 타입이 실제 구동체웹 시스템에서 담당하는 역할을 보여준다. 1, 2, 3, 4는 연결상태를 관리기능을 수행하고, 5는 매핑 테이블을 관리역할을 수행하고, 6, 7은 구동체를 제어관리역할을 담당하고, 8은 구동체의 동작상태 정보를 전송하는 역할을 수행한다.

메시지 예제:

- ✧ “00&DS00001&12315342&Request”: DS00001이라는 디바이스가 서버를 연결 요청한다.
- ✧ “00&DS00001&12315342&Repond&True”: DS00001이라는 디바이스 보내는 “12315342”라는 연결 요청메시지에 대해 연결 성공한다.
- ✧ “01&MW00012&DS00001;DS00002;DS000003”: MW00012라는 미들웨어가 지금 DS00001, DS00002, DS00003과 연결하고 있다.
- ✧ “10&DS00001&12315342&Request&POW_STA:OFF”: DS000001라는 디바이스가 전원 꺼짐을 요청한다.
- ✧ “10&DS00001&12315342&Request&False”: DS00001이라는 디바이스 보내는 “12315342”라는 제어 요청메시지에 대해 제어를 실패한다.
- ✧ “11&DS00001&POW_STA:ON;LIG_LEV:Strong”: DS00001라는 디바이스의 파워 상태 “ON”이고 조명 등급 “Strong”이다.

1.5 구동체 연결 라우팅

본 논문이 제시하는 구동체웹 시스템의 원격 제어 메시지 전송기능은

Socket TCP를 기반으로 구현된다. 그러나 서비스 공급자와 미들웨어, 미들웨어와 구동체 사이에 일 대 다 통신방식을 사용하기 때문에 서비스 공급자 및 미들웨어 입장에서 메시지의 전달 메커니즘을 필요로 한다. 이는 네트워크기술의 라우팅 개념과 아주 유사하다.

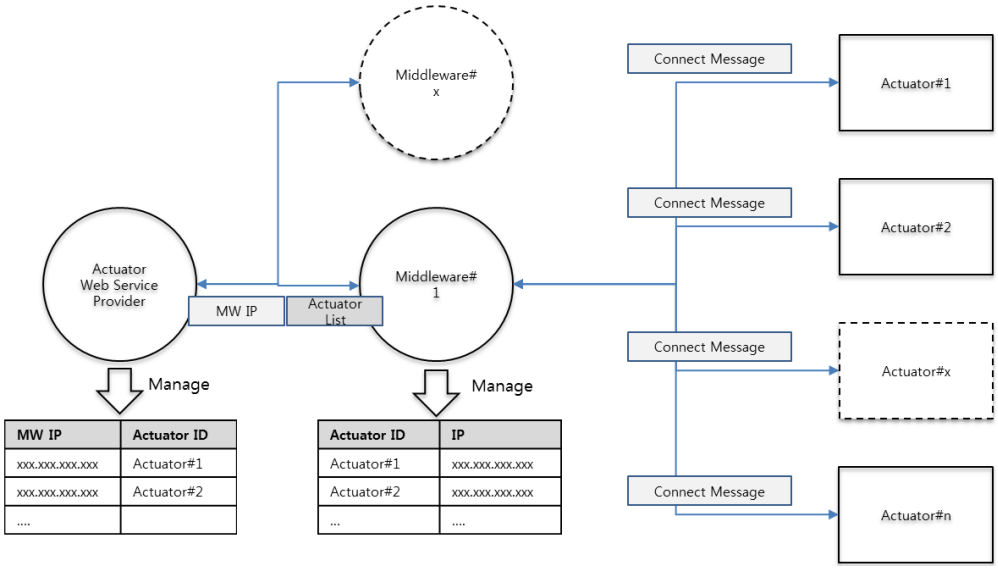

그림 76 매핑 테이블 관리

그림 76은 매핑 테이블의 관리 생성 및 관리 과정이다. 처음에 미들웨어가 각 구동체에서 연결 요청 메시지를 수신하여 자기 메모리에서 매핑 테이블을 생성하고 이 정보를 구동체웹 공급자에게 전송한다. 이 서비스 공급자가 매핑 테이블 정보를 기존의 매핑 테이블과 비교하여 관리한다.

2 실내 구동체웹 시스템 구현

구동체웹 시스템은 구동체 관련하는 정보를 공급, 관리 및 제어하는 역할을 수행한다. 구동체웹 시스템부분은 구동체웹 공급자, 구동체웹 저작도구, 구동체 미들웨어, 구동체 에뮬레이터, GIS 공급자, GIS 저작도구, 응용서버, 응용서버 저작도구, 서비스 등록자와 클라이언트로 구성된다. GIS 공급자와 GIS 저작도구의 구현결과는 VI장의 2절에서 기술하여 있다. 응용서버, 응용서버 저작도구, 서비스 등록자와 클라이언트의 구현결과는 V장의 3절에서 기술하여 있다.

2.1 구동체웹 공급자

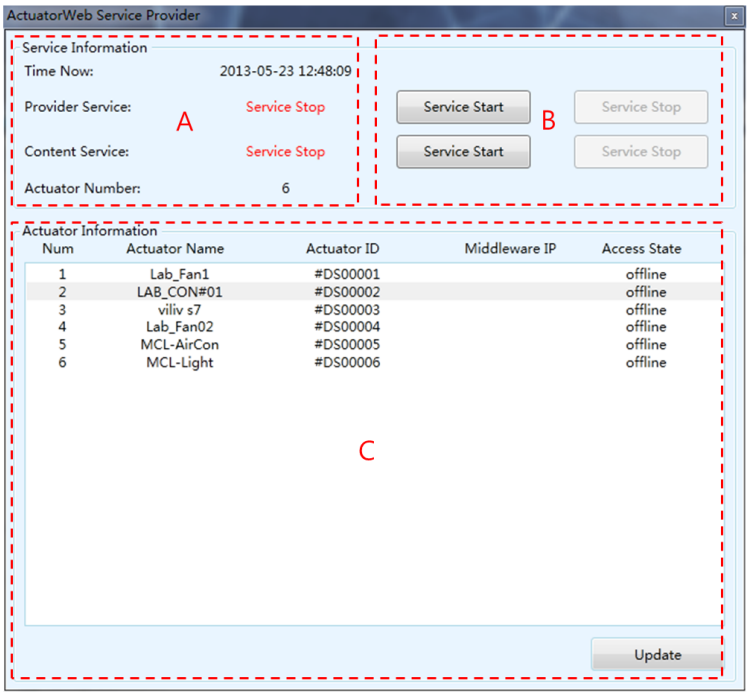

그림 77 구동체웹 공급자 실행화면

그림 77은 구동체웹 공급자 실행화면이다. A구역은 서비스의 구동 상태를 표시한다. Time Now는 구동체웹 공급자를 동작하는 현재시간을 표시한다. Provider Service는 구동체웹 공급자의 공급 서비스 동작상태(꺼짐/켜짐)를 표시한다. Content Service는 구동체웹 공급자의 콘텐츠 서비스 동작상태(꺼짐/켜짐)

를 표시한다. Actuator Number는 현재 DB에 등록되는 구동체 노드정보 개수를 표시한다. B구역은 구동체웹 공급자가 제공하는 Content Service 및 Provider Service의 켜짐, 꺼짐을 관리하는 역할을 수행한다. C구역은 DB에 등록된 구동체 정보(구동체 이름, ID, 연결하는 미들웨어 IP 주소, 연결상태)를 표시한다.

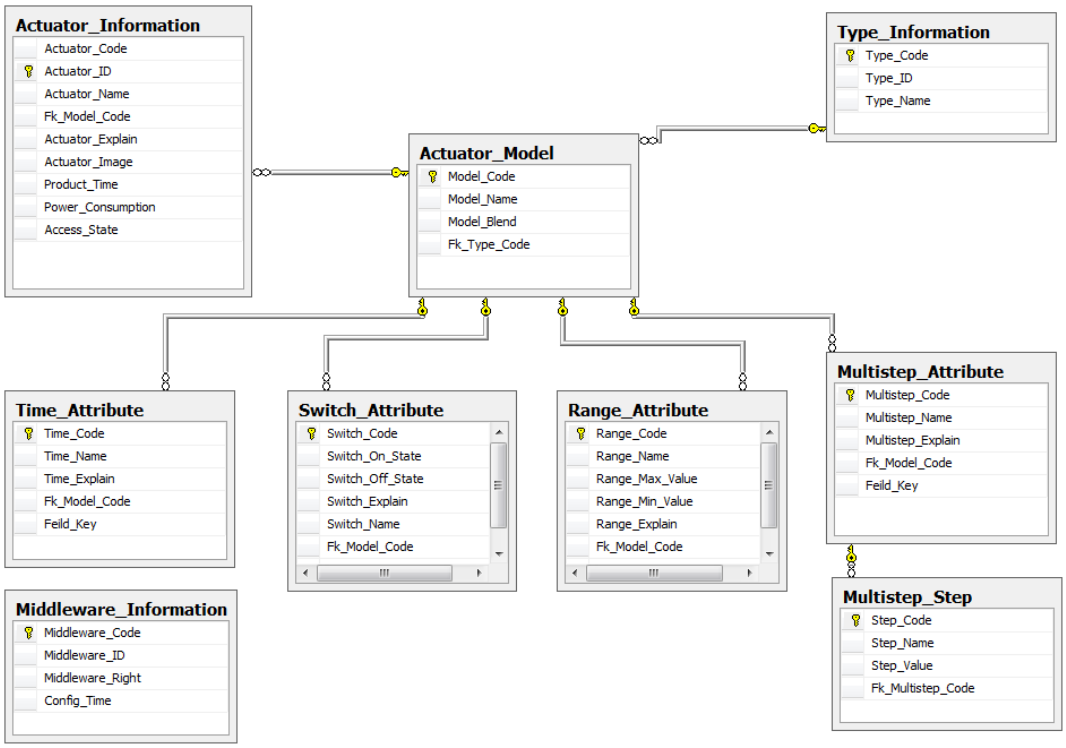

그림 78 구동체웹 공급자 데이터베이스 관계도

그림 78은 구동체웹 공급자의 데이터베이스이다. Actuator_Information은 구동체의 정보를 저장하는 테이블이다. Actuator_Model은 구동체의 모델정보를 저장하는 테이블이다. 보통 가전 제품들은 4가지의 속성으로 나눌 수 있다. 즉 시간 속성, 스위치 속성, 범위 속성, 다중 단계 속성이 있다. 예를 들어 에어컨의 예약 시간은 시간 속성이며, 전원의 꺼짐, 켜짐은 스위치 속성이며, 온도의 조정은 범위 속성이며, 선풍기의 풍속 조정은 다단계 속성이다. Time_Attribute, Switch_Attribute, Range_Attribute, Multistep_Attribute는 위와 같은 속성 정보를 저장하는 테이블이다. Type_Information은 가전의 기능 분류 정보를 저장하는 테이블이다. Middleware_Information은 미들웨어의 정보 및 접속 권한을 저장하

는 테이블이다.

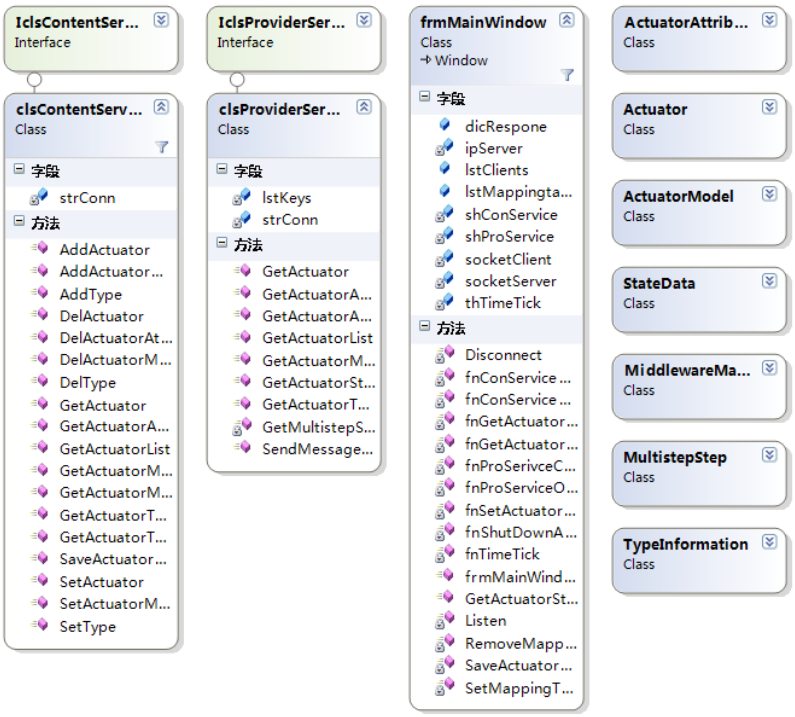

그림 79 구동체웹 공급자 클래스 다이어그램

그림 79는 구동체웹 공급자의 클래스 다이어그램이다. IclsContentService 및 IclsProviderService가 서비스를 외부에 노출하기 위한 인터페이스를 나타낸 것이다. clsContentService는 DB의 구동체 정보를 생성 및 관리하는 기능을 제공한다. clsProviderService는 DB에서 저장하는 구동체 정보를 읽어서 외부에 공급한다. frmMainWindow는 서비스 제어 및 서비스 상태의 뷰어기능을 제공한다. ActuatorAttibute는 Actuator속성 정보를, Actuator는 구동체 정보를, ActuatorModel은 구동체의 모델정보를, StateData는 구동체의 동작상태를 저장하는 클래스이다. MiddwareMappingItem은 매핑 정보를, MultistepStep은 다단계 속성의 단계정보를, TypeInformation은 구동체의 기능 분류정보를 저장하는 클래스이다.

2.2 구동체웹 저작도구

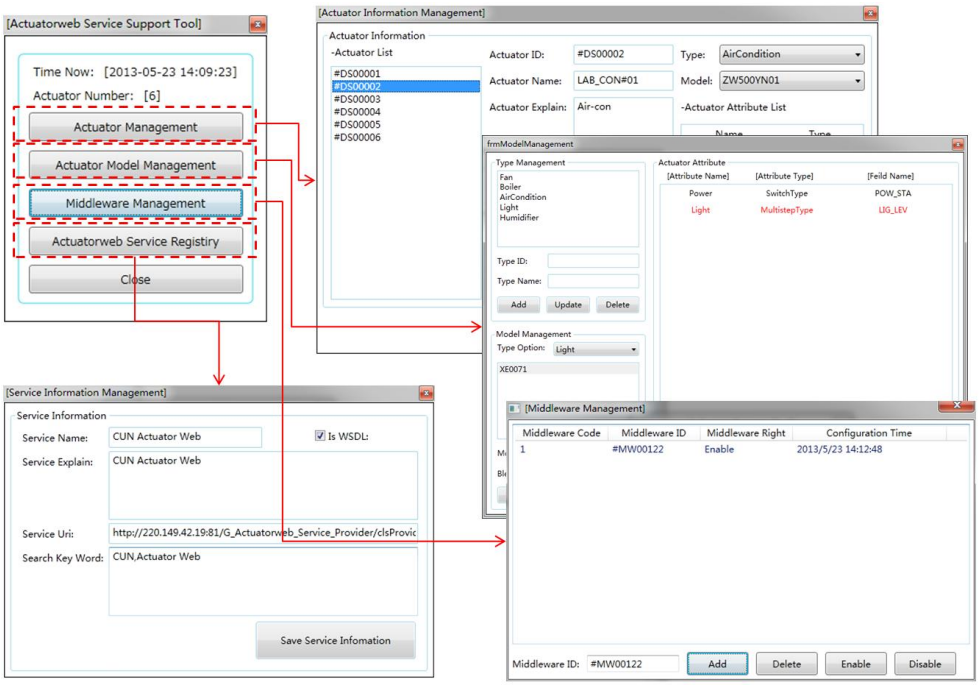

그림 80 구동체웹 저작도구 실행화면

그림 80은 구동체웹 저작도구 실행화면이다. 구동체웹 저작도구는 구동체 정보 생성 및 관리, 구동체 모델 생성 및 관리, 미들웨어 정보 생성 및 관리, 서비스 정보 생성 및 관리역할을 제공한다.

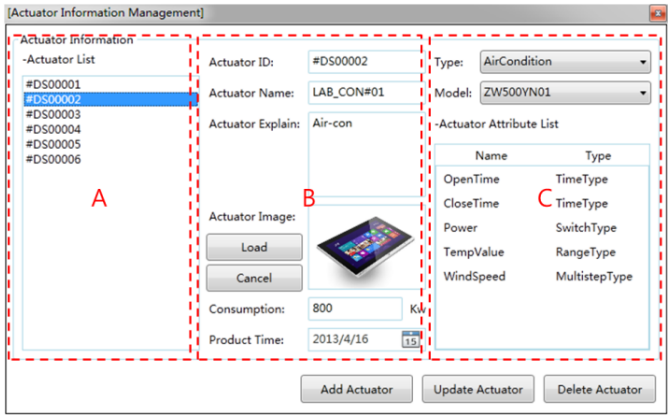

그림 81 구동체 정보 관리 실행화면

그림 81은 구동체 정보 관리 실행화면이다. A구역은 이미 등록된 구동체 아이디를 나열한다. B구역은 구동체의 정보(구동체 아이디, 이름, 설명, 이미지, 부하 및 생산 날짜)를 생성 및 수정하는 역할을 담당한다. 구동체 ID는 구동체의 유일 식별자이다. Actuator Name는 구동체의 이름을 지정한다. Actuator Explain는 구동체에 대한 기능 설명이다. Actuator Image는 구동체의 사진이다. Consumption는 해당 구동체의 에너지부하이다. Product Time는 구동체의 생산 날짜이다. C구역은 구동체의 모델정보를 세팅하는 역할을 수행한다. Type는 구동체의 기능 타입(선풍기, 에어컨, 보일러 및 조명기구 등)을 지정한다. 모델은 해당 구동체의 모델명을 선택한다. Actuator Attribute는 구동체의 기능 속성이다.

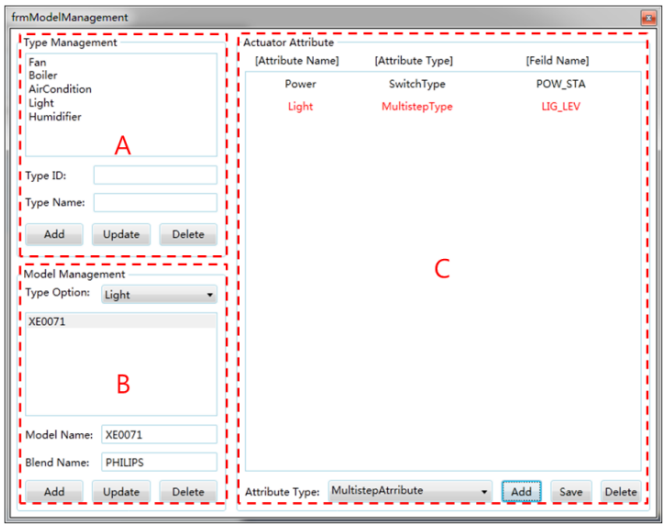

그림 82 구동체 모델정보 관리 실행화면

그림 82는 구동체 모델정보 관리 실행화면이다. A구역은 구동체의 타입을 생성 및 관리하는 기능을 담당한다. Type ID는 구동체 기능 타입의 유일 식별자이다. Type Name는 구동체의 기능 타입 이름이다. B구역은 지정하는 타입의 모델정보를 생성 및 관리하는 역할을 수행한다. Type Option는 DB에 기존되는 기능 타입을 선택한다. Model Name는 구동체의 모델 이름이다. Blend Name는 해당 구동체 모델명의 제조회사이다. C구역은 선택된 모델에 대한 구동체 속성을 생성

및 관리하는 역할을 수행한다. Attribute Name는 구동체 동작상태 속성 이름이다. Attribute Type은 구동체 동작상태 속성 유형(스위치 속성, 범위 속성, 시간 속성 및 다중 단계 속성)이다.

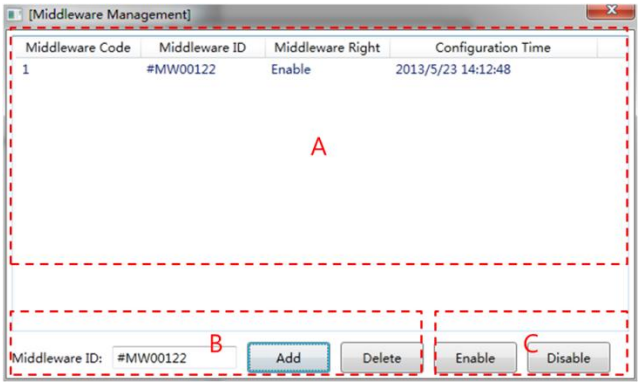

그림 83 미들웨어 관리 실행화면

그림 83은 미들웨어 관리 실행화면이다. A구역은 미들웨어 리스트를 표시하는 역할을 담당한다. 여기서 서비스 공급자의 DB에서 등록되는 미들웨어 정보(미들웨어 코드, ID, 접속권한, 등록하는 시간)를 나열한다. B구역은 미들웨어 정보를 추가 및 삭제기능을 제공한다. C구역은 미들웨어의 서버 접속 권한을 관리하는 역할을 담당한다.

.

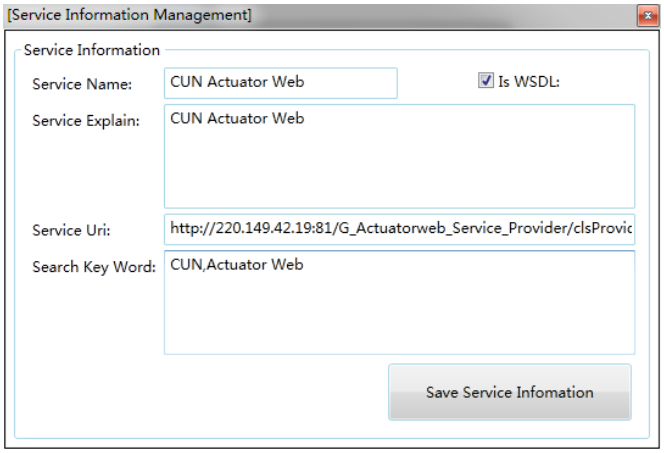

그림 84 구동체웹 서비스 정보 관리 실행화면

그림 84는 구동체웹 서비스 정보 관리 실행화면이다. 이는 서비스 정보를 생성하여 서비스 등록자에서 등록하는 역할을 수행한다. 서비스 정보는 서비스 이름, WSDL제공 여부, 서비스 설명, 서비스 접속 주소, 서비스 검색 키워드를 포함한다. Service Name는 서비스 이름을 입력하는 구역이다. Is WSDL는 서비스 공급자의 WSDL(Web Services Description Language)의 제공여부를 관리한다. Service Explain는 서비스에 대한 설명이다. Service Uri는 서비스 공급자의 접속 주소이다. Search Key Word는 서비스 공급자를 검색하는 키워드를 지정한다. Save Service Information는 서비스 공급자의 정보를 서비스 등록자에게 등록시키는 버튼이다.

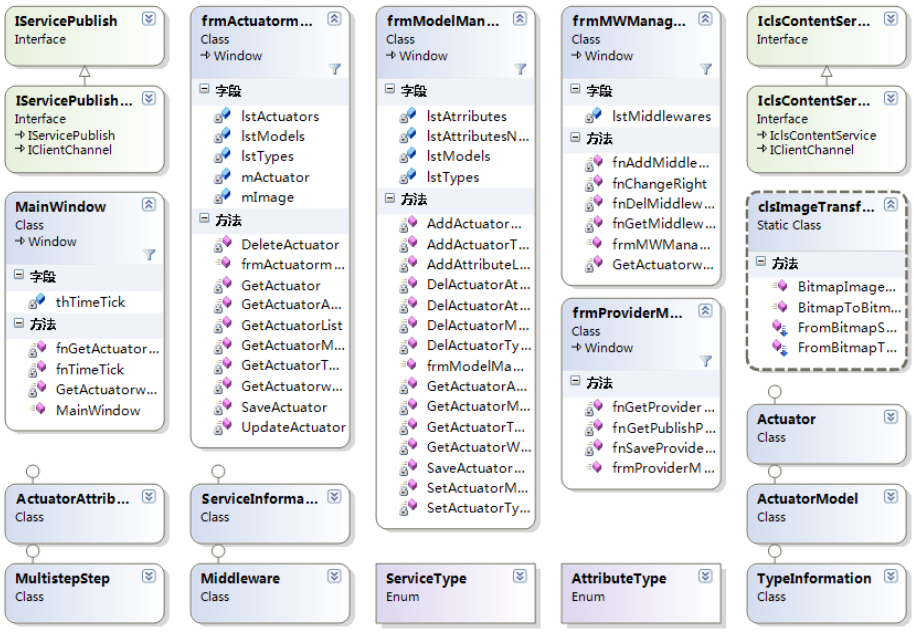

그림 85 구동체웹 저작도구 클래스 다이어그램

그림 85는 구동체웹 저작도구 클래스 다이어그램이다. IServicePublish 및 IcslContentService는 WCF Service를 참조하기 위한 인터페이스이다. MainWindow, frmActuatorManagement, frmModelManagement, frmMWManagement, frmProviderManagement는 실제 화면에 표시되는 WPF

form을 상속받는 클래스를 나타낸다. MainWindow는 다른 창을 호출하는 역할을 담당하여 frmActuatorManagement는 구동체 정보의 생성 및 관리기능을 제공한다. frmModelManagement는 구동체의 모델 정보의 생성 및 관리역할을 수행하고 frmMWManagement는 구동체 미들웨어의 정보 관리역할을 담당한다. 그리고 frmProviderManagement는 공급 서비스의 정보를 생성하여 등록하는 역할을 수행한다. clsImageTransform은 WPF에서 각 종 이미지 클래스간의 변환 메소드를 제공한다.

2.3 구동체 미들웨어

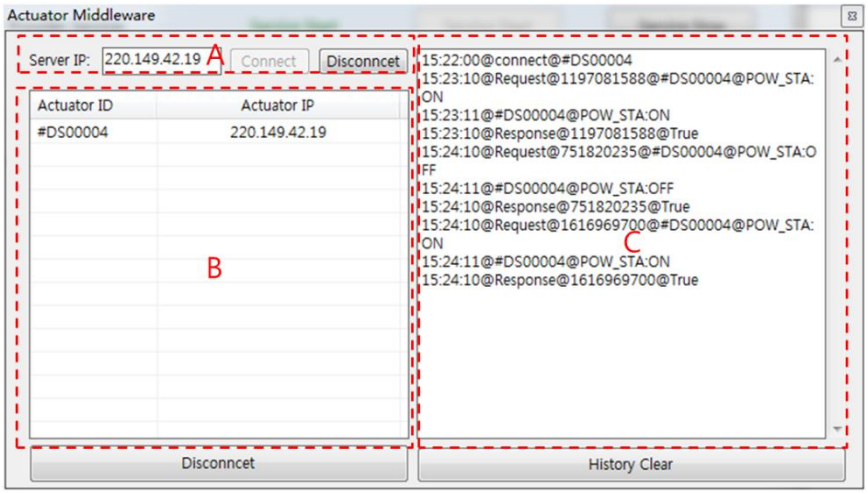

그림 86 구동체 미들웨어 실행화면

그림 86은 구동체 미들웨어 실행화면이다. A구역은 구동체웹 공급자의 IP주소를 세팅하는 역할을 담당한다. Server IP는 연결할 구동체웹 공급자의 IP주소를 지정한다. B구역은 미들웨어와 연결하는 구동체 연결상태 정보를 표시한다. Actuator ID는 구동체의 유일 식별자이다. Actuator IP는 현재 구동체의 IP 주소이다. C구역은 구동체웹 서비스와 구동체에서 수신되는 메시지를 표시한다.

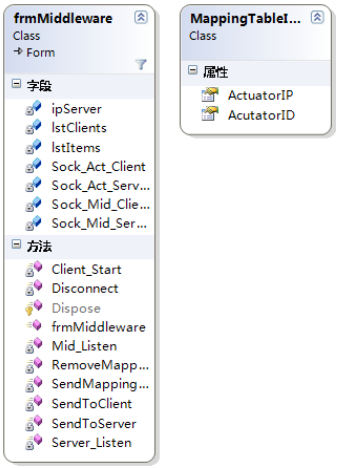

그림 87 구동체 미들웨어 클래스 다이어그램

그림 87은 구동체 미들웨어의 클래스 다이어그램이다. frmMiddleware는 실제 화면에 표시되는 Window form을 상속받는 클래스를 나타낸다. 이는 미들웨어의 연결상태, 동작상태 그리고 송수신 메시지의 가시화 기능을 담당한다. MapingTableItem은 연결하고 있는 구동체의 IP 정보를 저장하는 클래스이다.

2.4 구동체 에뮬레이터

이제 IPV6 시대가 점점 도래함에 따라 스마트 홈과 관련된 기술이 급속히 발전하고 있다. 미래는 모든 가전제품이 센서 칩을 탑재하고 네트워크를 연결하여 사용자가 원격에서 관리할 수 있다. 본 논문은 디바이스의 하드웨어구조를 논하는 대신 에뮬레이터를 구현하여 실제 상황을 모방한다.

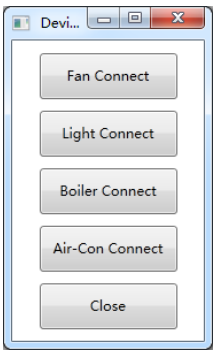

그림 88 구동체 오브젝트 생성 도구

그림 88은 구동체 오브젝트의 생성 도구이다. 이를 통해 선풍기 에뮬레이터, 조명 에뮬레이터, 보일러 에뮬레이터와 에어컨 에뮬레이터를 호출한다.

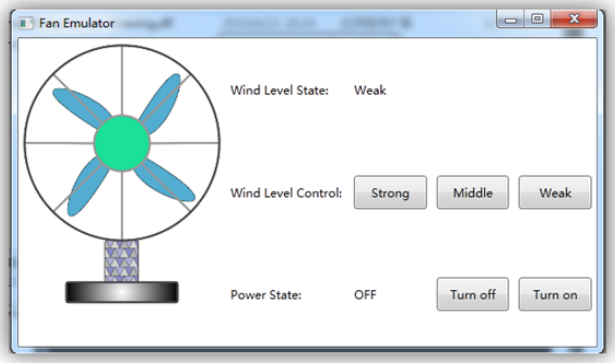

그림 89 선풍기 에뮬레이터

그림 89는 선풍기 에뮬레이터 화면이다. 풍량의 조정 및 전원을 제어하는 기능을 제공한다. Wind Level State는 선풍기의 현재 풍속 등급을 표시한다. Wind Level Control은 선풍기의 풍속 상태를 Strong, Middle 및 Weak 세 가지로 조정할 수 있다. Power State는 파워를 ON/OFF로 조정 가능하다.

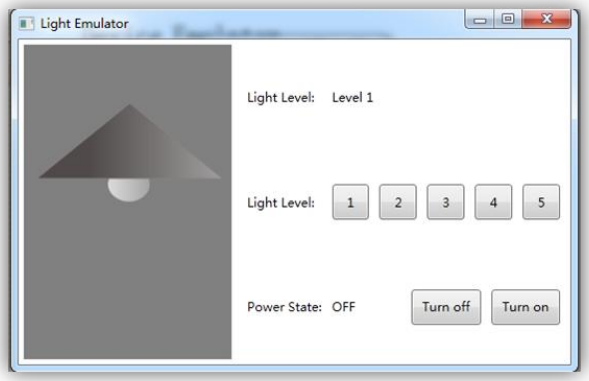

그림 90 조명 에뮬레이터

그림 90은 조명에뮬레이터 화면이다. 조도의 조정 및 전원을 제어하는 기능을 제공한다. Light Level은 조명 기구의 현재 조도 등급을 표시한다. Light Level Control은 조명 기구의 조도를 1부터 5까지 다섯 가지로 조정할 수 있다. Power State는 파워를 ON/OFF로 조정 가능하다.

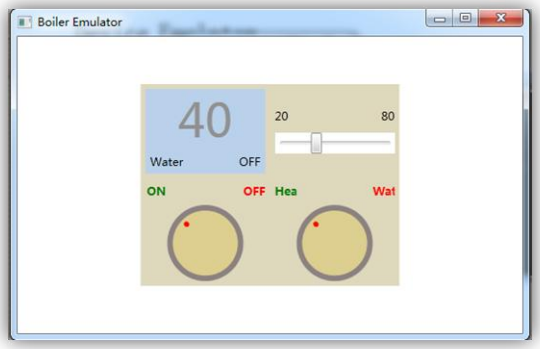

그림 91 보일러 에뮬레이터

그림 91은 보일러 에뮬레이터 화면이다. 온도 조정, 보일러 모드 조정 및 전원을 제어하는 기능을 제공한다. 온도는 20도부터 80도까지 조정하고, 동작 모드는 Heat와 Water 옵션을 선택하고, 파워는 ON/OFF 상태를 조정하는 기능을 제공한다.

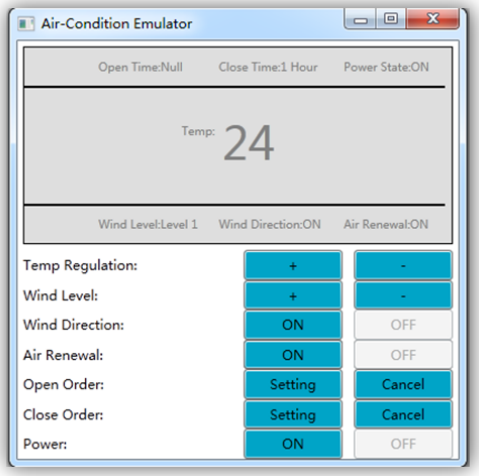

그림 92 에어컨 에뮬레이터

그림 92는 에어컨 에뮬레이터 화면이다. 온도조정, 풍량조정, 풍향조정, 환기조정, 예약배치 및 전원조정 기능을 제공한다. 그림의 위는 에어컨의 상태를 표시하는 구역이며 아래는 제어 구역이다. Temp Regulation은 온도 조정을, Wind Level은 풍속 강도 조정을 제공한다. Wind Direction은 작동 풍향모드, Air Renewal은 환기모드, Open Order 및 Close Order는 에어컨 예약 시간의 지정기능을 제공한다. Power는 에어컨의 파워상태 제어를 제공한다.

3 성능 분석 및 평가

3.1 실험환경

실험환경은 표 16과 같다. 실험은 구동체웹 시스템의 주요 기능에서 성능 분석을 한다. 모든 성능분석은 해당 주요모듈 별로 20회에 걸쳐 수행시간을 측정하고 분석한다.

표 16 실험환경

Division	Detail
Operating System	Microsoft Windows 7(X64)
Development Environment	.Net Framework 3.5, 4.0
Development Tool	Visual Studio .Net 2010
Programming Language	C#, XMAL, XML
DBMS	Microsoft SQL Server 2008 R2
Hardware	CPU: Intel® Core™ i3-2125 @ 3.30GHz RAM: 4GB Graphics: NVIDIA GeForce GT 440

구동체웹 시스템은 주로 구동체의 세부 정보 및 구동체의 제어 요청하여 응답하는 수행시간을 측정한다. 구동체웹 공급자의 데이터베이스에 저장된 구동체 세부 정보를 요청하여 응답을 수행하는 시간 및 서비스 공급자를 통해 특정 가상 구동체의 제어를 요청하여 응답하는 시간을 측정한다.

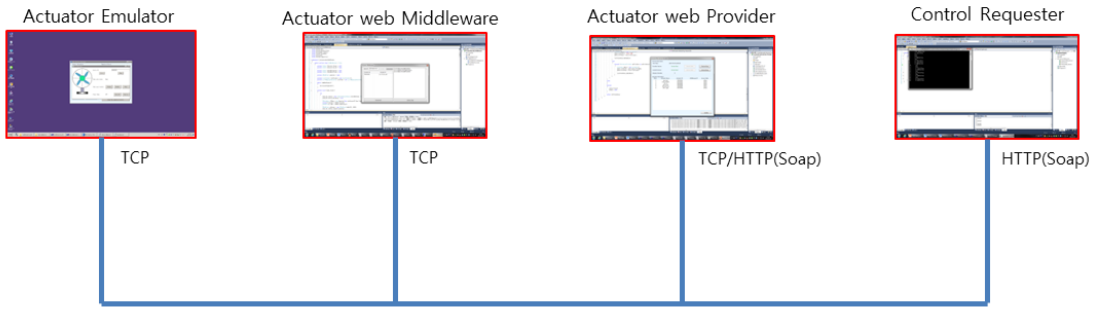

그림 93 구동체 웹의 실험 네트워크 구성

실험환경의 네트워크 구성은 그림 93과 같다. 실험을 위해 구동체웹 공급자를 구동하기 위한 데스크탑 서버 1대, 제어 요청자, 구동체 에뮬레이터, 구동체 미들웨어를 구동하기 위한 노트북 각 1대를 사용한다.

표 17 구동체 웹의 실험 네트워크 환경

Module Name	Address
구동체웹 공급자	http://220.149.42.19:81/G_Actuatorweb_Service_Provider/clsProviderService/
구동체 미들웨어	220.149.42.19
구동체 에뮬레이터	117.17.102.128
제어 요청자	117.17.102.197

3.2 구동체 원격 제어 성능평가

본 논문에서 구동체 웹 시스템은 실내 가전을 원격 제어하는 서비스 제공하는 역할을 담당한다. 그래서 클라이언트가 명령 생성부터 구동체 웹 서비스 공급자와 구동체 미들웨어를 통해 구동체에게 명령을 내리고 수행하여 응답 메시지를 수신되는 데까지의 소요시간을 측정하여 분석한다.

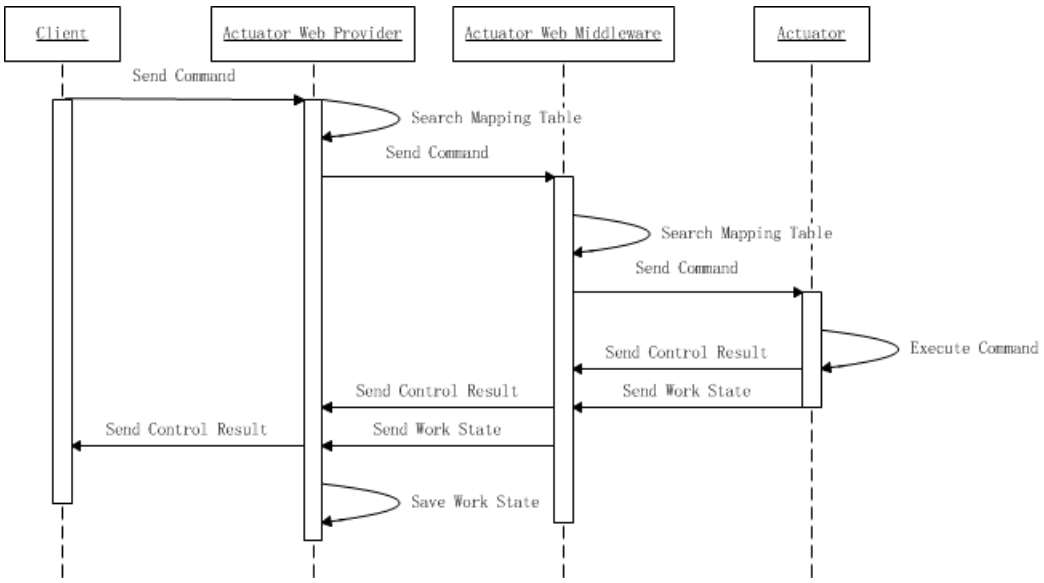

그림 94 구동체 제어 과정

본 실험은 그림 94과 같은 구동체 제어 과정을 수행하는 시간을 측정한다. 본 실험은 1개, 5개, 25개와 50개의 클라이언트 환경을 별도 만들어서 동시에 한 구동체에 대해 제어 명령을 내리고 응답 미시지를 수신한다.

표 18 실험 결과(구동체 제어)

	1 Client	5 Client	25 Client	50 Client
1	183.0156	1267.192	3943.354	7147.555
2	304.0234	1070.598	3630.404	6923.292
3	204.0156	996.5969	3857.866	9866.007
4	234.0234	1056.397	4479.644	10598.92
5	208.0234	969.8031	4216.742	10147.83
6	191.0156	110.3953	3663.966	10723.09
7	230.0156	966.8031	4225.102	10879.15
8	299.0234	1073.598	4007.681	11296.19
9	162.0078	1082	4244.626	11095.63
10	120.0078	1172.197	4167.218	7421.22
11	247.0234	1048.402	4416.801	5905.111
12	99.02344	1135.603	4370.038	7476.85
13	242.0234	985.2	4444.924	6968.816
14	264.0234	994.2984	4638.785	7054.586
15	355.0313	1062.506	4759.434	7797.719
16	234.0234	1069.506	4712.511	7504.249
17	219.0156	1175.522	4515.493	7459.004
18	208.0156	1154.114	4695.709	8023.143
19	222.0313	1046.108	4698.389	8658.947
20	222.0313	1152.916	4699.588	9034.884
Avg	222.3707	1029.488	4319.414	8599.11

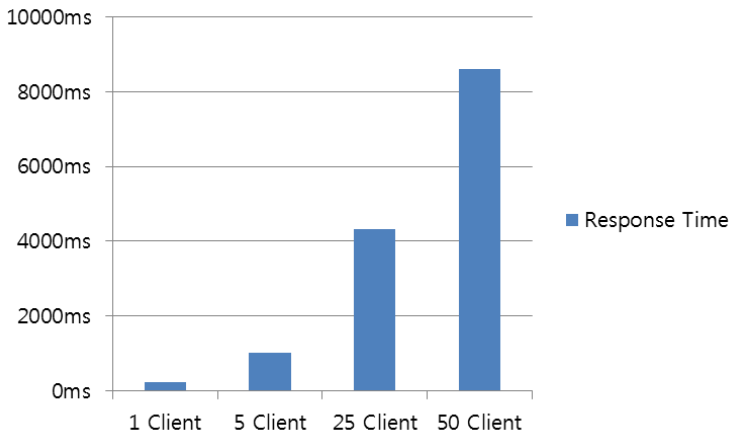

그림 95 실험 결과 비교(구동체 제어)

표 18은 클라이언트가 구동체를 제어 요청하는 실험결과이다. 가로는 동시에 구동체를 제어하는 클라이언트의 개수를 의미한다. 세로는 실험하는 횟수 및 평균을 의미한다.

그림 95는 표 18에 대한 실험 결과 비교 그래프이다. 왼쪽의 그래프는 실험 횟수에 따라서 1개, 5개, 25개, 50개의 클라이언트가 동시 제어 요청할 때 평균 소요하는 시간이다. 1개 클라이언트가 특정 구동체를 제어하는 평균시간이 222.3707ms이다. 5개 클라이언트가 특정 구동체를 제어하는 평균시간이 1029.488ms이다. 25개 클라이언트가 특정 구동체를 제어하는 평균시간이 4319.414ms이다. 50개 클라이언트가 특정 구동체를 제어하는 평균시간이 4319.414ms이다.

3.3 서비스 공급자 및 응용서버를 기반 구동체웹 비교 성능평가

본 절에서 서비스 공급자 기반 IOT 시스템의 클라이언트가 지정하는 구동체에게 제어 명령을 보내고 응답하는 시간과 응용서버 기반 IOT 시스템의 클라이언트가 지정하는 구동체에게 제어 명령을 보내고 응답하는 시간과 비교한다. 테스트를 위해 제주대학교 공대4호관의 D423호실에 설치하고 있는 ID는 “#DS00001”이라는 선풍기 에뮬레이터를 대상으로 성능 분석 진행한다. 그리고 여러 클라이언트를 준비하여 1대, 5대와 25대의 클라이언트가 별도 동시에 구동체를 원격 제어하는 시간을 측정하여 분석한다.

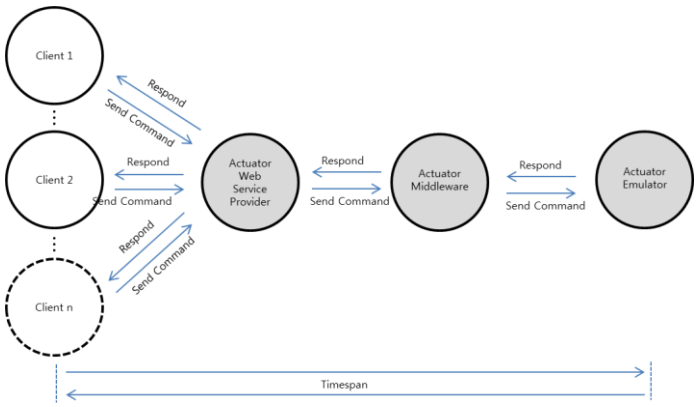

그림 96 서비스 공급자 기반 구동체 제어

그림 96은 서비스 공급자 기반의 IOT 시스템에서 클라이언트가 구동체를 제어하는 과정이다. 1대, 5대와 25대 클라이언트가 동시 접속경우를 각각 20회 실험을 진행한다.

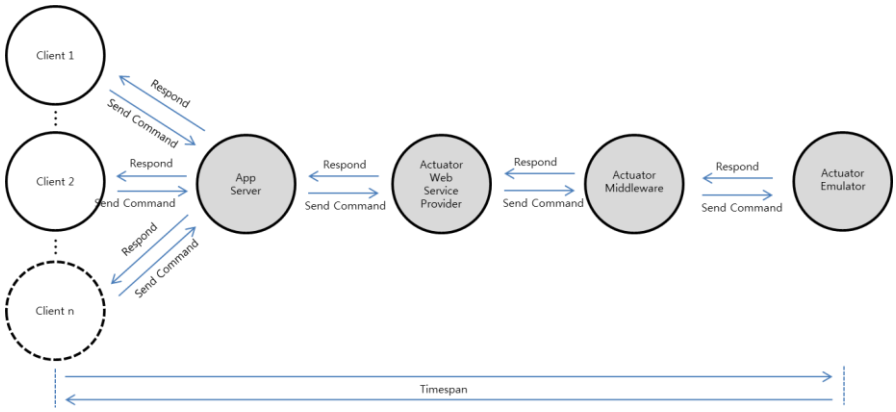

그림 97 응용서버 기반 구동체 제어

그림 97은 응용서버 기반의 IOT 시스템에서 클라이언트가 구동체를 제어하는 과정이다. 1대, 5대와 25대 클라이언트가 동시 접속경우를 각각 20회 실험을 진행한다.

표 19 구동체 제어 응답시간

	Base on App Server			Base on Service Provider		
	1 Client	5 Client	25 Client	1 Client	5 Client	25 Client
1	168.02ms	540.66ms	2242.38ms	98.99ms	446.40ms	1946.84ms
2	146.02ms	588.86ms	2210.98ms	91.99ms	442.80ms	2067.64ms
3	108.01ms	555.46ms	2498.77ms	67.99ms	445.20ms	1955.16ms
4	123.01ms	549.06ms	2345.31ms	91.02ms	398.60ms	2251.28ms
5	141.00ms	530.65ms	2415.64ms	77.01ms	446.60ms	2202.00ms
6	129.02ms	405.84ms	2759.40ms	70.98ms	446.80ms	2266.40ms
7	131.02ms	571.46ms	2730.04ms	71.00ms	415.80ms	2391.80ms
8	147.02ms	539.85ms	2839.04ms	119.00ms	440.20ms	2225.24ms
9	112.01ms	567.25ms	2812.92ms	95.00ms	568.80ms	2552.20ms
10	109.01ms	514.45ms	2809.08ms	72.00ms	479.20ms	2515.96ms
11	123.01ms	597.66ms	3404.06ms	119.00ms	434.60ms	2643.92ms
12	130.02ms	542.65ms	2850.00ms	88.99ms	488.60ms	2804.20ms
13	101.02ms	476.25ms	3063.79ms	166.98ms	471.80ms	2595.40ms
14	140.01ms	509.05ms	3218.20ms	110.01ms	452.60ms	2706.44ms
15	147.01ms	579.46ms	3048.51ms	132.01ms	504.20ms	1977.52ms
16	112.01ms	487.65ms	3196.40ms	111.01ms	433.99ms	2954.44ms
17	124.02ms	581.06ms	3386.62ms	99.00ms	518.80ms	1914.36ms
18	178.01ms	601.86ms	3518.47ms	99.01ms	445.40ms	2864.20ms
19	189.02ms	557.06ms	3570.00ms	72.00ms	573.80ms	2036.72ms
20	106.01ms	658.47ms	3436.66ms	91.99ms	484.40ms	1852.04ms

표 19는 그림 96과 그림 97에 대해 성능분석 결과이다. 응용서버 기반의 IOT 시스템에서 클라이언트가 간접방식으로 센서웹 공급자와 접속하기 때문에 직접 접속방식을 사용하는 서비스 공급자 기반의 IOT 시스템보다는 구동체 제어하고 응답하는 시간이 더 길다.

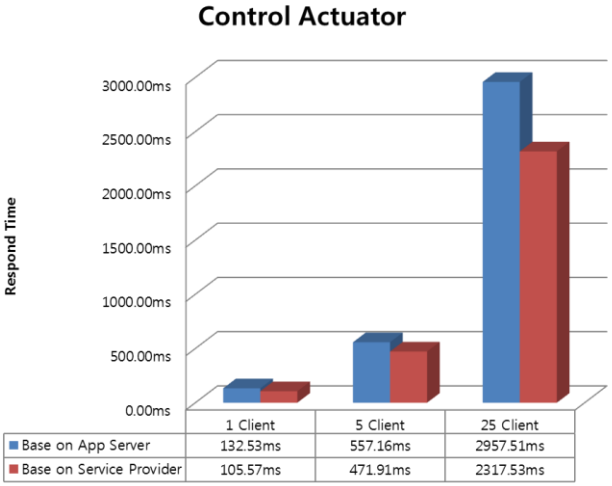

그림 98 구동체 제어 평균 응답시간 비교

그림 98은 구동체 제어 평균 응답하는 시간 비교 그래프이다. 1개 클라이언트가 구동체 제어하는 경우에 응용서버 기반의 시스템 구조를 사용하는 시스템은 132.53ms 걸리며, 서비스 공급자 기반의 시스템 구조를 사용하는 시스템은 105.57ms 걸린다. 5개 클라이언트가 동시에 구동체 제어하는 경우에 응용서버 기반의 시스템 구조를 사용하는 시스템은 557.16ms 걸리며, 서비스 공급자 기반의 시스템 구조를 사용하는 시스템은 471.91ms 걸린다. 25개 클라이언트가 동시에 구동체 제어하는 경우에 응용서버 기반의 시스템 구조를 사용하는 시스템은 2957.51ms 걸리며, 서비스 공급자 기반의 시스템 구조를 사용하는 시스템은 2317.53ms 걸린다.

V. 저작도구를 기반의 실내 IOT 시스템

1 실내 IOT 시스템 구조 방안

실내 IOT 시스템은 센서웹 시스템, 구동체웹 시스템, GIS엔진 및 응용 시스템으로 구성된다. 그런데 III장의 센서웹과 IV장의 구동체웹의 시스템 구조에 대해 서비스 공급자하고 응용서버 기반의 시스템 구조를 별도 설계한다, 따라서 실내 센서웹과 구동체웹을 통합되는 IOT 시스템은 당연히 두 자기 시스템 구조가 존재한다.

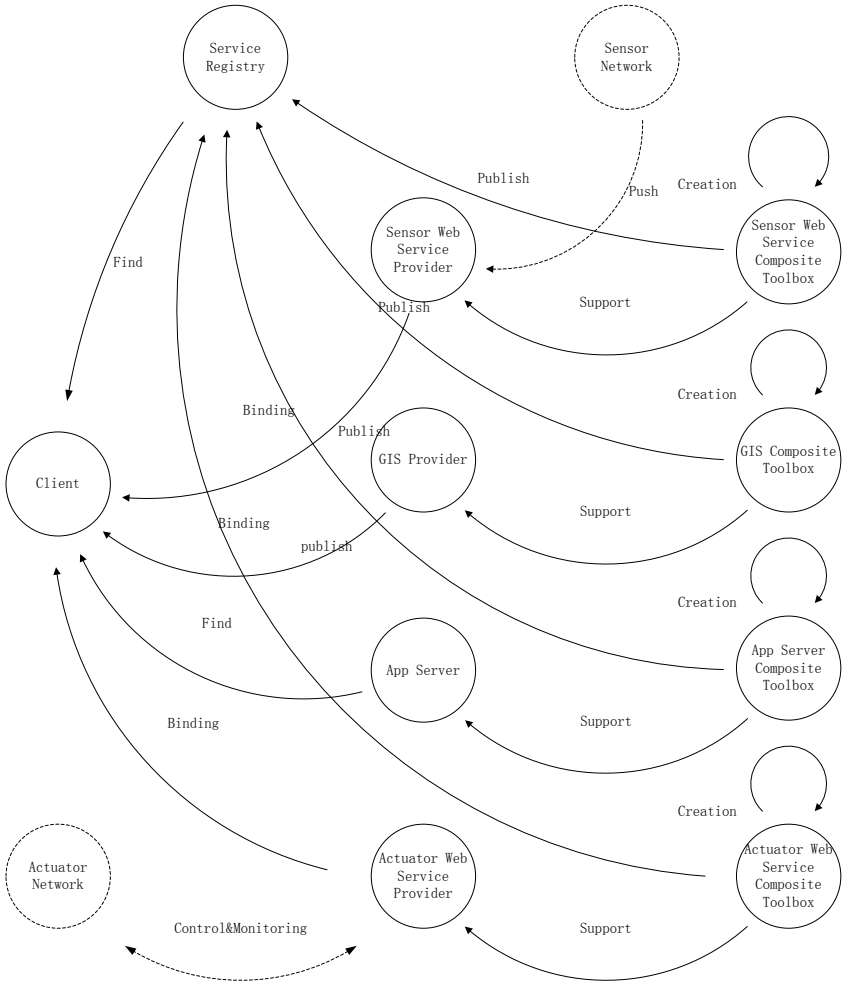

그림 99 서비스 공급자 기반 실내 IOT 시스템 구조방안

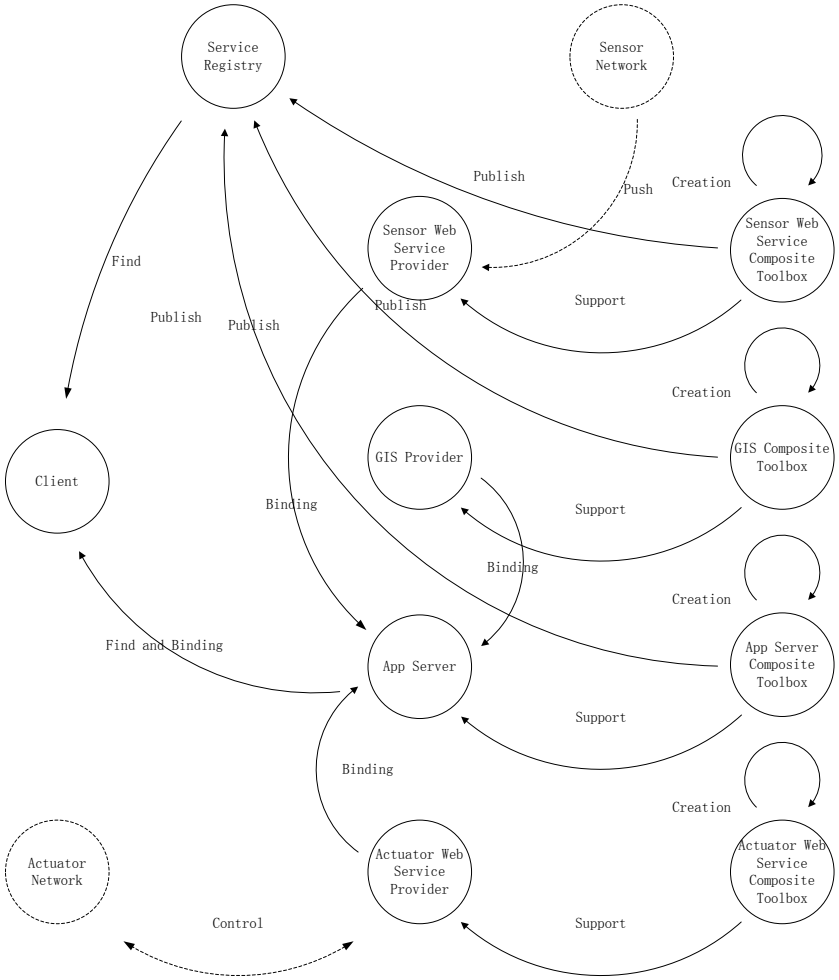

그림 100 응용서버 기반 실내 IOT 시스템 구조방안

그림 99는 서비스 공급자 기반 실내 IOT 시스템 구조방안이다. 센서웹 시스템은 센서웹 공급자, 센서웹 저작도구, 센서 미들웨어로 구성된다. 센서웹은 센서 네트워크에 존재하는 센서 노드의 센싱 데이터, 센서 정보 등 여러 가지 서비스를 제공한다. 구동체웹 시스템은 구동체웹 공급자, 구동체웹 저작도구, 구동체 미들웨어로 구성된다. 구동체웹은 구동체 네트워크와 연결되는 구동체에 대해 모니터링, 제어 등 다양한 서비스를 제공한다. GIS엔진은 GIS 공급자 및 GIS 서비스 저작도구로 구성된다. GIS 공급자는 지도와 관련된 서비스를 제공하고 GIS 서비스 저작도구는 지도와 관련된 서비스 콘텐츠 생성 및 관리역할을 수행한다. 응용시스템은 클라이언트, 응용서버 서비스 공급자, 응용서버 저작도구로 구성된다. 클라이언트는 사용자에게 특정 지도 서비스상의 센서와 구동체에 대해 모니터링

을 제공한다. 응용서버 서비스 공급자는 지도 서비스의 바인딩 정보 및 지도상의 센서노드 및 구동체 노드의 위치정보를 제공하고 Smart Control을 통해 실내 지정공간의 쾌적 상태를 자동적으로 조정하는 역할을 수행한다. 응용서버 저작도구는 지도 서비스 바인딩과 오브젝트의 지도상의 위치 정보를 지정하는 역할을 담당한다.

그림 100은 응용서버 기반 실내 IOT 시스템 구조방안이다. 그림 99의 방안을 비교하여 응용서버의 역할을 차이점을 알 수 있다. 서비스 공급자 기반 IOT 시스템의 응용서버가 센서 노드 및 구동체 노드의 위치 정보, 공급 서비스 정보를 제공하여 클라이언트가 이를 통해 직접적으로 센서웹 공급자, 구동체웹 공급자에서 센서와 구동체 정보를 요청한다. 응용서버 기반 실내 IOT 시스템의 응용서버는 센서웹 공급자, 구동체웹 공급자가 제공하는 API들이 통합하여 클라이언트에게 제공한다. 그래서 클라이언트가 서비스 등록자에서 응용서버의 주소를 검색하여 응용서버를 통해 센서웹 공급자, 구동체웹 공급자를 방문한다.

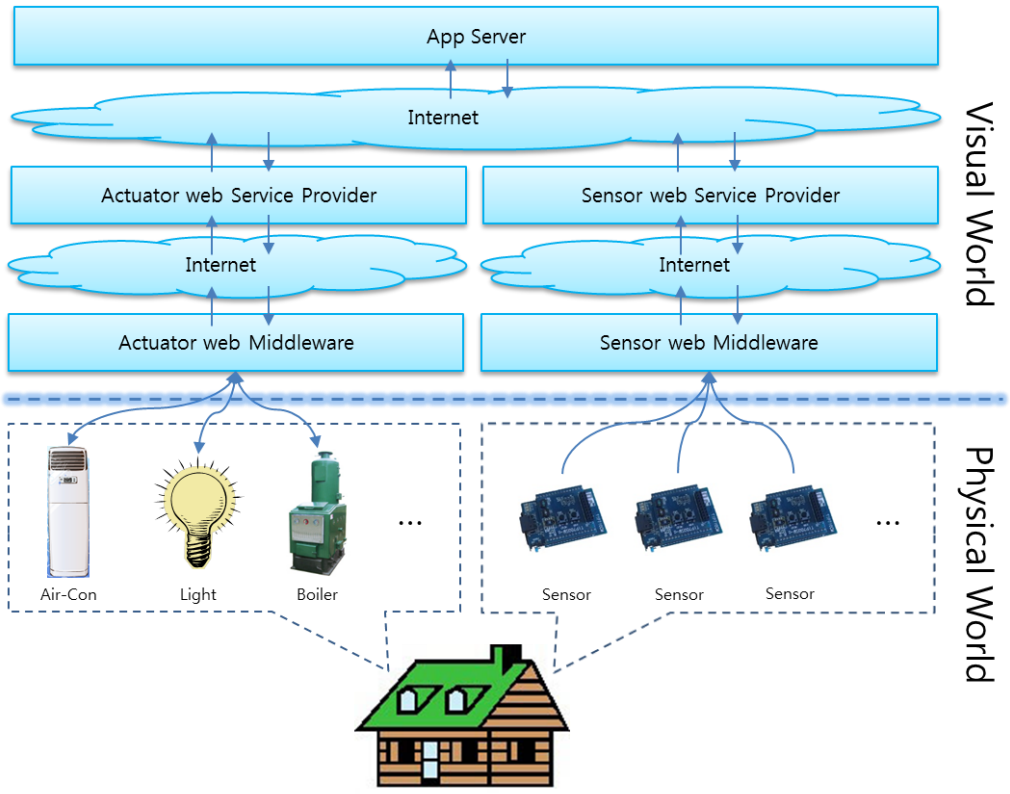

그림 101 실내 지능 제어 시스템 구조

그림 101은 실내 지능 제어 시스템의 구조이다. 위 구조는 Visual World와 Physical World구역으로 구성된다. Visual World 구역은 응용서버, 구동체웹 공급자, 센서웹 공급자, 센서 미들웨어와 구동체 미들웨어라는 모듈로 구성된다. 구동체 미들웨어와 센서 미들웨어는 현실 세계의 디바이스와 연동하기 위한 부분이다. 센서 미들웨어가 센싱 데이터 수집역할을 수행하며 구동체 미들웨어는 구동체의 동작상태 수집역할과 제어명령을 전송하는 역할을 담당한다. 서비스 공급자의 서비스 콘텐츠가 외부 응용서버를 제공한다. 응용서버가 실시간 실내의 센서 노드에서 수신된 센싱 데이터를 감시하여 실내의 환경상태를 판단한다. Physical World구역은 실내환경 상태를 감지하는 센서와 실내 환경을 조정하는 가전제품으로 구성된다. 본 논문이 주로 소프트웨어 입장에서 지능 환경 제어 시스템의 구현 및 평가를 논하므로 실제 USN연결 기능을 갖고 있는 가전제품 대신 같은 기능을 수행하는 가전 에뮬레이터를 사용한다.

2 실내 IOT 시스템 설계

2.1 서비스 공급자 기반 실내 IOT 시스템

2.1.1 서비스 공급자 기반 실내 IOT 시스템의 인터페이스 프로토콜

실내 IOT 시스템은 센서웹 시스템, 구동체웹 시스템을 기반으로 통합되는 시스템이다. 그래서 본 절에서 실내 IOT 시스템의 구현 결과가 센서웹 및 구동체웹을 포함한다.

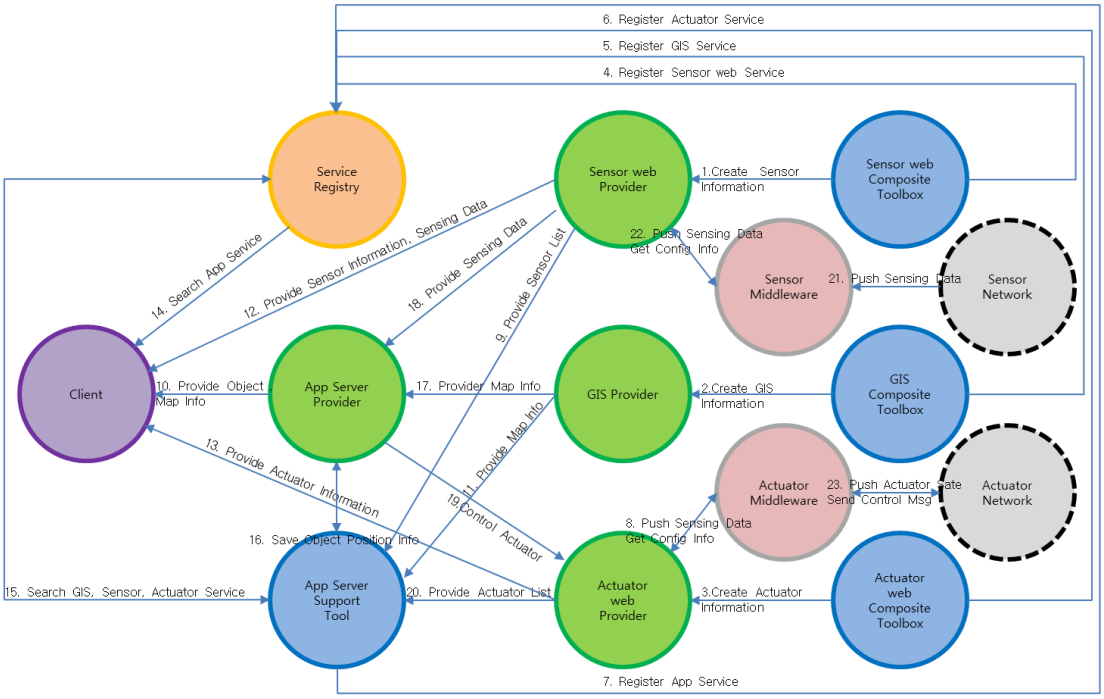

그림 102 서비스 공급자 기반 IOT 시스템 구조

그림 102는 서비스 공급자 기반 IOT 시스템의 모듈들 간의 협력 과정이다. 본 시스템은 클라이언트, 서비스 등록자, 응용서버, 응용서버 저작도구, 센서웹 공급자, GIS 공급자, 구동체웹 공급자, 센서 미들웨어, 구동체 미들웨어, 센서웹 저작도구, GIS 서비스 저작도구, 구동체웹 저작도구, 센서 네트워크, 구동체 네트워크라는 모듈들로 구성된다. 그리고 각 모듈간의 협력과정은 앞의 설계 단계에서 이미 상세히 언급하였고 다음으로 실제 구현할 때의 각 모듈간의 API의 참조에 대해 살펴본다. 본 분산 처리 시스템은 SOAP을 기반으로 구현함으로써 API는

Microsoft .NET 환경의 WCF 기술을 사용하여 구현한다.

표 20 서비스 공급자 기반 IOT 시스템의 API

Module Name	API Name	Description		Applied At
		Type	Operation	
Service Registry	Service Publish Interface Protocol	Service Information	Write	4, 5, 6, 7
	Service Search Interface Protocol	Service Information	Search, Read	14, 15,
GIS Service Provider	GIS Provider Interface Protocol	Map Information	Read	11, 17
	GIS Content Interface Protocol	Map Information	Read, Write	2
Sensor web Service Provider	Sensor web Service Provider Interface Protocol	Sensor List	Read, Search	9
		Sensor Information	Read	12
		Sensing Data	Read	12, 17
	Sensor web Service Content Interface Protocol	Sensor List	Read, Search	1
		Sensor Information	Write	1
Actuator web Service Provider	Actuator web Service Provider Interface Protocol	Actuator List	Read, Search	20
		Actuator Information	Read	13
		Control	Write	19
	Actuator web Service Content Interface Protocol	Actuator List	Read, Search	3
		Actuator Information	Write	3
App Server	App Service Provider Interface Protocol	Map Information	Read	10
		Object Information	Read	10
	App Service Content Interface Protocol	Map Information	Read	16
		Object Information	Read, Write	16

표 20은 서비스 공급자 기반 IOT 시스템의 각 모듈이 제공하는 API이다. 서비스 등록자는 Service Publish Interface Protocol 및 Service Search Interface Protocol을 포함한다. Service Publish Interface Protocol은 서비스 정보의 등록 역할을 제공하며, Service Search Interface Protocol은 서비스 정보의 검색역할을 제공한다. GIS 공급자는 GIS Service Provider Interface Protocol 및 GIS Content Interface Protocol을 포함한다. GIS Service Provider Interface Protocol은 지도 데이터, 빌딩 정보, 층 정보, 방 정보를 제공하며, GIS Content Interface Protocol은 지도 데이터, 빌딩 정보, 층 정보, 방 정보의 생성 및 관리를 제공한다. 센서웹 공급자는 Sensor Web Provider Interface Protocol 및 Sensor Web Service Content Interface Protocol을 포함한다. Sensor Web Provider Interface Protocol은 센서 정보, 센서 검색, 센싱 데이터를 공급하며, Sensor Web Service Content Interface Protocol 센서 정보의 생성 및 관리를 제공한다. 구동체웹 공급자는 Actuator Web Provider Interface Protocol 및

Actuator Web Content Interface Protocol을 포함한다. Actuator Web Provider Interface Protocol은 구동체 정보, 구동체 검색, 구동체 상태, 구동체 제어를 제공하며, Actuator Web Content Interface Protocol 구동체 정보의 생성 및 관리를 제공한다. 응용서버는 App Server Provider Interface Protocol 및 App Service Content Interface Protocol을 포함한다. App Server Provider Interface Protocol은 지도관련 정보(GIS Service를 제공하는 서비스 중계), 오브젝트의 정보를 제공하며, App Service Content Interface Protocol 지도 서비스 공급, 오브젝트정보의 생성 및 관리를 제공한다.

2.1.2 환경 설정 단계

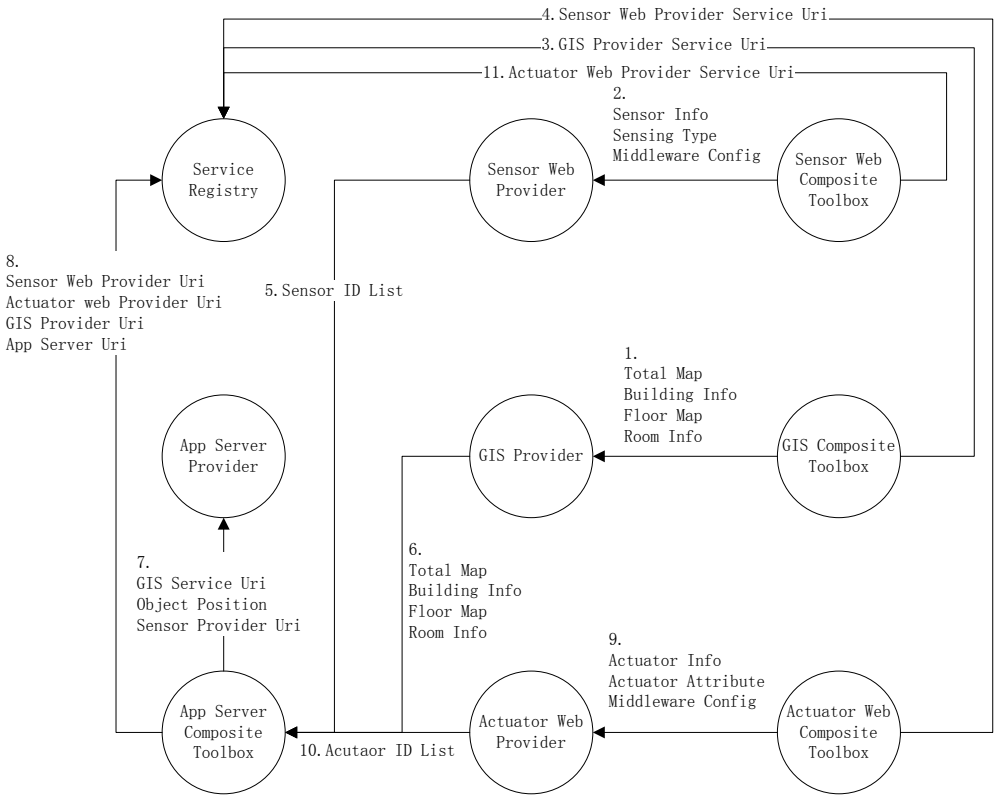

그림 103 서비스 공급자 기반 실내 IOT 시스템 환경 설정 단계

그림 103은 서비스 공급자 기반 실내 IOT 시스템의 환경 설정 단계이다. 통

합 서비스의 배치단계는 서비스의 필수 콘텐츠를 제대로 클라이언트에게 공급하고 검색서비스를 제공하기 위해 생성한다.

Total Map은 실외 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Building Info는 빌딩의 이름, 위치 정보, 층 이름을 포함한다. Floor Map은 실내 지도의 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Room Info는 방의 이름, 방 위치를 포함한다. Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. 센서 Type는 센서의 센싱 타입 정보를 의미한다. Middleware Config는 센서 미들웨어의 ID, IP주소과 접속권한 정보를 포함한다. Service Info는 각각 서비스 공급자의 서비스 이름, 검색 키워드, 접속 주소, 서비스 유형을 포함한다. 센서 List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 센서 ID들을 의미한다. GIS Service Uri는 GIS 공급자의 접속주소를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. Actuator List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 구동체 ID들을 의미한다. Actuator Position는 구동체의 지도상의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다. 센서웹 공급자 Uri는 센서웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

GIS 서비스 저작도구는 지도 관련 데이터를 외부에서 GIS 콘텐츠 API를 통해 GIS 공급자의 DB에 저장한다. 센서웹 저작도구는 각 센서노드의 정보를 센서웹 콘텐츠 API를 통해 생성하여 DB에 저장하며 미들웨어의 배치 정보를 생성한다. GIS 서비스 저작도구가 GIS 공급 서비스의 정보를 서비스 등록자에게 등록한다. 센서웹 저작도구가 센서웹 공급자의 정보를 서비스 등록자에게 등록한다. 센서웹 공급자가 제공하는 센서 리스트 정보와 GIS 공급자가 제공하는 지도 데이터를 응용서버 저작도구에게 제공한다. 응용서버 저작도구가 각 센서 노드와 지도의 위치정보를 생성하여 저장하고 응용서버의 서비스 정보를 등록한다. 구동체웹 저작도구는 각 구동체 노드 정보를 Actuator Web Content API를 통해 생성하고 DB에 저장하여 미들웨어의 배치 정보를 생성한다. 구동체웹 공급자가 제공하는 센서 리스트 정보를 응용서버 저작도구에 제공한다. 구동체 저작도구가 구동체웹 공급자 서비스 정보를 서비스 등록자에게 등록한다.

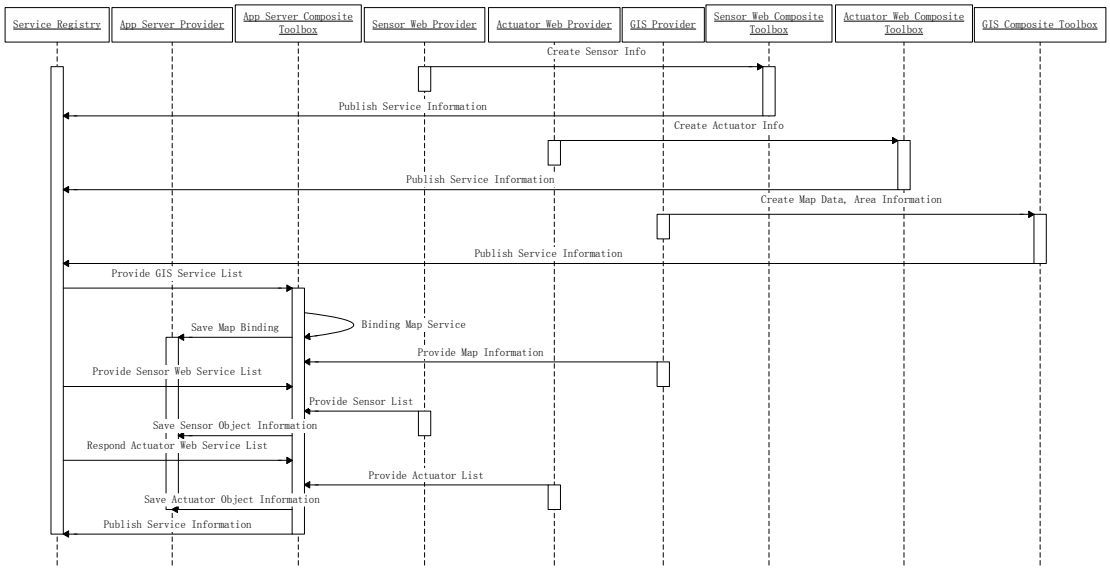

그림 104 서비스 공급자 기반 실내 IOT 시스템 환경 설정 단계 시퀀스

그림 104는 서비스 공급자 기반 실내 IOT 시스템의 환경 설정 단계 시퀀스 다이어그램이다. 위의 과정은 앞에서 기술했던 센서웹 시스템과 구동체웹 시스템의 환경 설정 단계를 통합한 것이다. 그래서 서비스 등록자, 응용서버 서비스 공급자, 응용서버 저작도구, 센서웹 공급자, 구동체웹 공급자, GIS 공급자, 센서웹 저작도구, 구동체웹 저작도구와 GIS 서비스 저작도구로 구성된다.

처음에 센서웹 저작도구, 구동체웹 저작도구와 GIS 서비스 저작도구가 센서 오브젝트 정보, 구동체 오브젝트 정보 및 지도 서비스 콘텐츠를 생성하여 센서웹 공급자, 구동체웹 공급자 와 GIS 공급자에 저장한다. 다음에 센서웹 공급자, 구동체웹 공급자 와 GIS 공급자의 서비스 정보를 서비스 등록자에 등록하고 응용서버 저작도구는 서비스 등록자에 등록된 GIS 서비스 정보를 검색하여 Map Service를 바인딩하고 응용서버 서비스 공급자에 저장한다. 응용서버 저작도구는 바인딩된 지도 서비스 정보에 따라서 지도 데이터를 요청하여 가시화를 제공한다. 서비스 등록자에서 센서웹 시스템 및 구동체웹 시스템 정보를 요청하여 관리자가 각 공급 서비스에 제공하는 오브젝트(센서, 구동체)를 선택하고 지도상에 오브젝트 콘텐츠를 생성하여 응용서버 서비스 공급자에 저장한다. 콘텐츠를 생성하고 나서

응용서버의 공급 서비스 정보를 서비스 등록자에 저장한다.

2.1.3 검색 단계

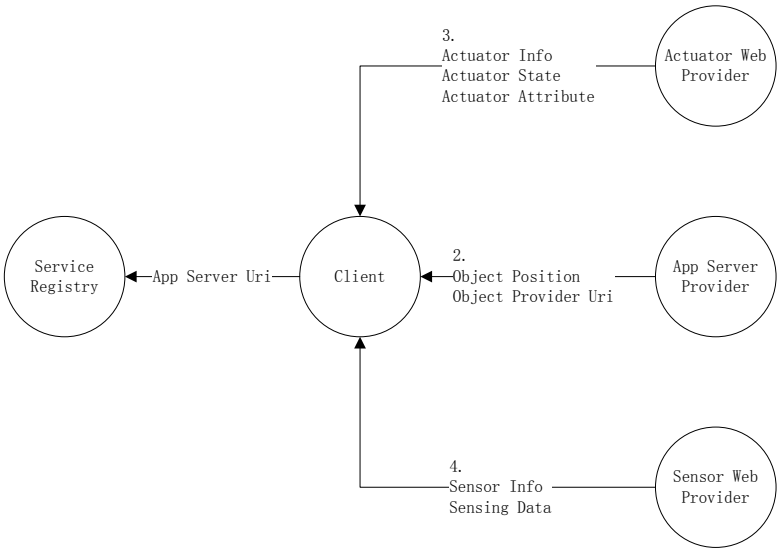

그림 105 서비스 공급자 기반 실내 IOT 시스템 검색 단계

그림 105는 서비스 공급자 기반 실내 IOT 시스템의 검색 단계이다. 통합 서비스의 검색 단계는 클라이언트가 서비스 등록자에서 쉽게 응용서버, 센서 노드 및 구동체 노드를 찾을 수 있도록 한다.

Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. 센싱 데이터는 센서의 실시간 센싱 데이터를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. 센서웹 공급자 Uri는 센서웹 공급자의 접속주소를 의미한다. Actuator Info는 구동체 노드의 이름, 구동체 ID를 포함한다. Actuator State는 구동체의 동작상태 정보를 의미한다. Actuator Attribute는 구동체의 작업 속성을 의미한다. Actuator Position는 구동체의 지도상의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

클라이언트는 사용자가 입력하는 서비스검색 키워드를 통해 서비스 등록자에

게 서비스 리스트를 요청하고 응용서버에 접속하여 센서의 위치 정보를 요청한다. 그리고 구동체 ID를 통해 구동체 정보를 요청하고 센서 ID를 통해 센서 정보를 요청한다.

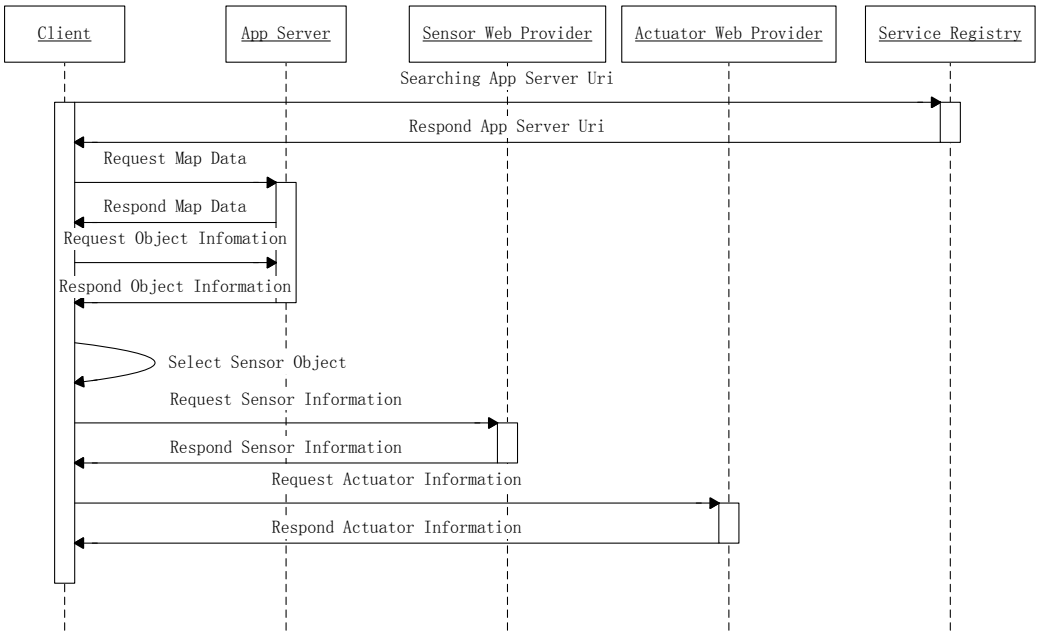

그림 106 서비스 공급자 기반 실내 IOT 시스템 검색 단계 시퀀스

그림 106은 서비스 공급자 기반 실내 IOT 시스템의 검색 단계 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 응용서버, 센서웹 공급자, 구동체웹 공급자와 서비스 등록자로 구성된다.

위의 과정은 처음에 클라이언트가 서비스 등록자에서 응용서버의 서비스 정보를 검색하여 접속한다. 다음에 응용서버가 제공하는 지도 서비스 정보에 따라서 지도 서비스 콘텐츠를 요청 받아 가시화를 제공하며 오브젝트(센서, 구동체)의 위치정보를 요청 받아서 출력한다. 사용자가 원하는 센서나 구동체 노드를 선택하여 대응의 서비스 공급자에게 세부정보를 요청하고 가시화한다.

2.1.4 지능 제어 단계

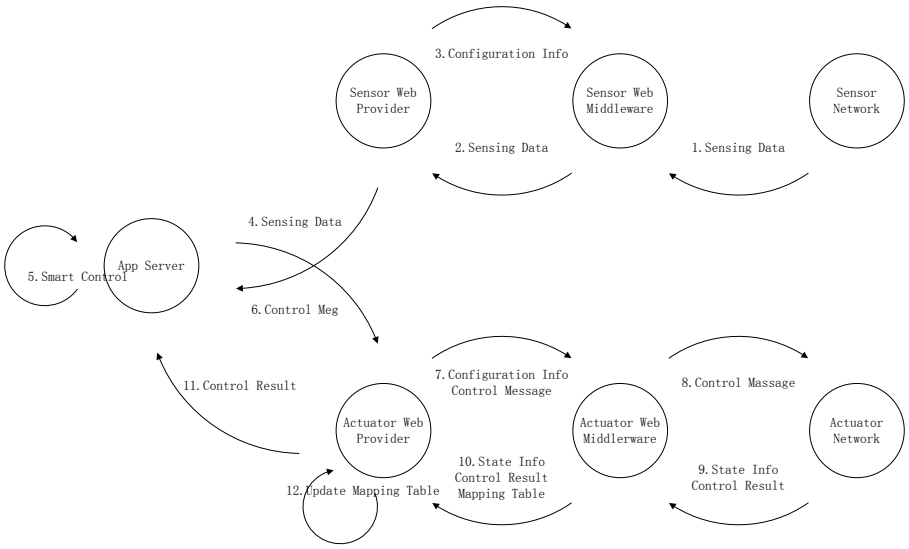

그림 107 서비스 공급자 기반 실내 IOT 시스템 지능 제어 단계

그림 107은 서비스 공급자 기반 실내 IOT 시스템의 지능 제어 단계이다. 이 단계는 응용서버가 저장된 지정 지역 정보에 따라서 센서 노드의 센싱 데이터를 실시간 감시를 통해 이 지역의 가전을 자동제어 하여 공간의 환경상태가 최적화 되도록 한다.

센싱 데이터는 센서 노드가 환경에 대한 측정하는 센싱 데이터를 의미한다. Configuration Info는 미들웨어의 ID, IP주소와 서비스 공급자의 접속권한 정보를 의미한다. Smart Control는 지능적으로 실내 환경정보에 이용해 구동체를 제어한다. Control Msg는 구동체 제어 메시지를 의미한다. Control Result는 제어 명령을 내리고 닫는 응답 메시지를 의미한다. Update Mapping Table는 구동체 미들웨어부터 수신하는 구동체 주소 관계정보를 갱신한다. State Info는 구동체의 동작 상태 정보를 의미한다.

센서 네트워크로 연결되는 센서 노드가 센싱 데이터를 실시간 센서 미들웨어를 통해 센서웹 공급자에 전송하여 저장한다. 센서 네트워크로 연결되는 구동체의 연결 상태에 따라서 구동체 미들웨어와 구동체웹 공급자 가 매핑 테이블을 생

성하고 관리한다. 응용서버가 자기가 갖고 있는 공간 및 오브젝트 정보에 따라서 실내의 환경 센싱 데이터(온도, 습도, 조도 등)를 요청한다. 다음에 요청 받은 환경 데이터를 퍼지 제어 모듈을 통해 제어대상(구동체) 및 제어 내용(명령)을 생성하여 구동한다. 구동체웹 공급자가 응용서버에서 제어 메시지를 수신하여 매핑 테이블에 저장된 구동체 라우팅 정보에 따라서 구동체 미들웨어를 통해 전송한다.

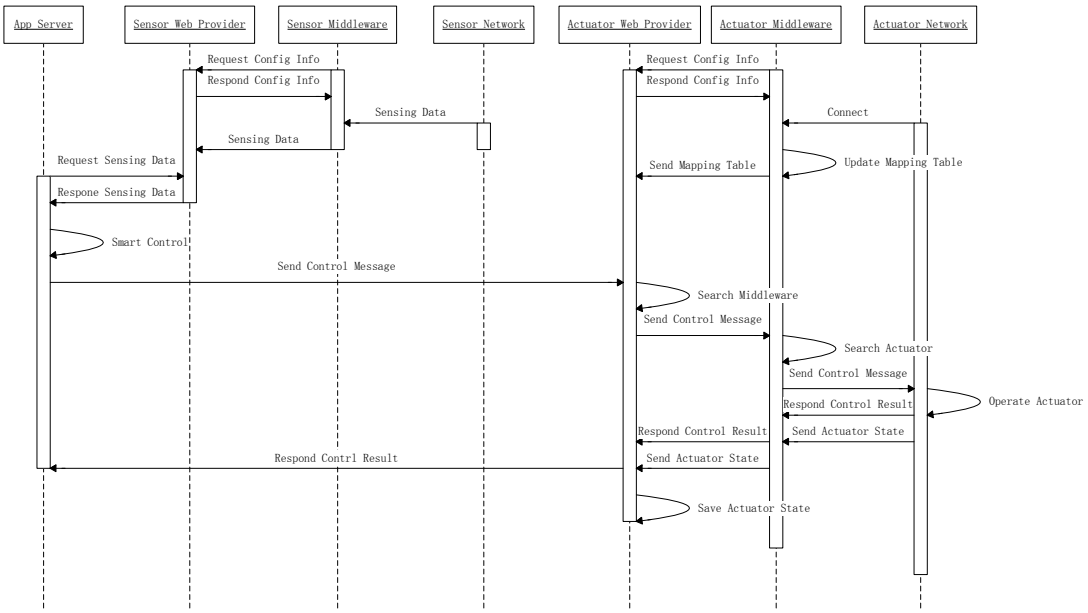

그림 108 서비스 공급자 기반 실내 IOT 시스템 지능 제어 단계 시퀀스

그림 108은 서비스 공급자 기반 실내 IOT 시스템의 지능 제어 단계 시퀀스 다이어그램. 위의 과정은 응용서버, 센서웹 공급자, 센서 미들웨어, 센서 네트워크, 구동체웹 공급자, 구동체 미들웨어, 구동체 네트워크로 구성된다.

처음에 센서 미들웨어 및 구동체 미들웨어가 자기를 지원하는 서비스 공급자에게 배치 정보를 요청하여 배치한다. 다음에 센서 네트워크가 계속 수집하는 센싱 데이터를 센서웹 공급자에게 보내고 저장한다. 그리고 구동체 미들웨어가 실시간 구동체 네트워크의 연결상태를 감시하여 매핑 테이블을 생성하고 관리한다. 응용서버가 센싱 데이터에 따라서 Smart Control을 구동하여 Control Message를 생성하고 구동체웹 공급자에게 보낸다. 구동체웹 공급자 및 구동체 미들웨어가

기존 매핑 테이블을 통해 목적 구동체를 찾아서 제어한다. 구동체가 Control Message를 수신하면 실행하여 제어 결과를 응답하고 변경후의 구동체 동작상태를 구동체웹 공급자에게 보내고 저장한다.

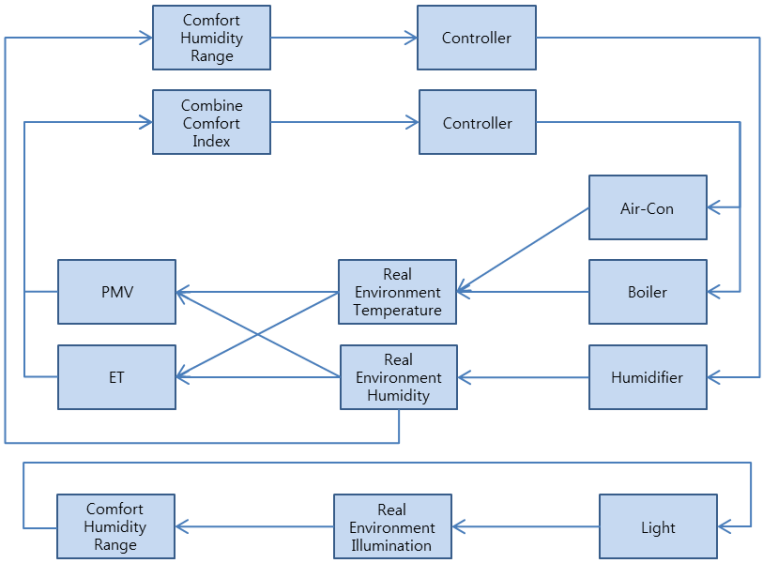

그림 109 지능 제어 개념도

그림 109는 지능 제어의 개념 개념도이다. 본 논문은 주로 에어컨, 보일러, 가습기, 조명 네 가지 가전제품을 통해 실내의 환경을 조정한다. 에어컨 하고 보일러가 실내의 온도 속성을 제어하며, 가습기가 실내의 습도를 제어하고 Light는 실내의 조도를 조정한다. PMV와 ET 지수를 이용하여 실내 환경의 쾌적상태를 쉽게 판단할 수 있다.

본 논문은 PMV, PPD, ET지수를 통해서 실내 쾌적 상황을 판단하는 것이다. 그런데 이 세 가지 쾌적 지수의 룰은 다르지만 실제 기능은 비슷하다. 환경 데이터 측정 오차가 발생할 때 다양한 쾌적지수가 통일한 표준을 사용하지 않아서 사용자가 정확하게 알 수 없다. 이 문제를 해결하려면 지수를 정규화하여 통합해야 한다.

표 21 PMV과 ET 쾌적상황 룰

PMV Index	ET Index	Comfort State
2.5~3	35~41	Hot
1.5~2.5	29~35	Warm
0.5~1.5	23~29	Slightly Warm
-0.5~0.5	18~23	Normal
-1.5~-0.5	13~18	Slightly Cool
-2.5~-1.5	8~13	Cool
-3~-2.5	4~8	Cold

표 21은 PMV과 ET 쾌적상황 룰 이다. PMV지수의 범위는 -3부터 3까지이고 ET지수의 범위는 4부터 41까지이다. 그런데 둘은 같은 쾌적 상황을 가리킨다. 여기서 정규화 수식을 이용하여 PMV과 ET지수에 각 각 적용하면 0부터 1까지의 지수로 전환할 수 있다.

종합 쾌적 수식:

$$\mathrm{PMV_{Nor}} = \frac{\mathrm{PMV_{Value}} - \mathrm{PMV_{Min}}}{\mathrm{PMV_{Max}} - \mathrm{PMV_{Min}}}$$

수식 7

$$\mathrm{ET_{Nor}} = \frac{\mathrm{ET_{Value}} - \mathrm{ET_{Min}}}{\mathrm{ET_{Max}} - \mathrm{ET_{Min}}}$$

수식 8

$$\mathrm{Combine_{Value}} = \frac{\mathrm{PMV_{Nor}} + \mathrm{ET_{Nor}}}{2}$$

수식 9

수식 7은 PMV지수 정규화 식이고 수식 8은 ET지수 정규화 식이다. 수식 9는 PMV와 ET종합 지수를 계산하는 식이다. 이 세 식을 이용하여 쾌적 지수를 쉽게 계산할 수 있다. 종합 쾌적 지수를 사용자에게 쉽게 알려 주려면 종합 쾌적 상황 룰이 있어야 한다. 그래서 PMV와 ET 정규화 상황 룰을 평균 처리하면 종합 상황 룰을 얻을 수 있다.

표 22 결합 쾌적 지수 룰

Comfort State	PMV Value	ET Value	Combine Value
Hot	0.916~1	0.838~1	0.877~1
Warm	0.75~0.916	0.676~0.838	0.713~0.877
Slightly Warm	0.583~0.75	0.514~0.676	0.5485~0.713
Normal	0.416~0.583	0.378~0.514	0.397~0.5485
Slightly Cool	0.25~0.416	0.243~0.378	0.2465~0.397
Cool	0.083~0.25	0.108~0.243	0.0955~0.2465
Cold	0~0.083	0~0.108	0~0.0955

표 22는 종합 쾌적상황 룰이다. 이상의 룰에 따라서 실내 환경의 쾌적 상황을 명확하게 알 수 있다.

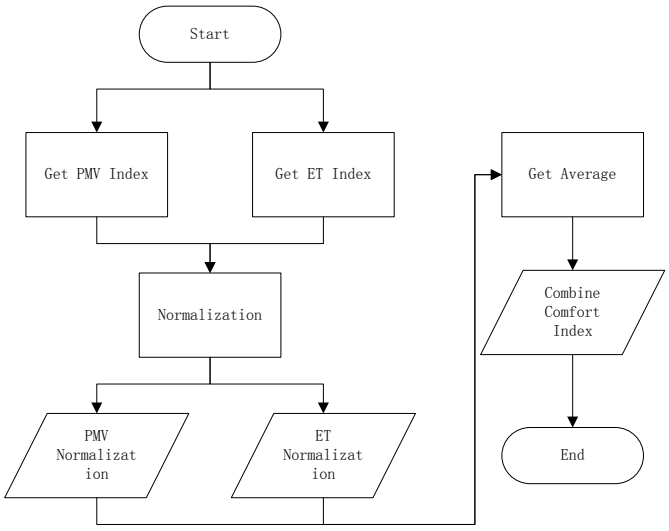

그림 110 쾌적 지수 종합 순차도

그림 110은 쾌적 지수 종합과정 순차도 이다. 처음에 서비스 공급자에게 특정한 방의 PMV지수와 ET지수를 요청한다. 다음에 PMV와 ET를 따로 정규화하고 PMV와 ET의 정규화 지수를 얻는다. 다음에 정규화 된 PMV와 ET의 평균을 구하고 종합 쾌적 지수를 얻으며 최종 PPD룰을 통해서 종합쾌적 상황을 얻는다.

표 23 쾌적 상태 퍼지 룰

PMV State	ET State	Comfort State	PMV State	ET State	Comfort State
Cold	Cold	Cold	Slightly Warm	Normal	Normal
Cold	Cool	Cold	Normal	Normal	Normal
Cold	Slightly Cool	Cold	Hot	Hot	Hot
Cool	Cool	Cold	Hot	Warm	Hot
Cool	Cold	Cold	Hot	Slightly Warm	Hot
Cool	Slightly Cool	Cold	Warm	Warm	Hot
Slightly Cool	Cold	Cold	Warm	Hot	Hot
Slightly Cool	Cool	Cold	Warm	Slightly Warm	Hot
Slightly Cool	Slightly Cool	Cold	Slightly Warm	Hot	Hot
Normal	Slightly Cool	Normal	Slightly Warm	Warm	Hot
Normal	Slightly Warm	Normal	Slightly Warm	Slightly Warm	Hot
Slightly Cool	Normal	Normal			

표 23은 실내 온도를 최적화하기 위한 퍼지 룰이다. 예를 들어 어떤 실내 공간의 PMV 환경상태가 Cool이고 ET 환경상태가 Slightly Cool이면 결합 쾌적 상태는 Cold를 의미한다. 즉 If (PMV State) and (ET State) then (Comfort State) 이다. Comfort State가 Cold이면 실내 온도를 올리도록 하고 Hot 이면 내리도록 한다. 이 방식을 사용하여 실내 공간을 Normal 수준으로 유지할 수 있다.

본 논문에서 제시하는 Smart Control의 알고리즘은 환경정보의 수집하고 가전제품의 제어 두 부분으로 구성한다. 다음 각각의 부분의 흐름도에 대해 살펴본다.

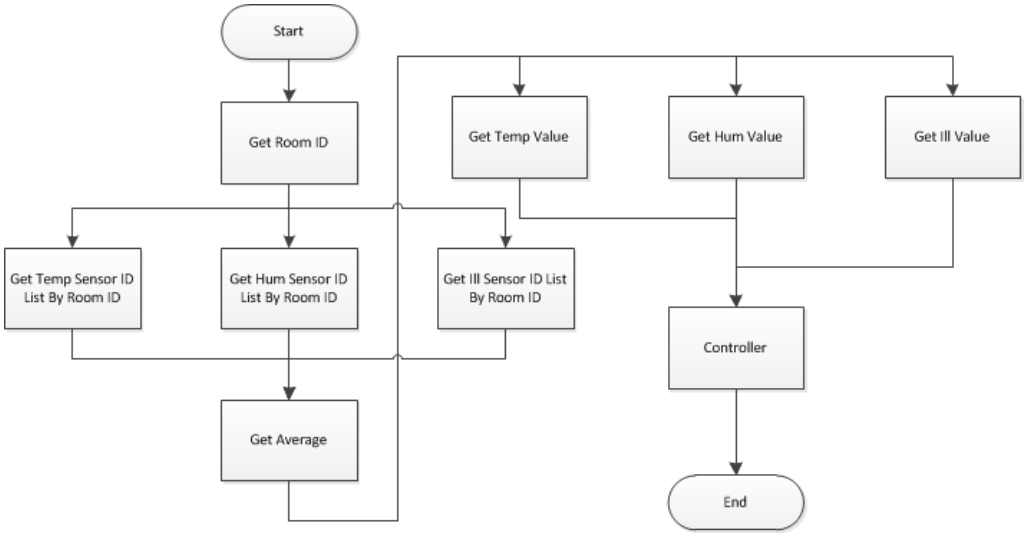

그림 111 실내 환경정보 수집 알고리즘

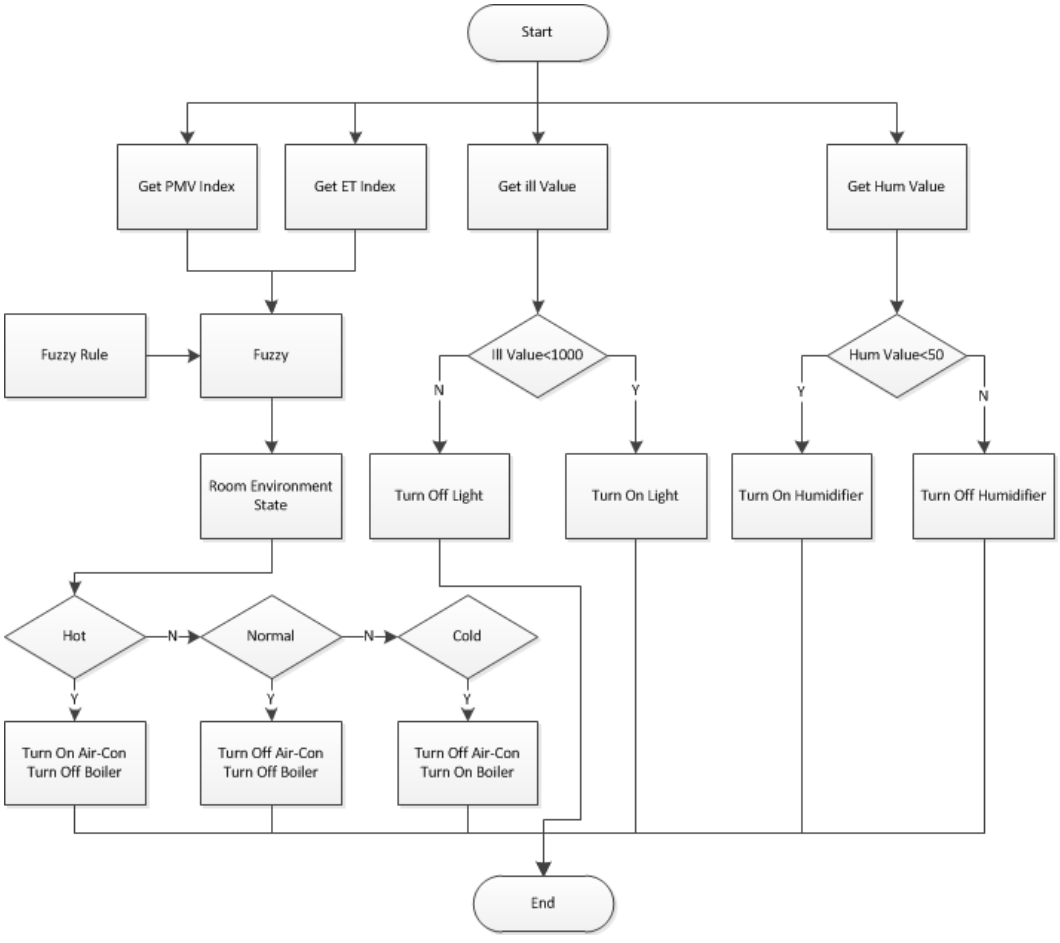

그림 112 실내 구동체 제어 알고리즘

그림 111은 실내 환경 정보를 수집하는 과정이다. 앞에서 제시한 센서 웹 시스템이 제공하는 센싱 데이터 요청 인터페이스하고 응용서버의 DB에서 저장된 센서노드의 위치 정보와 실내 공간정보를 통해 쉽게 특정공간에 있는 센서노드를 찾을 수 있다. Smart Control모듈은 그 공간의 센서 노드의 센싱 타입에 따라서 분류하고 기록한다. 같은 공간에서 같은 감지 기능을 수행하는 센서노드가 여러 개를 존재하는 경우도 있기 때문에 평균 수치를 구하여 공간의 환경상태를 파악하며 개별의 센서 감지 이상에도 전체 시스템은 제대로 동작할 수 있다. 최종 계산하는 공간의 평균 온도, 습도와 조도 정보를 Controller에게 보내고 다음단계의 처리를 진행할 것이다.

그림 112는 실내 구동체를 제어하는 과정이다. 센서 웹 시스템에서 지정 공간의 환경 정보를 정보를 받아서 쾌적 지수 PMV및 ET 지수를 계산하고 지수가 의미하는 상태 정보를 찾는다. 그리고 표 23이 제시하는 퍼지 룰을 통해 결합 상태를 얻고 이에 따라서 에어컨과 보일러 등 온도 조정 디바이스를 제어하며, 조도 정보에 따라서 조명기구를 제어하며, 실내 습도상태에 따라서 가습기를 구동한다. 보통 인간의 실내 쾌적 환경은 조도 1000lx를 유지하고, 습도는 50%~60% 범위를 유지한다.

2.1.5 지능 제어 라우팅 방안

표 24 라우팅 테이블

App Server Address	Object ID (Sensor, Actuator)	Object IP (Sensor, Actuator)	Space ID (Room ID)	Object Type	Middleware IP	Service Provider Address (Sensor web, Actuator web)
http://220.149.42.19/AppServiceProviderService/	DS00012	192.168.1.23	8	Actuator	220.149.42.19	http://117.17.102.28/ActuatorwebProviderService/
	SD00023	192.168.1.12	8	Sensor	220.149.42.115	http://117.17.102.7/SensorwebProviderService/
	DS00015	192.168.1.22	8	Actuator	220.149.42.19	http://117.17.102.154/ActuatorwebProviderService/
	DS00010	192.168.1.24	7	Actuator	220.149.42.43	http://117.17.102.44/ActuatorwebProviderService/
	SD00010	192.168.1.13	7	Sensor	220.149.42.51	http://117.17.102.7/SensorwebProviderService/
	SD00011	192.168.1.15	7	Sensor	220.149.42.66	http://117.17.102.197/SensorwebProviderService/
	...	...	...	...	...	...

이론적으로 응용서버입장에서 센서 웹 시스템이 제공하는 센서 정보를 통해 구동체 웹 시스템이 등록하는 구동체에 대해 제어 프로세스를 제대로 수행하려면 표 24와 같은 라우팅 테이블을 필요하다. 오브젝트 ID는 센서노드 및 구동체 노드의 유일 식별자이고, 오브젝트 IP는 센서노드 및 구동체 노드의 네트워크 연결하는 IP주소이다. Space ID는 Group개념과 같고 이를 통해 어떤 센서의 정보에 의해 어떤 구동체의 동작 상태를 결정하기 위한 것이다. 오브젝트 Type는 오브젝트의 실제 유형을 명시하고, Middleware IP는 해당 오브젝트와 연결한 미들웨어의 IP 주소이다.

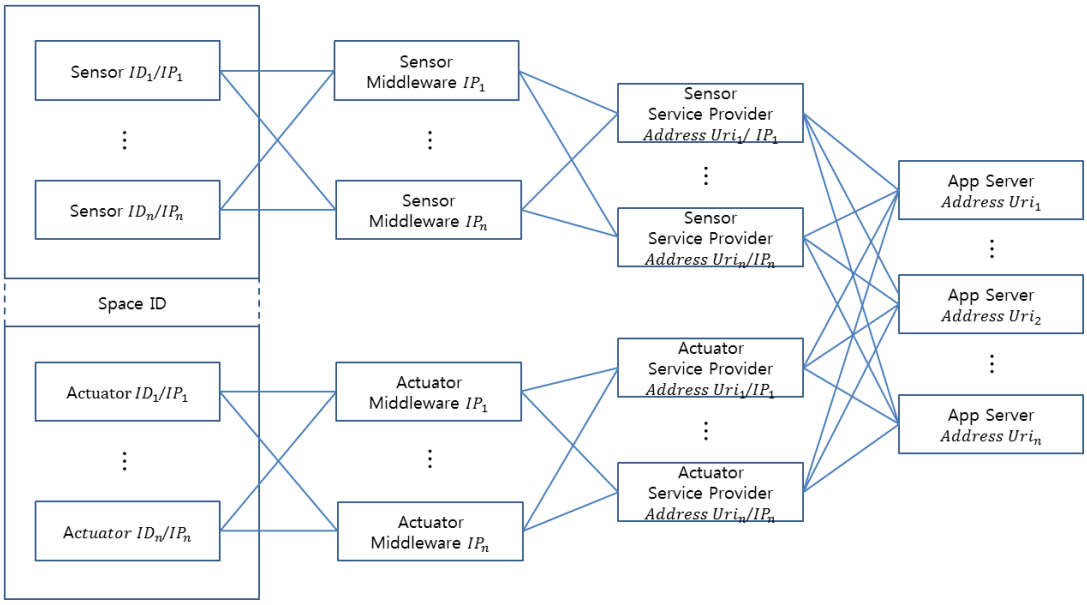

그림 113 스마트 제어 네트워크 구성

스마트 제어를 제대로 수행하기 위해 센서, 구동체, 미들웨어, 서비스 공급자 및 응용서버로 구성하는 네트워크 환경을 구축해야 된다. 메시지를 양방향을 송수신기능을 구현하기 위해 센서와 구동체는 ID, IP 정보가 필요하고, 미들웨어는 IP 정보가 필요하며, 서비스 공급자는 Uri, IP 정보가 필요하다. 응용서버는 Uri 주소 정보가 필요하다. 그림 113는 스마트 제어 네트워크 구성이다.

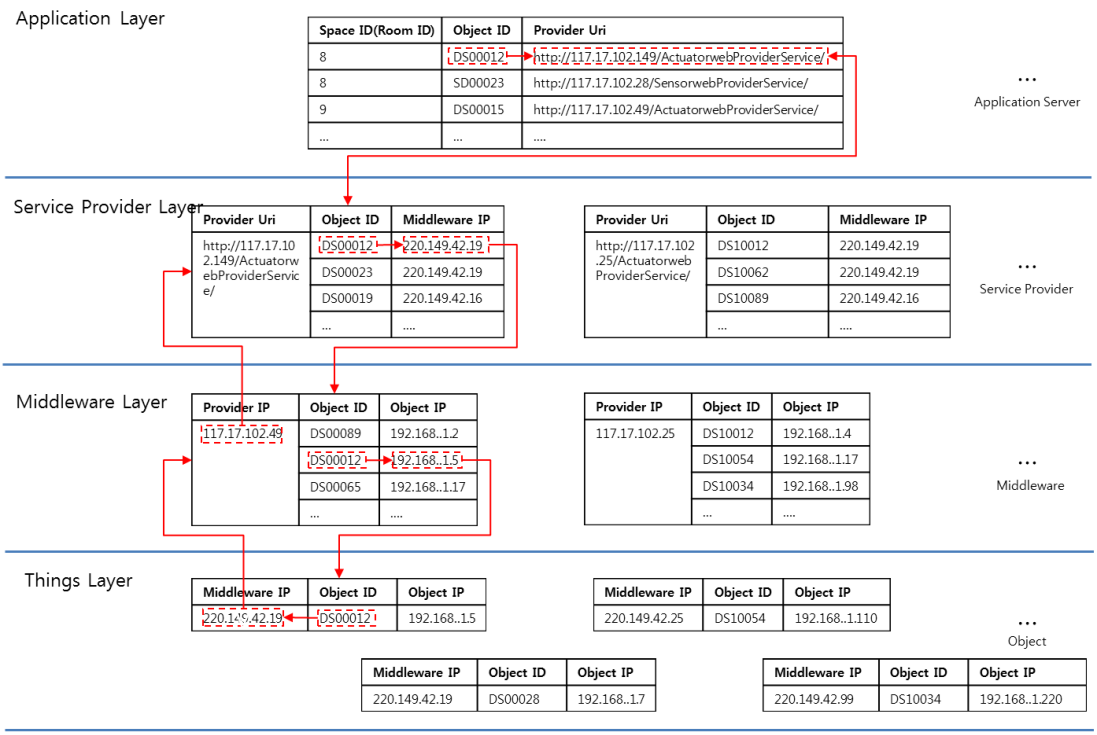

그림 114 오브젝트 라우팅 과정

그림 114는 응용서버에서 오브젝트(센서, 구동체)까지 메시지 전송 라우팅 과정이다. 그림을 대로 응용서버가 DS00012를 메시지를 전송한다고 가정한다. 처음에 응용서버의 매핑 테이블에서 공급서비스의 주소를 찾아서 전송한다. 서비스 공급자가 메시지를 수신하여 서비스 공급자의 매핑 테이블에 의해 미들웨어 주소를 찾아서 미들웨어에게 메시지를 전달한다. 마지막 미들웨어가 메시지를 수신하여 미들웨어의 매핑 테이블에 의해 오브젝트의 IP주소를 찾아내고 전송한다. 역방향 메시지 전송을 같은 경우에 오브젝트가 자기 저장된 미들웨어 IP를 통해 메시지를 전송하여, 미들웨어는 갖고 있는 공급서비스의 IP를 통해 메시지를 전송하고 서비스 공급자가 이 메시지를 수신하여 처리하여 DB에서 저장한다. 어떤 시점에 응용서버가 서비스 공급자에게 데이터를 요청하면 DB에서 데이터를 읽고 나서 응답한다.

센서는 네트워크를 연결하고 서버를 배치하면 자동적인 센싱 데이터를 생성하고 서버에게 전송한다. 센서웹 공급자가 수신되는 센싱 데이터를 가공하여 DB에서 저장한다. 그래서 응용서버입장에서 어떤 센서 노드의 센싱 데이터를 받으려면 단지 센서 ID와 서비스 공급자의 주소를 통해 센서웹 공급자에 저장된 센싱 데이터를 요청하여 받을 수 있다. 그런데 구동체 제어 과정은 조금 복잡하다.

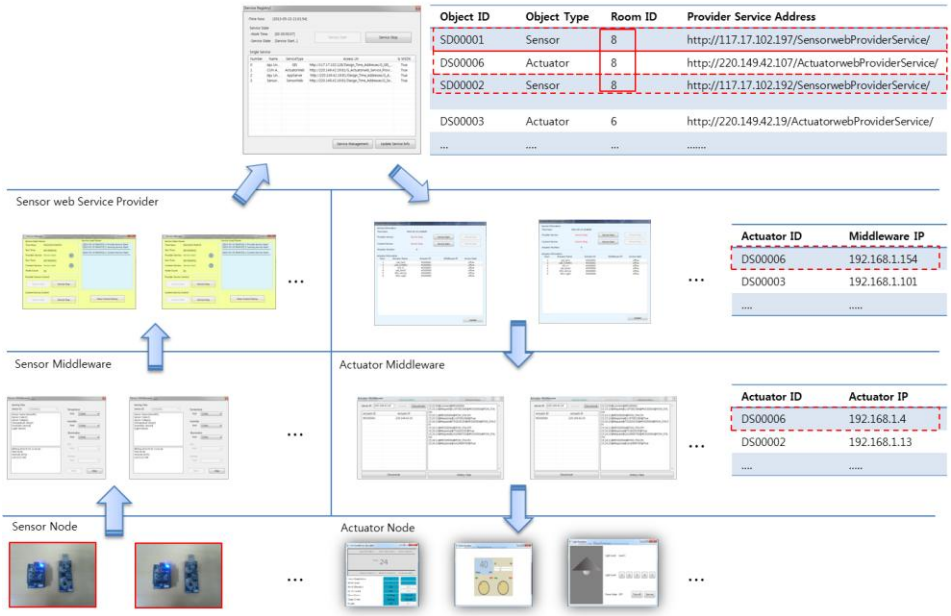

그림 115 센서와 구동체 연결라우팅 구성

구동체 제어 같은 경우에는 응용서버가 특정한 구동체에게 제어 명령을 전송한다. 그런데 실제 시스템구조에서 구동체웹 공급자가 서버 역할이고 구동체가 클라이언트 역할을 수행한다. 서버로부터 클라이언트에게 데이터를 전송하려면 네트워크 라우팅정보가 있어야 된다. 그림 115는 센서와 구동체 연결라우팅 구성이다. 응용서버의 스마트 제어 모듈은 지역정보(Room ID)를 통해 센서하고 구동체를 연결해 주는 것이다. 한편 이는 센서 노드와 직접적으로 연결하지 않고 센서웹 공급자의 DB를 통해 원하는 정보를 요청한다. 한편 구동체 ID를 통해 각 매핑 테이블에서 목적 구동체에게 전송한다. 아래 설계된 라우팅과정은 센싱 데이터 라우팅과 명령 라우팅으로 나누고 기술한다.

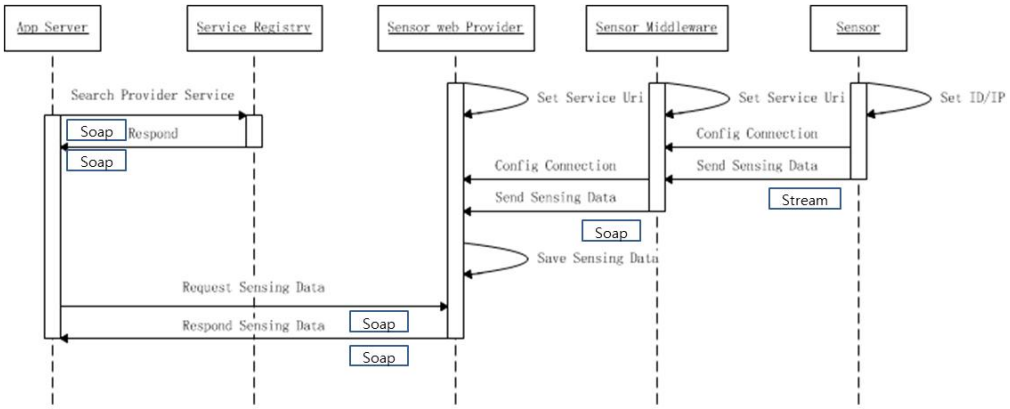

그림 116 센싱 데이터 라우팅 과정

그림 116은 본 논문에서 제시하는 센싱 데이터 라우팅 과정이다. 처음에 응용서버가 서비스 등록자에게서 센서웹 공급자의 주소를 검색한다. 센서에 ID하고 IP를 설치하고, 센서 미들웨어가 서비스 공급자의 주소를 설치하고, 센서웹 공급자가 자기의 주소를 설치한다. 연결 환경을 구축하여 센서가 수신된 데이터를 Steam방식으로 데이터를 전송하고 센서 미들웨어가 Soap를 사용하여 센서웹 공급자의 API를 통해 센싱 데이터를 전송하여 DB에 저장한다. 응용서버가 센서웹 공급자의 API를 통해 센싱 데이터를 요청하여 응답한다.

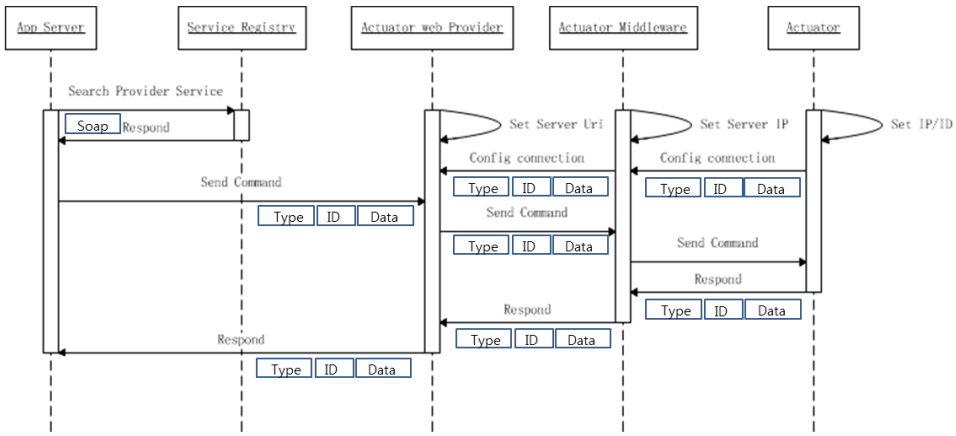

그림 117 구동체 명령 라우팅 과정

그림 117은 본 논문에서 제시하는 구동체 명령 라우팅 과정이다. 처음에 응

용서버가 서비스 등록자에게 센서웹 공급자의 주소를 검색한다. 구동체가 ID하고 IP를 설치하고, 구동체 미들웨어가 서비스 공급자의 주소를 설치하고, 구동체웹 공급자가 자기의 주소를 설치한다. 응용서버가 명령 메시지를 서비스 공급자에게 전송한다. 명령 메시지가 목적 구동체의 ID정보를 포함하여 ID를 통해 서비스 공급자 및 미들웨어의 매핑 테이블을 이용하여 메시지를 전송하여 응답 메시지를 역방향 전송한다.

2.2 응용서버 기반 실내 IOT 시스템

2.2.1 응용서버 기반 실내 IOT 시스템의 인터페이스 프로토콜

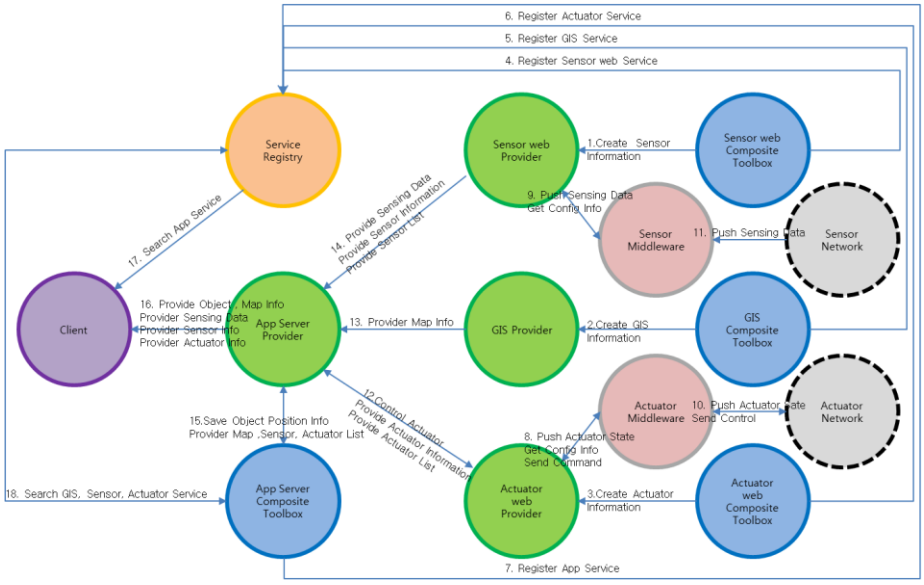

그림 118 응용서버 기반 IOT 시스템 구조

그림 118은 응용서버 기반 IOT 시스템의 모듈들 간의 협력 과정이다. 본 시스템은 클라이언트, 서비스 등록자, 응용서버, 응용서버 저작도구, 센서웹 공급자, GIS 공급자, 구동체웹 공급자, 센서 미들웨어, 구동체 미들웨어, 센서웹 저작도구, GIS 서비스 저작도구, 구동체웹 저작도구, 센서 네트워크, 구동체 네트워크라는 모듈들로 구성된다. 그리고 각 모듈간의 협력과정은 앞의 설계 단계에서 이미 상세히 언급하였고 다음으로 실제 구현할 때의 각 모듈간의 API의 참조에 대해 살

펴본다. 본 분산 처리 시스템은 SOAP을 기반으로 구현함으로써 API는 Microsoft .NET 환경의 WCF 기술을 사용하여 구현된다.

표 25 응용서버 기반 IOT 시스템의 API

Module Name	API Name	Description		Applied At
		Type	Operation	
Service Registry	Service Publish Interface Protocol	Service Information	Write	4, 5, 6, 7
	Service Search Interface Protocol	Service Information	Search, Read	17, 18
GIS Service Provider	GIS Provider Interface Protocol	Map Information	Read	13
	GIS Content Interface Protocol	Map Information	Read, Write	2
Sensor web Service Provider	Sensor web Service Provider Interface Protocol	Sensor List	Read, Search	14
		Sensor Information	Read	14
		Sensing Data	Read	14
	Sensor web Service Content Interface Protocol	Sensor List	Read, Search	1
		Sensor Information	Write	1
Actuator web Service Provider	Actuator web Service Provider Interface Protocol	Actuator List	Read, Search	12
		Actuator Information	Read	12
		Control	Write	12
	Actuator web Service Content Interface Protocol	Actuator List	Read, Search	3
		Actuator Information	Write	3
App Server	App Service Provider Interface Protocol	Map Information	Read	16
		Sensor Information	Read	16
		Sensing Data	Read	16
		Actuator Information	Read	16
		Object Information	Read	16
	App Service Content Interface Protocol	Map Information	Read	15
		Sensor Information	Read	15
		Sensor List	Read, Search	15
		Actuator Information	Read	15
		Actuator List	Read, Search	15
		Object Information	Read, Write	15

표 25는 응용서버 기반 IOT 시스템의 각 모듈은 제공하는 API이다. 서비스 등록자는 Service Publish Interface Protocol 및 Service Search Interface Protocol을 포함한다. Service Publish Interface Protocol은 서비스 정보의 등록 역할을 제공하며, Service Search Interface Protocol은 서비스 정보의 검색역할을 제공한다. GIS 공급자는 GIS Service Provider Interface Protocol 및 GIS Content Interface Protocol을 포함한다. GIS Service Provider Interface Protocol은 지도 데이터, 빌딩 정보, 층 정보, 방 정보를 제공하며, GIS Content Interface Protocol은 지도 데이터, 빌딩 정보, 층 정보, 방 정보의 생성 및 관리를 제공한다. 센서웹 공급자는 Sensor Web Provider Interface Protocol 및 Sensor Web Service Content Interface Protocol을 포함한다. Sensor Web

Provider Interface Protocol은 센서 정보, 센서 검색, 센싱 데이터를 공급하며, Sensor Web Service Content Interface Protocol 센서 정보의 생성 및 관리를 제공한다. 구동체웹 공급자는 Actuator Web Provider Interface Protocol 및 Actuator Web Content Interface Protocol을 포함한다. Actuator Web Provider Interface Protocol은 구동체 정보, 구동체 검색, 구동체 상태, 구동체 제어를 제공하며, Actuator Web Content Interface Protocol 구동체 정보의 생성 및 관리를 제공한다. 응용서버는 App Server Provider Interface Protocol 및 App Service Content Interface Protocol을 포함한다. App Server Provider Interface Protocol은 지도관련 정보(GIS Service를 제공하는 서비스 중계), 센서관련 정보, 구동체 관련 정보 오브젝트의 정보를 제공하며, App Service Content Interface Protocol은 지도 서비스, 센서관련 정보, 구동체관련 정보 공급, 오브젝트정보의 생성 및 관리를 제공한다.

2.2.2 환경 설정 단계

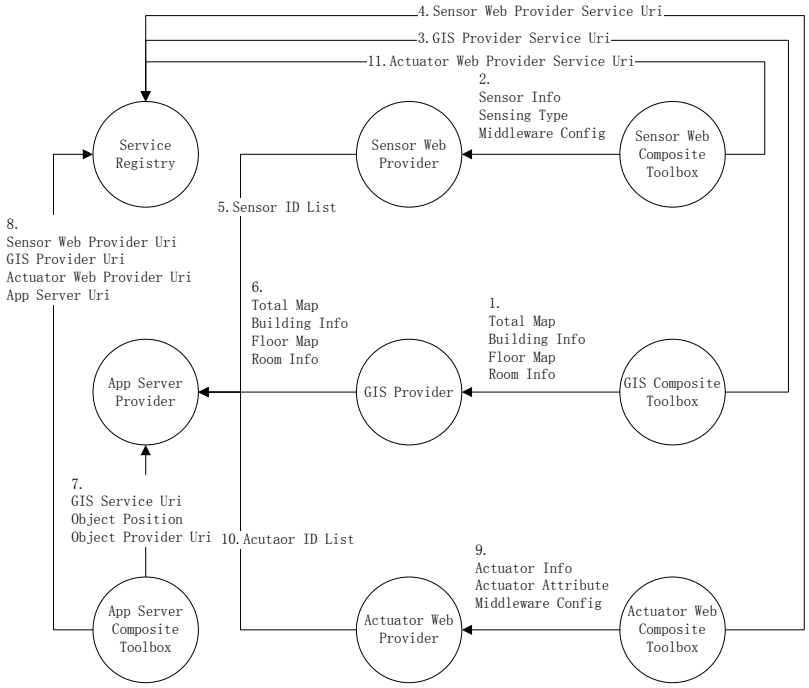

그림 119 응용서버 기반 실내 IOT 시스템 환경 설정 단계

그림 119는 응용 기반 실내 IOT 시스템의 환경 설정 단계이다. 통합 서비스의 배치단계는 서비스의 필수 콘텐츠를 제대로 클라이언트에게 공급하고 검색서비스를 제공하기 위해 생성한다.

Total Map은 실외 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Building Info는 빌딩의 이름, 위치 정보, 층 이름을 포함한다. Floor Map은 실내 지도의 지도의 이미지 데이터, 지도 사이즈, 퍼센트 이미지와 미니 지도를 포함한다. Room Info는 방의 이름, 방 위치를 포함한다. Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. 센서 Type는 센서의 센싱 타입 정보를 의미한다. Middleware Config는 센서 미들웨어의 ID, IP주소과 접속권한 정보를 포함한다. Service Info는 각각 서비스 공급자의 서비스 이름, 검색 키워드, 접속 주소, 서비스 유형을 포함한다. 센서 List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 센서 ID들을 의미한다. GIS Service Uri는 GIS 공급자의 접속주소를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. Actuator List는 서비스 공급자가 사용자의 검색 조건에 따라서 응답하는 구동체 ID들을 의미한다. Actuator Position는 구동체의 지도상의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다. 센서웹 공급자 Uri는 센서웹 공급자의 접속주소를 의미한다. App Server Service Provider Uri는 응용서버의 접속주소를 의미한다.

GIS 서비스 저작도구는 지도 관련 데이터를 외부에서 GIS Content API를 통해 GIS 공급자의 DB에 저장한다. 센서웹 저작도구는 각 센서노드의 정보를 센서웹 Content API를 통해 생성하여 DB에 저장하며 미들웨어의 배치 정보를 생성한다. GIS 서비스 저작도구가 GIS 공급 서비스의 정보를 서비스 등록자에게 등록한다. 센서웹 저작도구가 센서웹 공급자 서비스 정보를 서비스 등록자에게 등록한다. 센서웹 공급자가 제공하는 센서 리스트 정보와 GIS 공급자가 제공하는 지도 데이터를 응용서버를 통해 응용서버 저작도구에게 제공한다. 응용서버 저작도구가 각 센서 노드와 지도의 위치정보를 생성하여 저장하고 응용서버의 서비스 정보를 등록한다. 구동체웹 저작도구는 각 구동체 노드 정보를 Actuator Web Content API를 통해 생성하고 DB에 저장하여 미들웨어의 배치 정보를 생성한다.

구동체웹 공급자가 제공하는 센서 리스트 정보를 응용서버를 통해 응용서버 저작도구에 제공한다. 구동체웹 저작도구가 구동체웹 공급자 서비스 정보를 서비스 등록자에게 등록한다.

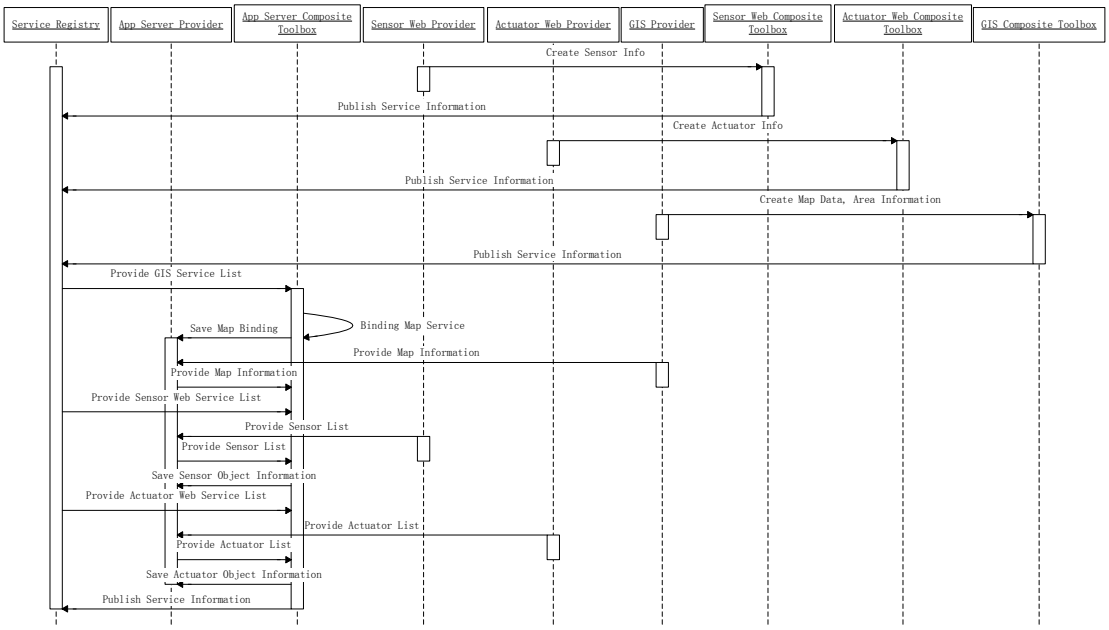

그림 120 응용서버 기반 실내 IOT 시스템 환경 설정 단계 시퀀스

그림 120은 응용서버 기반 실내 IOT 시스템의 환경 설정 단계 시퀀스 다이어그램이다. 위의 과정은 앞에서 기술했던 센서웹 시스템과 구동체웹 시스템의 환경 설정 단계를 통합한 것이다. 그래서 서비스 등록자, 응용서버 서비스 공급자, 응용서버 저작도구, 센서웹 공급자, 구동체웹 공급자, GIS 공급자, 센서웹 저작도구, 구동체웹 저작도구와 GIS 서비스 저작도구로 구성된다.

처음에 센서웹 저작도구, 구동체웹 저작도구와 GIS 서비스 저작도구가 센서 오브젝트 정보, 구동체 오브젝트 정보 및 지도 서비스 콘텐츠를 생성하여 센서웹 공급자, 구동체웹 공급자 와 GIS 공급자에 저장한다. 다음에 센서웹 공급자, 구동체웹 공급자 와 GIS 공급자의 서비스 정보를 서비스 등록자에 등록하고 응용서버 저작도구는 서비스 등록자에 등록된 GIS 서비스 정보를 검색하여 Map Service를

바인딩하고 응용서버 서비스 공급자에 저장한다. 응용서버 저작도구는 바인딩된 지도 서비스 정보에 따라서 응용서버를 통해 지도 데이터를 요청하여 가시화를 제공한다. 서비스 등록자에서 센서웹 시스템 및 구동체웹 시스템 정보를 요청하여 관리자가 각 공급 서비스에 제공하는 오브젝트(센서, 구동체)를 선택하여 지도상에 오브젝트 콘텐츠를 생성하여 응용서버 서비스 공급자에 저장한다. 콘텐츠를 생성하고 나서 응용서버의 공급 서비스 정보를 서비스 등록자에 저장한다.

2.2.3 검색 단계

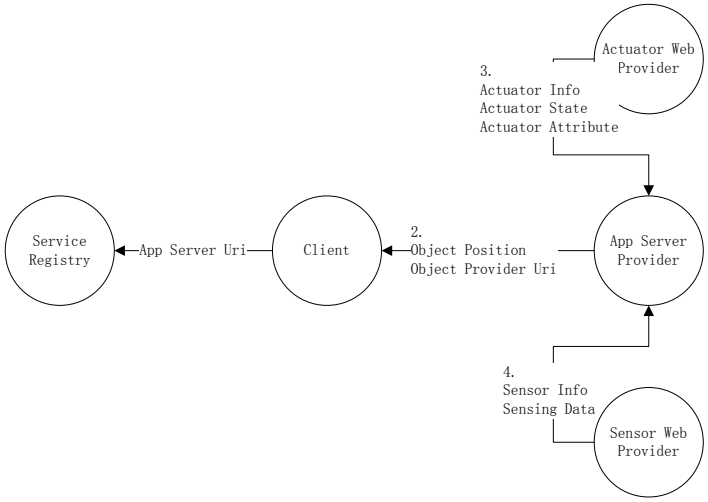

그림 121 응용서버 기반 실내 IOT 시스템의 검색 단계

그림 121은 응용서버 기반 실내 IOT 시스템의 검색 단계이다. 통합 서비스의 검색 단계는 클라이언트가 서비스 등록자에서 쉽게 응용서버, 센서 노드 및 구동체 노드를 찾을 수 있도록 한다.

Sensor Info는 센서 노드의 이름, 센서 ID를 포함한다. 센싱 데이터는 센서의 실시간 센싱 데이터를 의미한다. Sensor Position는 센서의 지도상의 위치 정보를 의미한다. 센서웹 공급자 Uri는 센서웹 공급자의 접속주소를 의미한다. Actuator Info는 구동체 노드의 이름, 구동체 ID를 포함한다. Actuator State는 구동체의 동작상태 정보를 의미한다. Actuator Attribute는 구동체의 작업 속성을 의미한다.

Actuator Position는 구동체가 지도상의 위치 정보를 의미한다. Actuator Provider Uri는 구동체웹 공급자의 접속주소를 의미한다. App Server Uri는 응용서버의 접속주소를 의미한다.

클라이언트는 사용자가 입력하는 서비스검색 키워드를 통해 서비스 등록자에게 서비스 리스트를 요청하고 응용서버에 접속하여 센서의 위치 정보를 요청한다. 그리고 구동체 ID과 센서 ID를 통해 응용서버에게 Actuator 및 센서 정보를 요청한다.

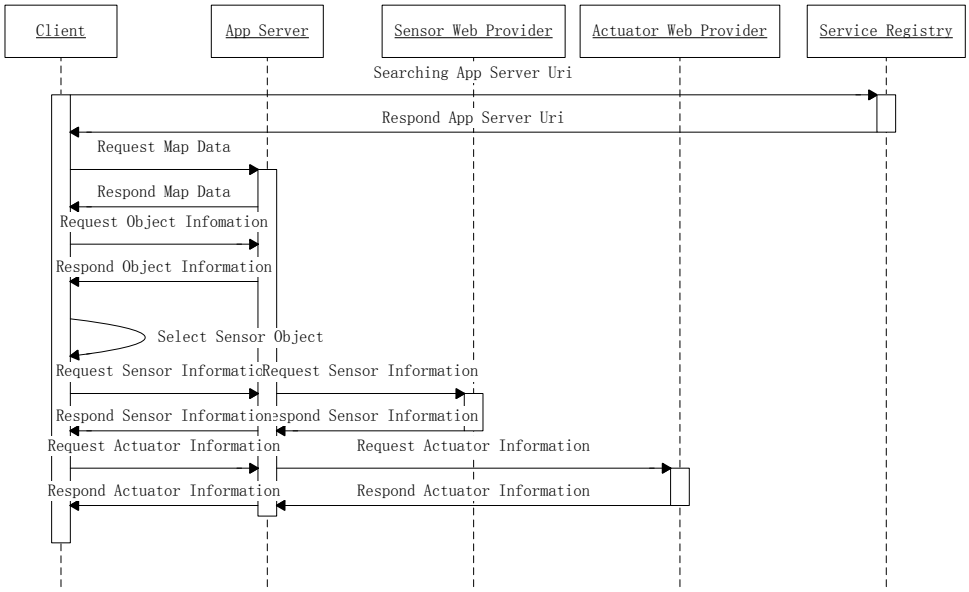

그림 122 응용서버 기반 실내 IOT 시스템의 검색 단계 시퀀스

그림 122는 응용서버 기반 실내 IOT 시스템의 검색 단계 시퀀스 다이어그램이다. 위의 과정은 클라이언트, 응용서버, 센서웹 공급자, 구동체웹 공급자와 서비스 등록자로 구성된다.

위의 과정은 처음에 클라이언트가 서비스 등록자에서 응용서버의 서비스 정보를 검색하여 접속한다. 다음에 응용서버가 제공하는 지도 서비스 정보에 따라서 지도 서비스 콘텐츠를 요청 받아 가시화를 제공하며 오브젝트(센서, 구동체)의

위치정보를 요청 받아서 출력한다. 사용자가 원하는 센서나 구동체 노드를 선택하여 응용서버를 통해 대응의 서비스 공급자에게 세부정보를 요청하고 가시화한다.

그리고 응용서버 기반 실내 IOT 시스템의 지능 제어 단계하고 제어 라우팅 방안은 2.1.4절과 2.1.5절에서 제시하는 방안을 같다.

2.3 실내 IOT 시스템 세부설계

2.3.1 서비스 공급자 기반 IOT 시스템 응용서버

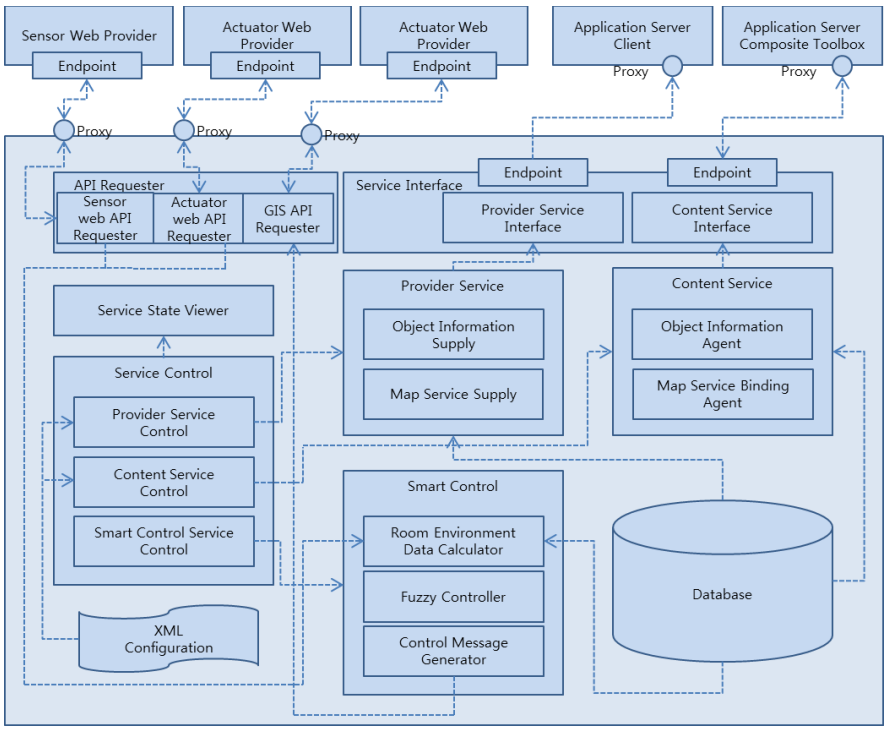

그림 123 서비스 공급자 기반 IOT 시스템 응용서버 세부구조

그림 123은 응용서버의 세부구조이다. 응용서버는 지능제어 시스템의 핵심 모듈이다. 이는 센서웹 시스템과 구동체웹 시스템간의 다리이다. 자기가 갖고 있는 지역 정보에 의해 센서웹 시스템에서 특정 구역의 환경 정보를 요청하고 그 구역의 현 환경 상태에 따라서 자동적으로 구동체웹 시스템을 통해 구동체를 제

어한다. 응용서버는 주로 Provider Service, Content Service 및 Smart Control로 구성된다. Provider Service는 클라이언트에게 오브젝트 정보(위치 정보, 타입, 공급 서비스 주소) 및 지도 서비스를 제공한다. Content Service는 응용서버 저작도구로서 오브젝트 정보관리 및 지도 서비스 바인딩 관리 서비스를 제공한다. Smart Control은 방을 대상으로 환경 데이터를 센서웹 공급자로부터 요청 받아서 환경 상태를 판단하여 Fuzzy Controller가 제어 콘텐츠를 생성한다. Control Message Generator가 제어 콘텐츠에 따라서 Control Message를 생성하고 구동체웹 공급자가 제공하는 제어 API를 통해 원격에 있는 구동체를 제어한다. XML Configuration은 WCF Service의 동작 환경을 제공한다. Database는 Map Service 바인딩 정보와 오브젝트 정보를 저장한다.

2.3.2 서비스 공급자 기반 IOT 시스템 응용서버 저작도구

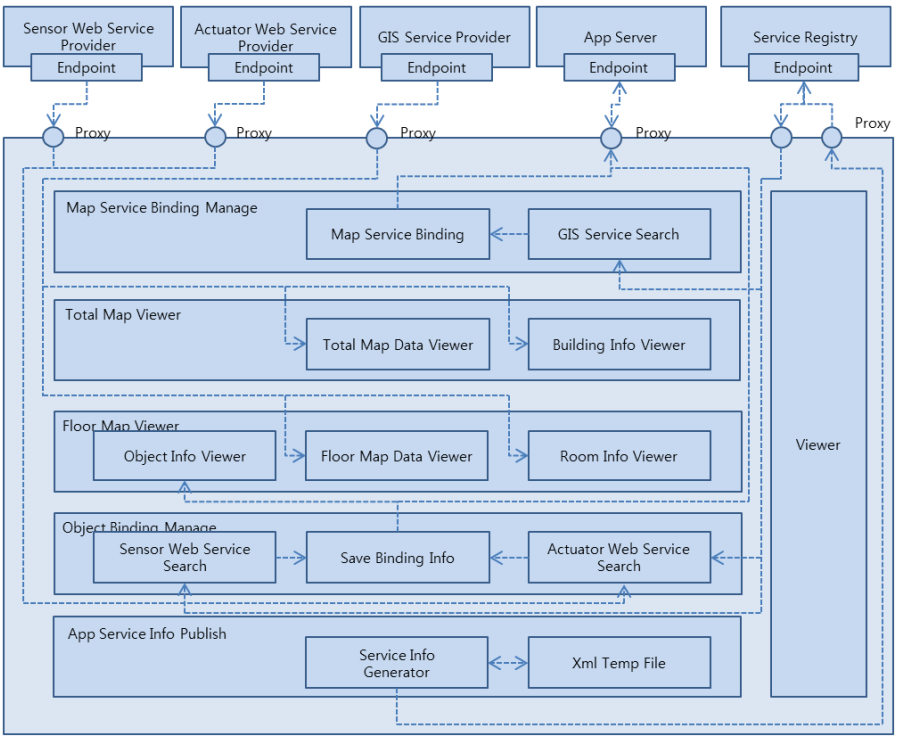

그림 124 서비스 공급자 기반 IOT 시스템 응용서버 저작도구 세부구조

그림 124는 응용서버 저작도구의 세부구조이다. 이 모듈은 지도 서비스 바인딩, 오브젝트 위치 바인딩을 관리하기 위해 가시화를 제공한다. 처음에 GIS Service Search가 서비스 등록자에서 관리자가 원하는 지도 서비스를 검색하여 바인딩 한다. 다음에 Total Map Viewer는 GIS 공급자에서 전체지도 데이터 및 빌딩 정보를 요청한다. 관리자가 전체지도 뷰어에서 특정 빌딩의 층을 선택하여 Floor Map Viewer가 층 지도 데이터, 방 정보 및 오브젝트를 도시한다. 오브젝트 Binding Manage가 센서 노드 Binding 및 Actuator Node Binding 역할을 수행한다. App Service Info Publish는 Service Info Generator를 통해 Service정보(검색 키워드, 이름, 주소 등)를 생성하여 XML 임시파일에 저장하여 서비스 등록자에게 등록한다.

2.3.3 서비스 공급자 기반 IOT 시스템 클라이언트

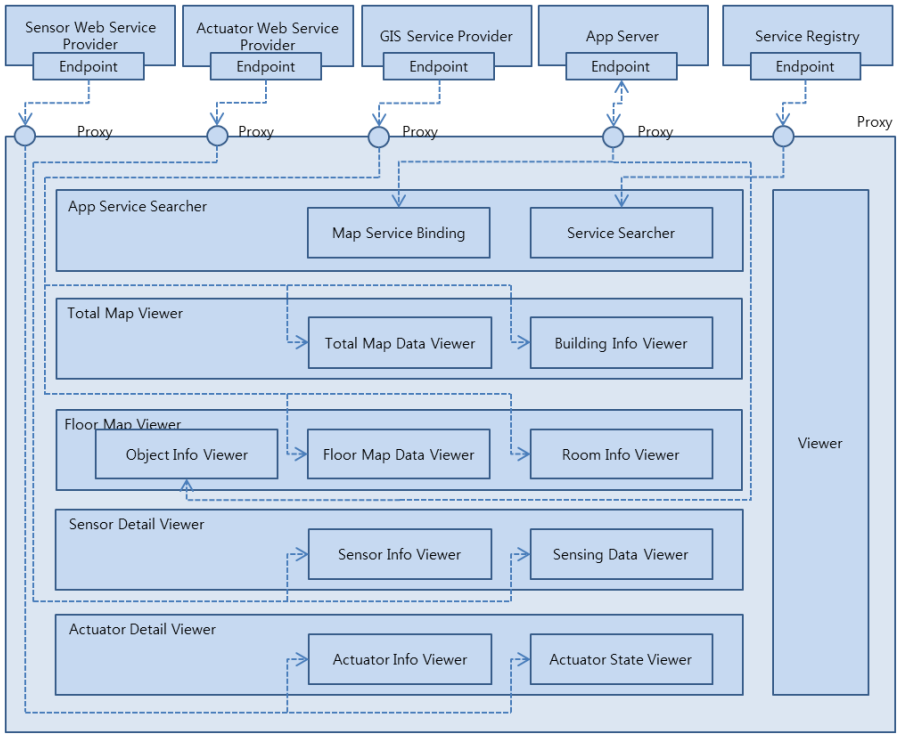

그림 125 서비스 공급자 기반 IOT 시스템 클라이언트 세부구조

그림 125는 응용 클라이언트의 세부구조이다. 클라이언트가 단순한 가시화를 제공한다. 처음에 App Service Searcher의 Service Searcher가 서비스 등록자에게 응용서버를 검색하여 지정한 서비스를 접속하고 Map Service Binding이 응용서버에서 Map 서비스 정보를 요청하여 바인딩 한다. 다음에 Total Map Viewer는 GIS 공급자에서 전체지도 데이터 및 빌딩 정보를 요청한다. 관리자가 전체지도 뷰어에서 특정 빌딩의 층을 선택하여 Floor Map Viewer가 층 지도 데이터, 방 정보 및 오브젝트를 도시한다. 사용자가 센서 오브젝트를 선택하면 Sensor Info Viewer가 센서 이름, 아이디, Sensing Type 등 속성을 출력하며 센싱 데이터 Viewer가 실시간 센싱데이터를 출력한다. 구동체 오브젝트를 선택하면 Actuator Info Viewer가 구동체 이름, 아이디, 구동체 타입, 모델 정보 등 속성을 출력한다.

2.3.4 응용서버 기반 IOT 시스템 응용서버

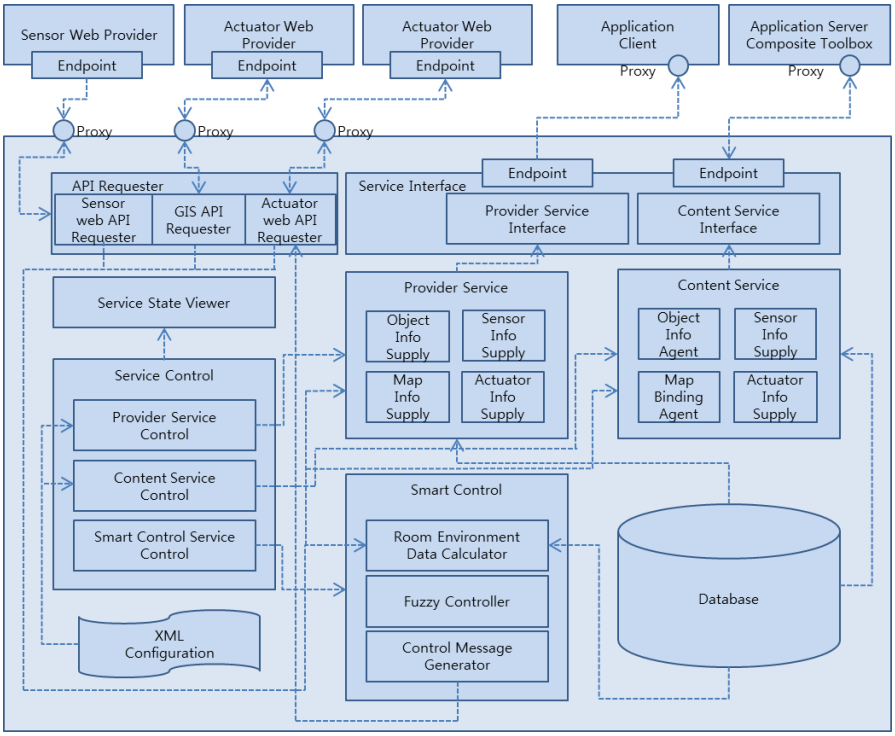

그림 126 응용서버 기반 IOT 시스템 응용서버 세부구조

그림 126은 응용서버 기반 IOT 시스템의 응용서버 세부구조이다. 본 논문의 III장 1절에서 제시하는 서비스 공급자 기반 IOT 시스템의 응용서버 세부구조와 비슷하지만 두 가지 차이점 있다. 하나는 공급서비스가 센서웹 공급자와 구동체웹 공급자가 제공하는 API들이 모여서 클라이언트에게 제공한다. 또 하나는 공급서비스가 센서웹 공급자와 구동체웹 공급자가 제공하는 API들이 모였어 응용서버 저작도구를 제공한다. 클라이언트와 응용서버 저작도구는 간접연결 방식으로 센서웹 공급자 및 구동체웹 공급자를 연동한다.

2.3.5 응용서버 기반 IOT 시스템 응용서버 저작도구

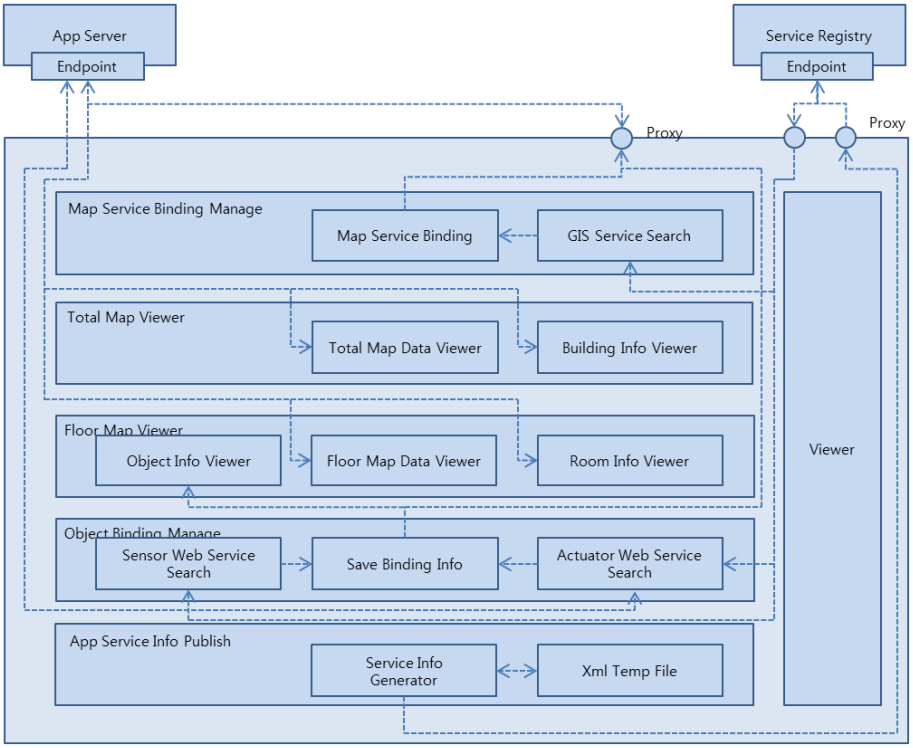

그림 127 응용서버 기반 IOT 시스템 응용서버 저작도구 세부구조

그림 127은 응용서버 기반 IOT 시스템의 응용서버 저작도구 세부구조이다. 2.3.2절에서 제시하는 서비스 공급자 기반 IOT 시스템의 응용서버 저작도구 세부구조와 같다. 단지 여기에서 응용서버 저작도구는 응용서버에서 센서와 구동체

정보를 직접적으로 요청할 수 있다.

2.3.6 응용서버 기반 IOT 시스템 클라이언트

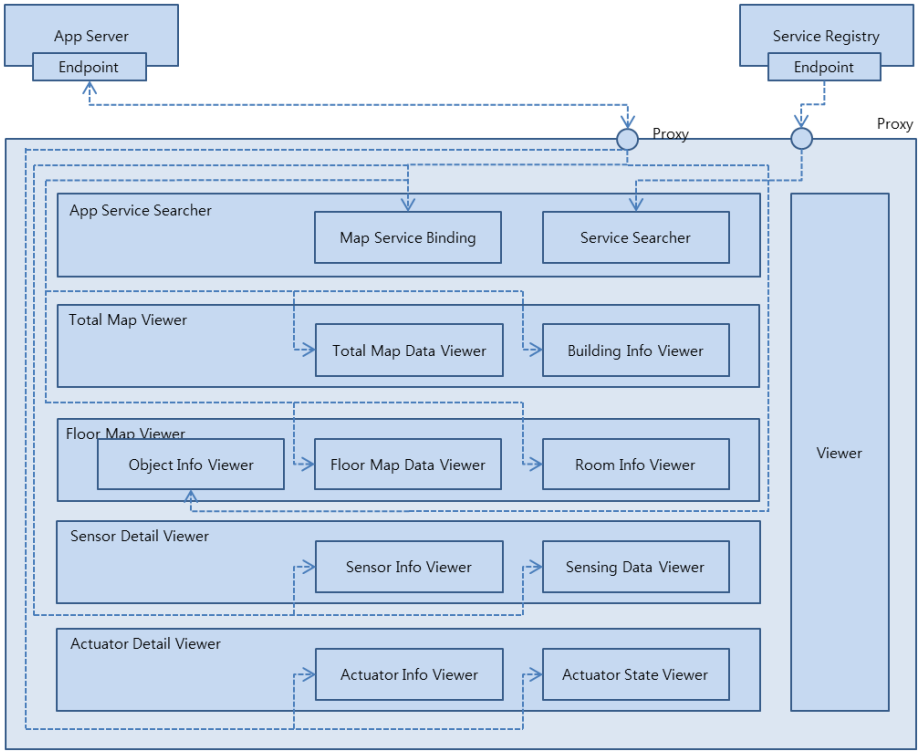

그림 128 응용서버 기반 IOT 시스템 클라이언트 세부구조

그림 128은 응용서버 기반 IOT 시스템의 클라이언트 세부구조이다. 2.3.3절에서 제시하는 서비스 공급자 기반 IOT 시스템의 클라이언트 세부구조와 같다. 단지 클라이언트가 다른 서비스 공급자에서 센서 및 구동체 정보를 요청을 대신에 응용서버에 요청하여 받을 수 있다.

2.3.7 서비스 등록자

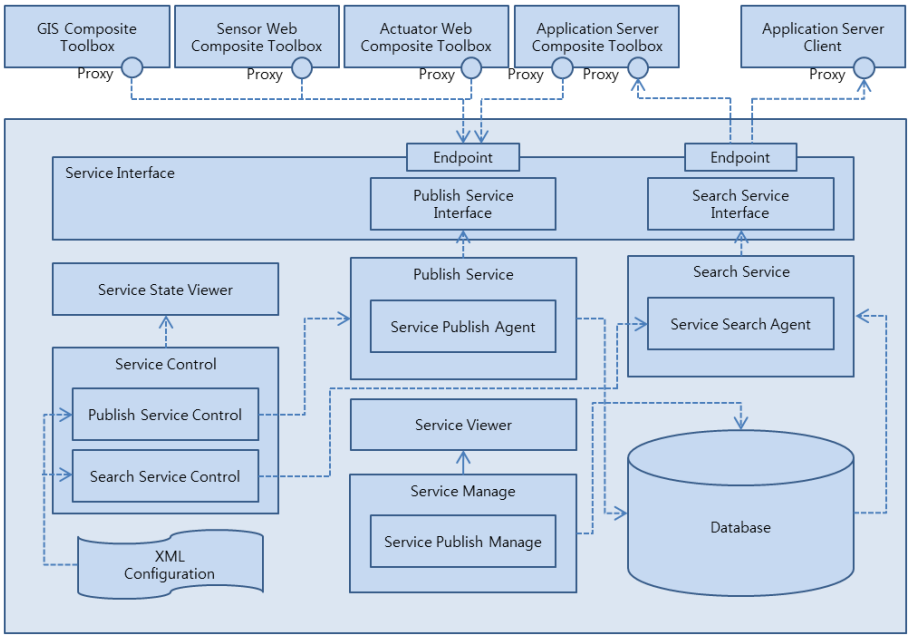

그림 129 서비스 등록자 내부고조

그림 129는 서비스 등록자의 세부구조이다. 서비스 등록자는 각 시스템의 저작도구에 서비스 정보를 등록하는 서비스를 제공한다. 그리고 각 응용서버 저작도구 및 클라이언트에게 검색 서비스를 제공한다. 서비스 등록자가 Publish Service 및 Search Service로 구성한다. Publish Service는 서비스 공급자의 정보(서비스 이름, 서비스 유형, 서비스 주소, 서비스 검색 키워드)를 등록하는 역할을 수행한다. Search Service는 다른 시스템에게 서비스 검색하는 역할을 수행한다. Service Control은 Publish Service, Search Service의 켜짐, 꺼짐 역할을 담당한다. Service Publish Manage가 등록된 서비스에 대한 공개여부를 제어하고, 서비스정보의 삭제를 관리한다. XML Configuration은 WCF Service의 동작 환경을 제공한다. Database는 등록된 서비스 정보를 저장하는 것이다.

3 실내 IOT 시스템 구현

3.1 응용서버

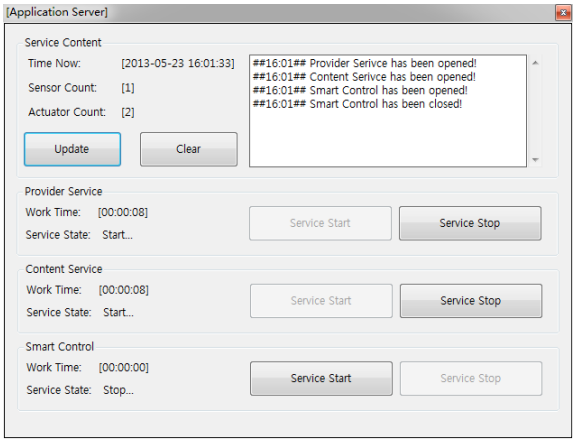

그림 130 응용서버 실행화면

그림 130은 응용서버 실행화면이다. 이는 응용서버 DB에 저장하는 오브젝트 상태를 도시하며, Provider Service, Content Service, Smart Control 서비스에 대해 제어한다. Time Now는 응용서버의 동작환경의 현재시간이다. 센서 Count는 DB에서 바인딩된 센서 오브젝트 개수이다. Actuator Count는 DB에서 바인딩된 구동체 오브젝트 개수이다. Work Time는 해당 서비스의 정상 동작하는 시간이다. Service State는 해당 서비스의 동작상태를 의미한다. Service Start 및 Service Stop 버튼은 해당 서비스의 꺼짐/켜짐을 제어하는 역할을 수행한다.

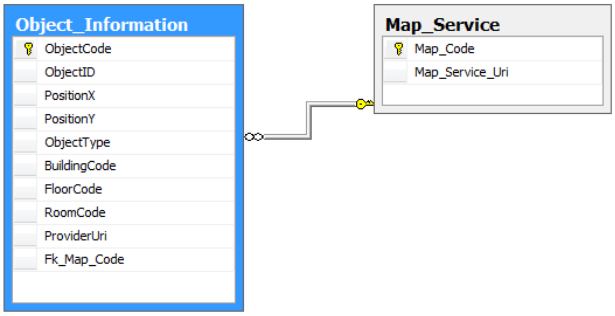

그림 131 응용서버 데이터베이스 관계도

그림 131은 응용서버의 데이터베이스이다. Object_Information은 오브젝트(센서, 구동체)의 위치정보, ID, 타입정보, 공급서비스 주소를 저장하는 테이블이다. Map_Service는 지도 서비스 바인딩 정보를 저장하는 테이블이다.

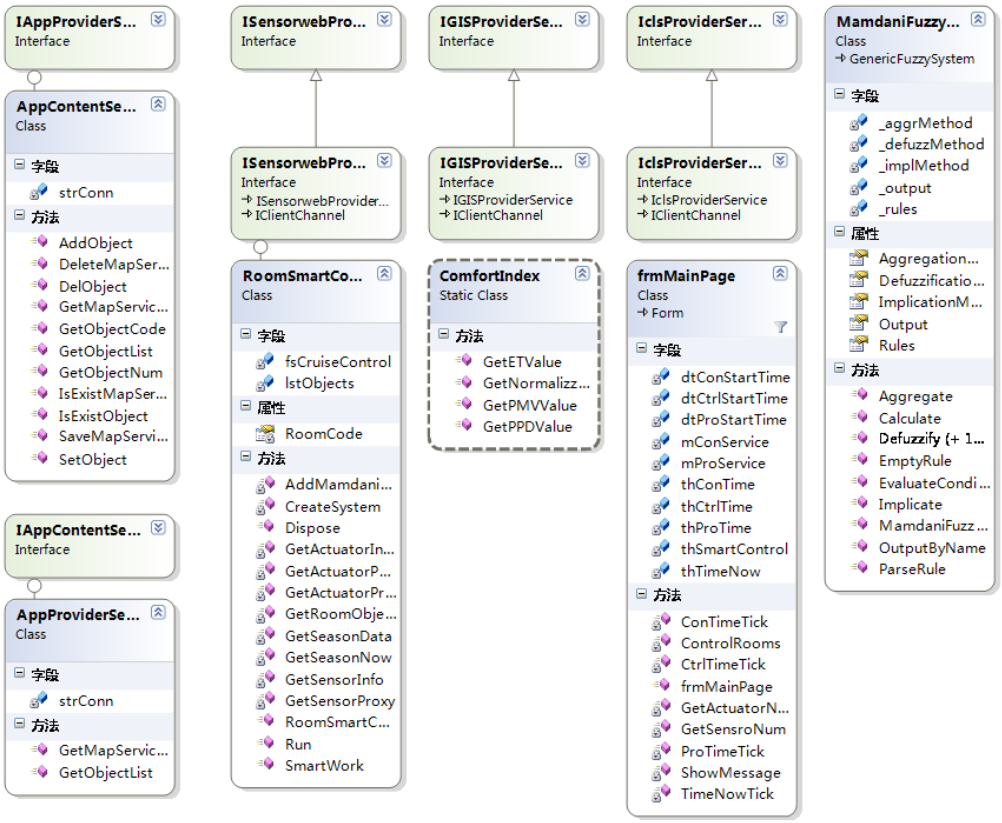

그림 132 응용서버 클래스 다이어그램

그림 132는 응용서버의 클래스 다이어그램이다. IAppContentService 및 IAppProviderService는 외부에 노출되는 인터페이스이다. AppContentService 및 AppProviderService는 인터페이스를 상속 받는 클래스이다. AppContentService는 지도 서비스, 오브젝트 바인딩 정보를 생성 및 관리하는 역할을 담당하며 AppProviderService는 지도 서비스, 오브젝트 바인딩 정보를 공급하는 역할을 담당한다. ISensorwebPorivder, IGISProvider, IclsProviderService는 센서 웹 공급 서비스, 지도 공급 서비스 및 구동체 웹 공급 서비스를 참조하기 위한 인터페이스이다. frmMainPage는 실제 화면에 표시되

는 Window form을 상속받는 클래스를 나타낸다. ComforIndex는 PMV, PPD, ET 등 쾌적 지수를 계산하는 방법을 제공한다. RoomSmartControl은 자동적으로 방의 실내 환경상태를 판단하여 구동체를 조정하는 역할을 담당한다. MamdaniFuzzySystem은 Mamdani 퍼지 개념을 사용하여 구현되는 퍼지 시스템이다.

3.2 응용서버 저작도구

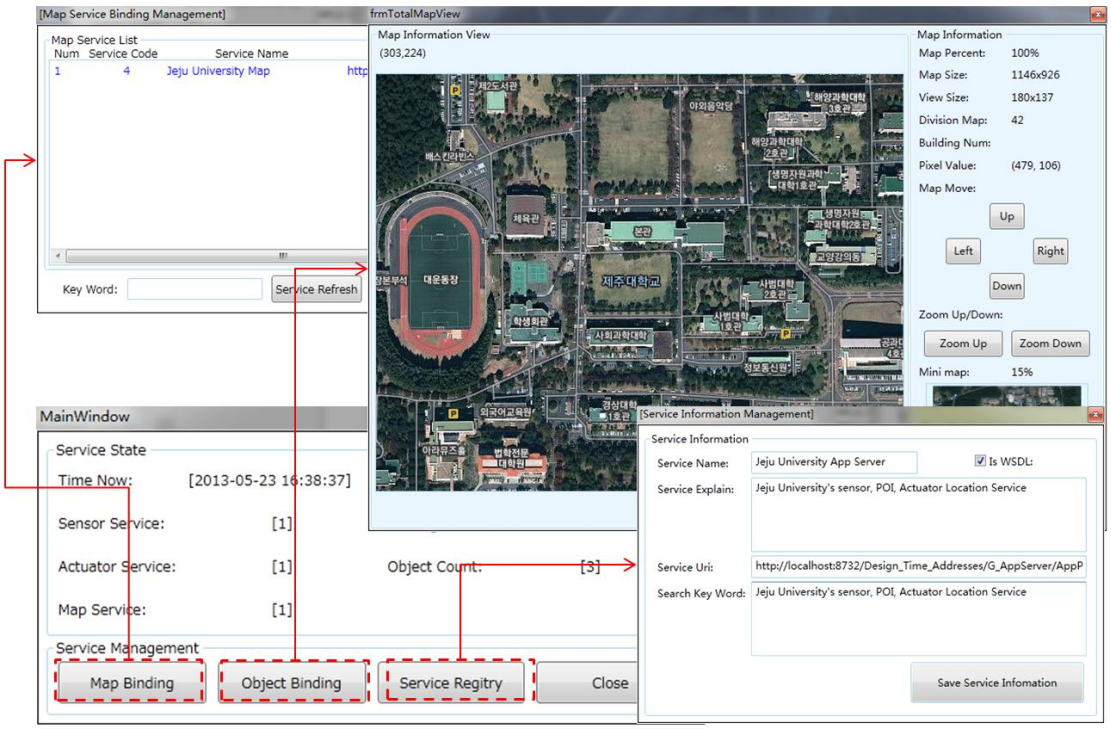

그림 133 응용서버 저작도구 실행화면

그림 133은 응용서버 저작도구의 실행화면이다. Map Binding 버튼을 통해 지도 서비스 바인딩 관리자로 넘어간다. Object Binding 버튼을 누르면 전체 지도 뷰어를 나타낸다. 서비스 등록자 버튼을 누르면 서비스 정보 관리가 나타난다.

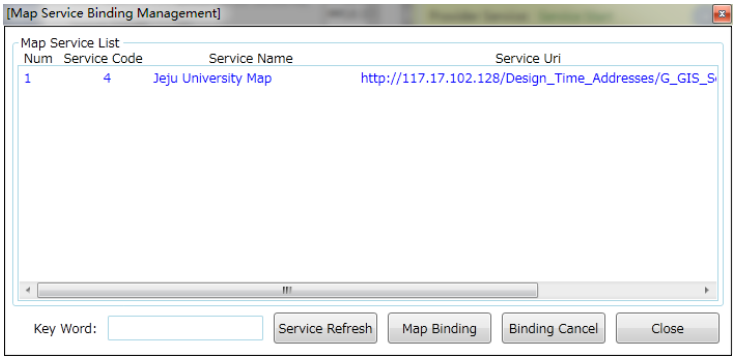

그림 134 지도 서비스 바인딩 관리 실행화면

그림 134는 지도 서비스 바인딩 관리의 실행화면이다. 이는 지도 서비스를 검색하고 지정 지도 서비스 바인딩 역할을 담당한다. Service Name는 GIS 서비스 이름이다. Service Uri는 GIS 서비스 접속주소를 의미한다. Key Word는 GIS 서비스 검색 키워드이다. Service Refresh 버튼은 GIS 서비스를 다시 검색한다. Map Binding 버튼은 GIS 서비스를 바인딩 역할을 수행한다. Binding Cancel는 이미 바인딩 되어있는 GIS 서비스 정보를 삭제한다.

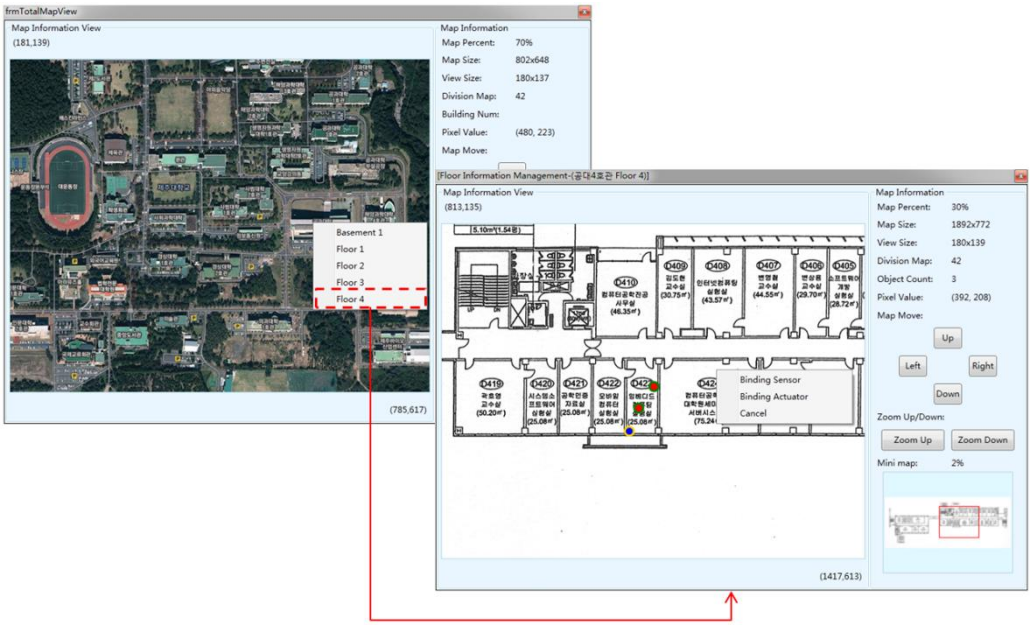

그림 135 실외 지도 뷰어의 실행화면

그림 135는 실외 지도 뷰어의 실행화면이다. 실외 지도 뷰어는 야외의 지도 및 지도상에 있는 빌딩 정보의 가시화를 제공한다. 빌딩의 층 옵션을 선택하여 층(실내) 지도 뷰어를 나타낸다. 층 지도 뷰어는 층 지도 및 지도상에 있는 방 정보, 오브젝트 노드 정보를 보여주고 오브젝트 노드를 생성 및 관리하는 도구를 제공한다.

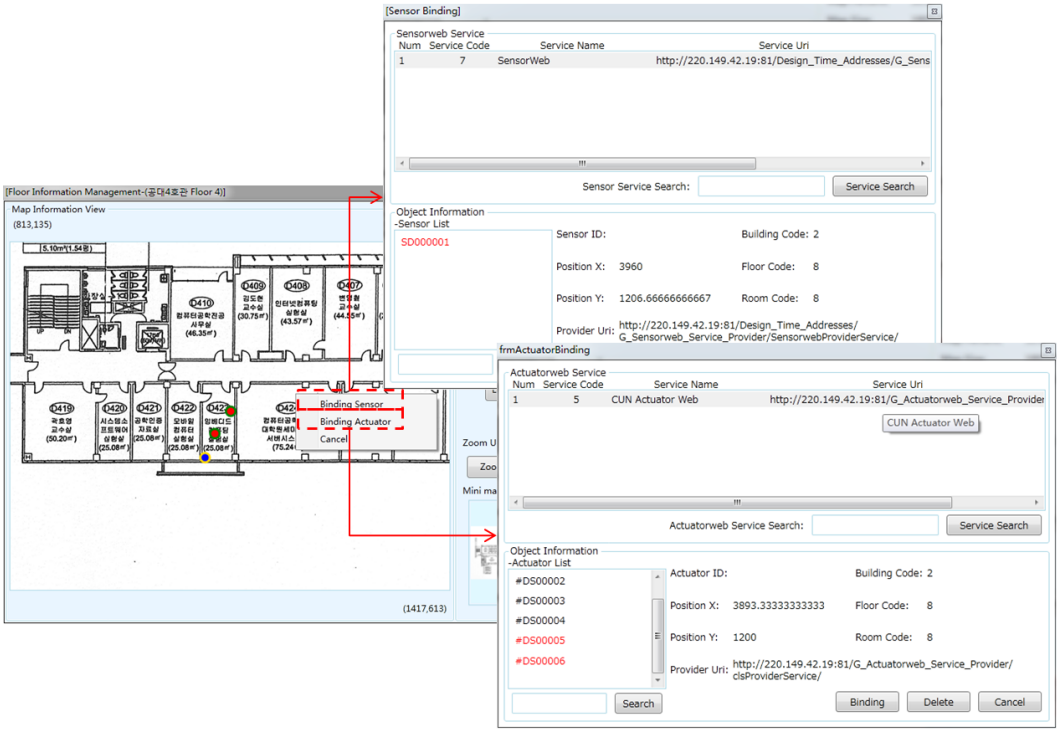

그림 136 오브젝트 위치 바인딩 관리 실행화면

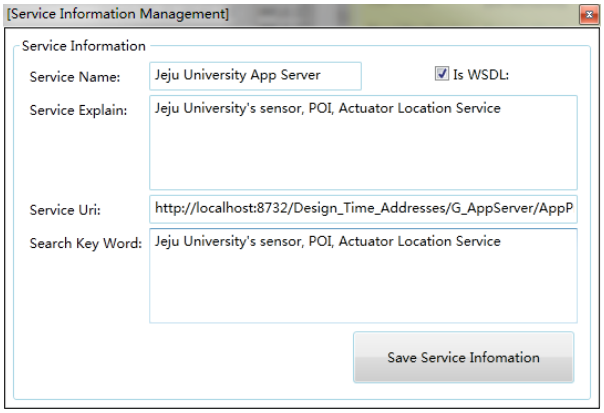

그림 137 응용 서비스 정보 관리 실행화면

그림 136은 오브젝트 위치 바인딩 관리 실행화면이다. 층 지도 뷰어에서 방을 오른쪽 클릭하여 오브젝트 바인딩 메뉴를 나타내어 선택하면 센서 바인딩 및 구동체 바인딩 도구가 나타난다. 바인딩 도구는 서비스를 검색하여 제공하는 오브젝트를 선택하여 바인딩 기능을 제공한다.

그림 137은 응용 서비스 정보 관리 실행화면이다. 이는 서비스 정보를 생성하여 서비스 등록자에서 등록하는 역할을 수행한다. 서비스 정보는 서비스 이름, WSDL제공 여부, 서비스 설명, 서비스 접속 주소, 서비스 검색 키워드를 포함한다. Service Name는 서비스 이름을 입력하는 구역이다. Is WSDL는 서비스 공급자의 WSDL(Web Services Description Language)의 제공여부를 관리한다. Service Explain는 서비스에 대한 설명이다. Service Uri는 서비스 공급자의 접속 주소이다. Search Key Word는 서비스 공급자를 검색하는 키워드를 지정한다. Save Service Information는 서비스 공급자의 정보를 서비스 등록자에게 등록시키는 버튼이다.

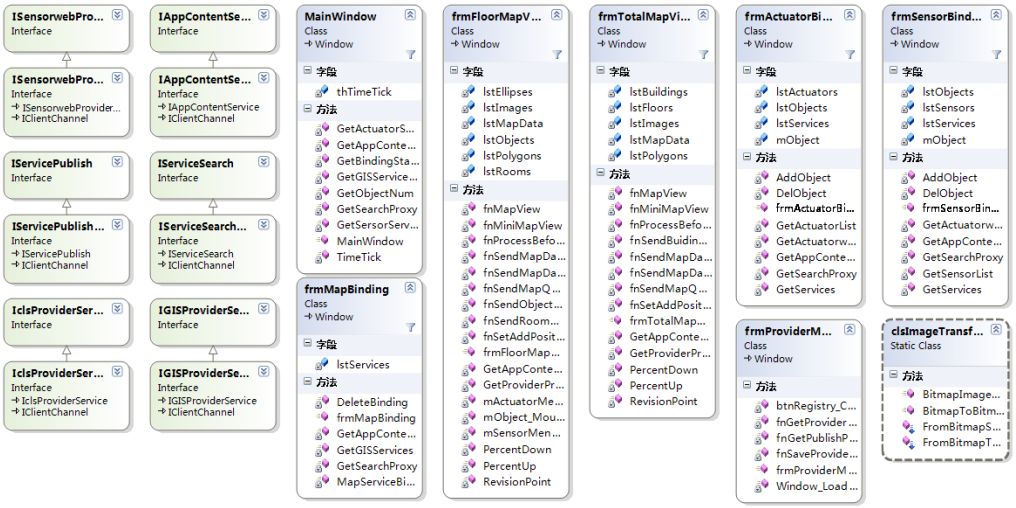

그림 138 응용서버 저작도구 클래스 다이어그램

그림 138은 응용서버 저작도구의 클래스 다이어그램이다. ISensorwebProviderService, IAppContentService, IServicePublish, IServiceSearch, IclsProviderService, IGISProviderService는 외부의 서비스를 참조하기 위한 인터페이스이다. MainWindow, frmFloorMapView, frmTotalMapView, frmActuatorBindinn, frmSensorBinding, frmMapBinding,

frmProviderManagement는 실제 화면에 표시되는 Window form을 상속받는 클래스를 나타낸다. MainWindow는 서비스 콘텐츠 상태의 도시 및 지도 바인딩, 오브젝트 바인딩, 서비스 정보 관리 도구를 호출하는 역할을 담당한다. frmFloorMapView는 층 지도 가시화를 제공하고 frmTotalMapView는 전체 지도 가시화를 제공한다. frmMapBinding은 지도 서비스 바인딩을 위해 가시화를 제공한다. frmActuatorBinding 및 frmSensorBinding은 오브젝트의 바인딩을 위해 가시화를 제공한다. frmProviderMange는 서비스 정보의 관리기능을 담당한다. clsImageTransform은 WPF에서 각종 이미지 유형을 변환하는 메소드를 제공한다.

3.3 클라이언트

그림 139 클라이언트 실행화면

그림 139는 클라이언트 실행화면이다. 클라이언트는 웹 기반의 서비스 등록자가 원하는 서비스 정보를 검색하기 위해 접속할 때 가시화 기능을 담당한다.

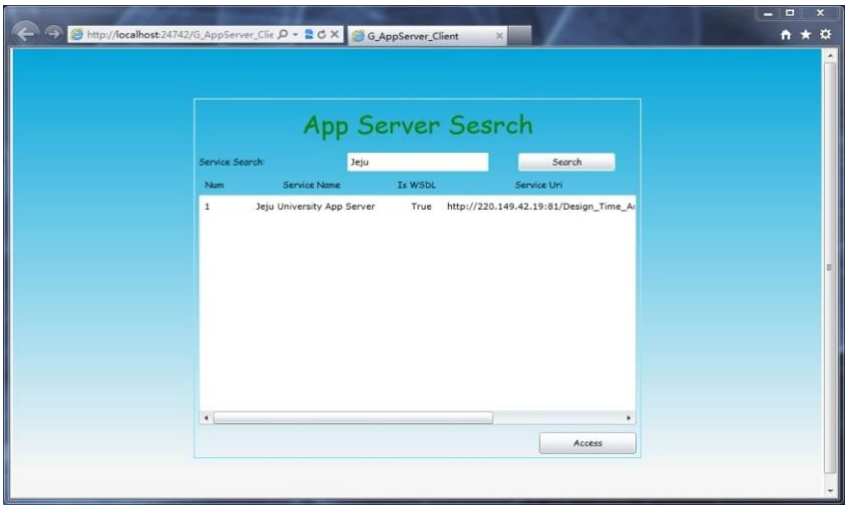

그림 140 응용 서비스 검색 실행화면

그림 140은 클라이언트의 응용 서비스 검색 실행화면이다. 클라이언트 웹 페이지를 접속할 때 그림과 같은 서비스 검색 화면이 나타난다. 사용자가 검색 키워드를 입력하여 검색하고 서비스 등록자가 응용서버의 검색 결과를 화면에 나열한다. 다음에 검색 결과 중에서 원하는 서비스를 선택하면 접속 가능하다.

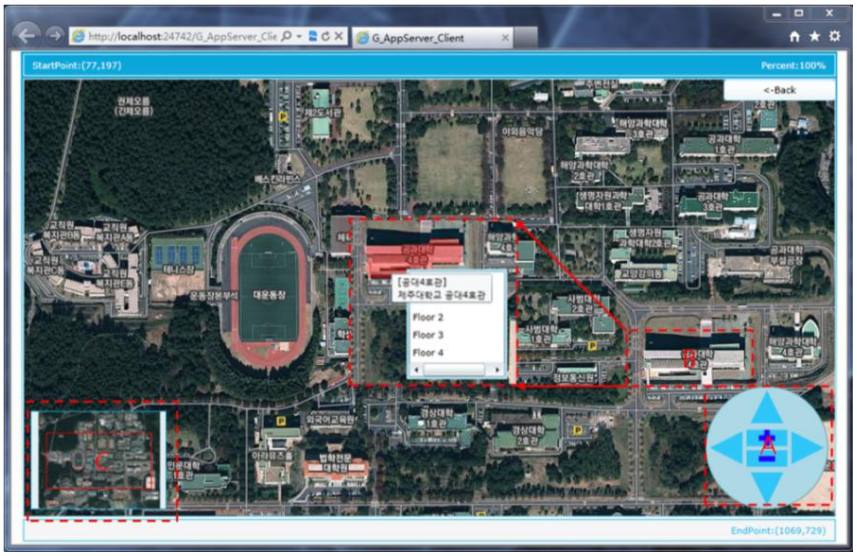

그림 141 실외 지도 뷰어 실행화면

그림 141은 실외 지도 뷰어 실행화면이다. 서비스 검색기에서 원하는 응용서버를 접속하여 처음에 전체 지도 정보를 도시한다. 이는 주로 지도상에 등록된 빌딩의 정보를 조회하는 역할을 담당한다. A구역은 지도 조정(지도 이동, 확대, 축소) 역할을 수행한다. B구역은 등록되는 빌딩의 마크를 지도상에 도시한다. 이 마크를 오른쪽 클릭하면 층 옵션을 나타낸다. C구역은 미니지도를 도시하는 부분이다.

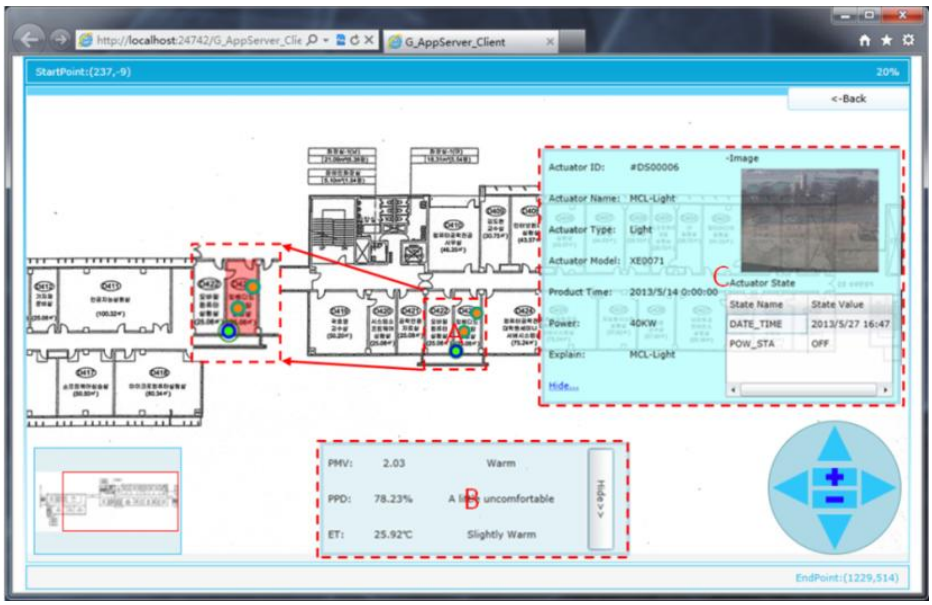

그림 142 실내 지도 뷰어 실행화면(구동체 정보)

그림 142는 실내 지도 뷰어 실행화면이다. 실내(층) 지도 뷰어는 방, 센서 노드, 구동체 노드 정보를 도시하는 역할을 수행한다. A는 실내에 등록된 센서 및 구동체 노드를 보여주고 B구역은 방을 선택하여 그 방의 실내 쾌적 지수 및 상태를 도시한다. C는 구동체 노드를 선택하여 그 구동체의 세부 정보를 도시한다.

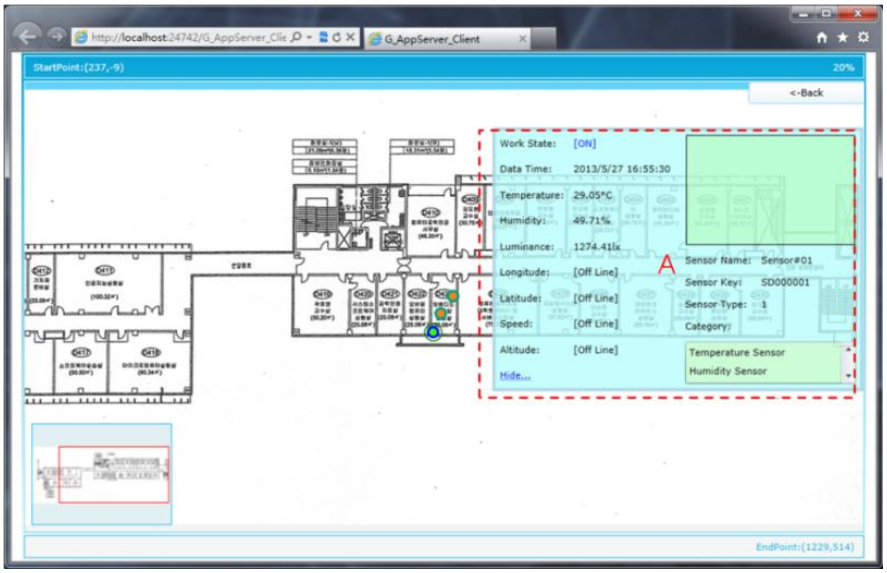

그림 143 실내 지도 뷰어 실행화면(센서 정보)

그림 143은 층 지도에서 센서 노드를 선택하여 센서 노드의 세부정보 및 센싱 데이터를 도시하는 화면이다.

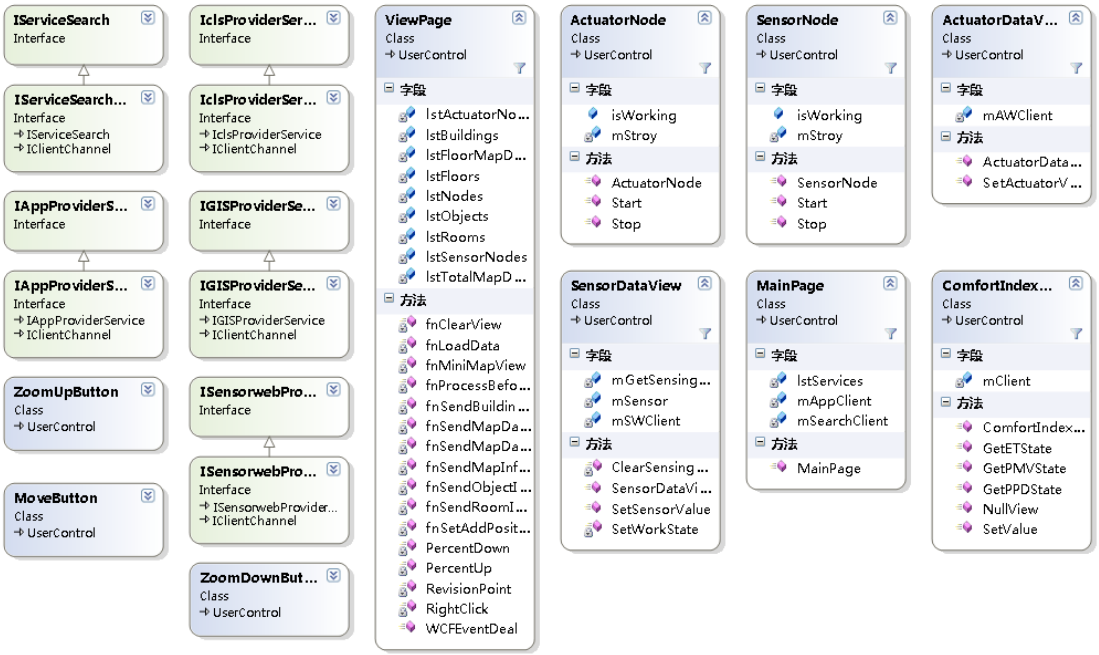

그림 144 클라이언트 클래스 다이어그램

그림 144는 클라이언트의 클래스 다이어그램이다. IServiceSearch, IclsProviderService, IAppProviderService, IGISProviderService, ISensorwebProviderService는 외부 WCF 서비스를 접속하기 위한 인터페이스이다. MainPage, ViewPage, ActuatorNode, SensorNode, ActuatorDataView, SensorDataView, ComfortIndexView, ZoomUpButton, ZoomDownButton, MoveButton은 실제 화면을 표시하는 실버라이트 Control이다. MainPage는 응용서비스의 검색 가시화를, ViewPage는 지도의 가시화 기능을 제공한다. ActuatorNode는 구동체 노드 마크를, 센서노드는 센서 노드 마크를 도시한다. SensorDataView는 센서의 세부 정보 및 센싱 데이터를, ActuatorDataView는 구동체의 세부 정보 및 동작 상태 정보를 도시하는 역할을 담당한다. ComfortIndexView는 방의 쾌적 지수, 상태 정보를 도시하는 역할을 수행한다. ZoomUpButton, ZoomDownButton 및 MoveButton은 지도를 조정하는 역할을 수행한다.

3.4 서비스 등록자

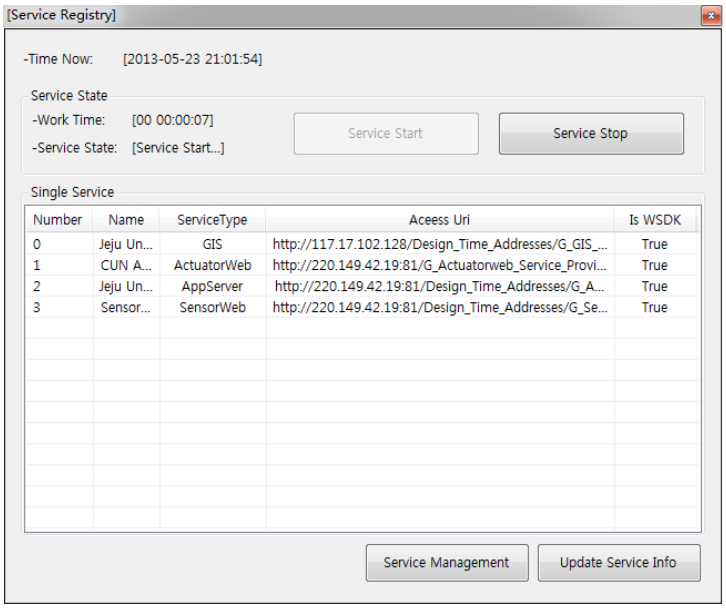

그림 145 서비스 등록자 실행화면

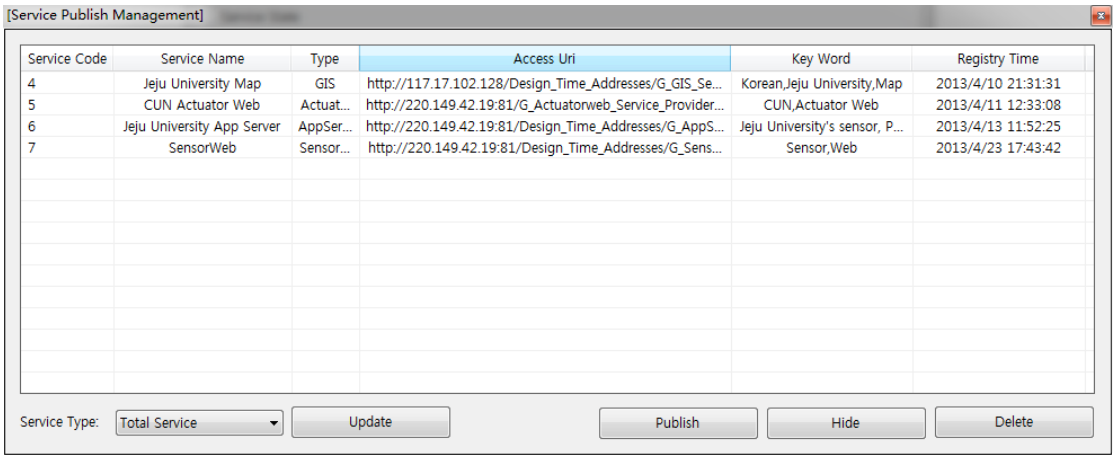

그림 146 서비스 관리자 실행화면

그림 145는 서비스 등록자의 실행화면이다. 서비스 등록자는 외부에서 서비스 등록을 할 때 검색 서비스를 제공하며 등록된 서비스 정보를 관리하는 역할을 수행한다. Time Now는 서비스 등록자가 동작환경의 현재시간이다. Work Time는 서비스 등록자가 정상하게 동작하는 시간이다. Service State는 서비스 등록자의 서비스 동작상태(꺼짐/켜짐)를 제어한다.

그림 146은 서비스 관리자의 실행화면이다. Service Management 버튼은 서비스 관리자를 호출한다. Update Service Info는 등록되는 서비스 정보(서비스 이름, 서비스 유형, 서비스 주소)를 다시 요청한다. Update버튼은 등록되는 서비스 정보(서비스 이름, 서비스 유형, 서비스 주소)를 다시 요청한다. Publish버튼은 선택하는 서비스를 외부에게 검색 개방한다. Hide버튼은 선택하는 서비스를 외부에게 검색 금지한다. Delete는 선택하는 서비스 정보를 삭제한다.

그림 147은 서비스 등록자의 클래스 다이어그램이다. IServicePublish 및 IServiceSearch는 서비스 검색, 등록을 위한 외부 노출 인터페이스이다. ServicePublish는 등록되는 서비스 정보를 DB에 저장한다. ServiceSearch는 DB에 저장되는 Service정보를 읽고 보낸다. frmMainPage, frmServiceManage는 실제 화면에 표시되는 Window form을 상속받는 클래스를 나타낸다. frmMainPage는 서비스 제어 및 등록되는 서비스 정보의 가시화를 제공하며,

frmServiceManage는 등록되는 서비스 정보를 관리하는 역할을 수행한다.

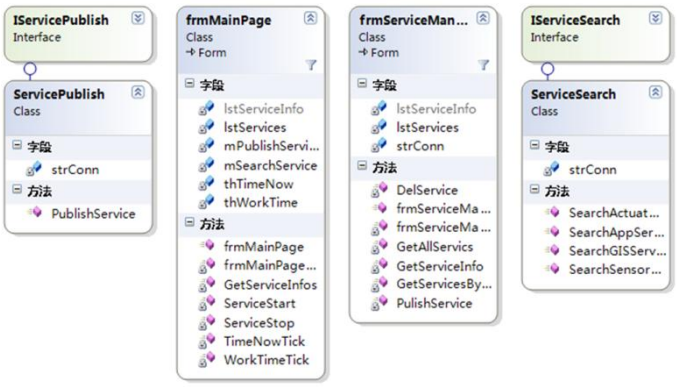

그림 147 서비스 등록자 클래스 다이어그램

4 성능 분석 및 평가

4.1 실험환경

실험환경은 표 26와 같다. 실험은 실내 IOT 시스템의 주요 기능에서 성능 분석을 한다. 모든 성능분석은 해당 주요모듈 별로 20회에 걸쳐 수행시간을 측정하고 분석한다.

표 26 실행환경

Division	Detail
Operating System	Microsoft Windows 7(X64) Microsoft Windows Server 2008(X64)
Development Environment	.Net Framework 3.5, 4.0
Development Tool	Visual Studio .Net 2010
Programming Language	C#, XMAL, XML
DBMS	Microsoft SQL Server 2008 R2
Hardware	CPU: Intel® Core™ i3-2125 @ 3.30GHz Intel® Core™ i3-3220 @ 3.30GHz RAM: 4GB Graphics: NVIDIA GeForce GT 440 AMD Radeon HD 6570

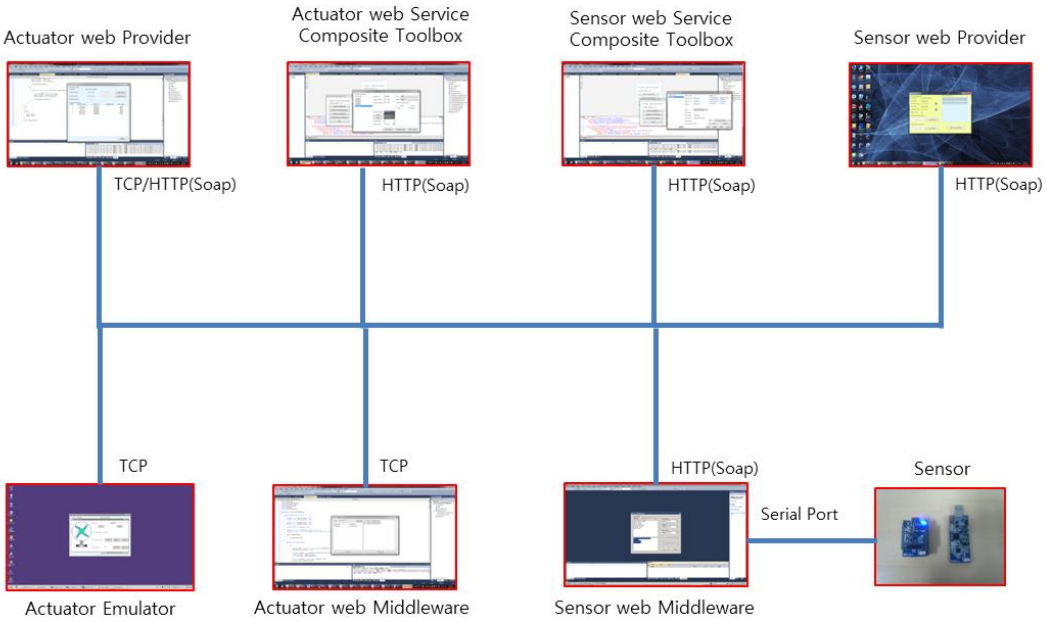

그림 148 스마트 제어의 실험 네트워크 구성

본 논문의 III장 1절에서 IOT 시스템에 대해 서비스 공급자를 기반의 IOT

시스템 구조와 응용서버를 기반의 IOT 시스템 구조를 제출한다. 그래서 본 절에서 두 시스템 구조를 구현하고 클라이언트가 IOT 시스템의 구동체 제어 및 센싱 데이터를 요청하고 응답하는 시간을 측정한다.

실험환경의 네트워크 구성은 그림 148과 같다. 실험을 위해 센서웹 공급자를 위한 데스크탑 서버 1대, 구동체웹 공급자를 구동하기 위한 데스크탑 서버 1대, 응용서버를 위한 데스크탑 서버 1대, 클라이언트, 구동체 에뮬레이터, 구동체 미들웨어, 센서 미들웨어를 구동하기 위한 데스크탑 각 1대를 사용한다.

표 27 스마트 제어의 실험 네트워크 환경

Module Name	Address
센서웹 서비스 공급자	http://220.149.42.19:81/Design_Time_Addresses/G_Sensorweb_Service_Provider/SensorwebProviderService/
구동체웹 서비스 공급자	http://220.149.42.19:81/G_Actuatorweb_Service_Provider/clsProviderService/
센서웹 미들웨어	117.17.102.28
구동체웹 서비스 공급자	220.149.42.19
구동체 에뮬레이터	117.17.102.128

4.2 실내 환경 지능 제어

기존 응용서버의 데이터베이스에서는 제주대학교 공대4호관의 D423호실에 조도 센서 및 조명기구를 설정한다. 응용서버가 그 방에 대해 1초 한번 실내 조도 상태를 확인하여 조명기구를 제어한다.

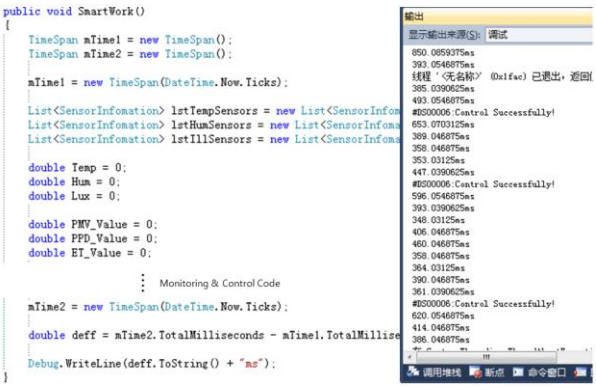

그림 149 지능 실내 환경제어 분석

그림 149는 응용서버의 지능 제어의 수행시간을 측정하는 소스이다. 지능 제어는 내부 처리하는 모듈이라서 외부에서 요청 불 가능해서 전용 측정 프로그램을 사용 못 한다. 그래서 집적 응용서버의 스마트 제어 모듈에서 수행시간을 계산하는 기능을 추가한다.

Times	Cost
1	385.0390625
2	493.0546875
3	653.0703125
4	389.046875
5	358.046875
6	353.03125
7	447.0390625
8	596.0546875
9	393.0390625
10	348.03125
11	406.046875
12	460.046875
13	358.046875
14	364.03125
15	390.046875
16	361.0390625
17	620.0546875
18	414.046875
19	386.046875
20	435.046875

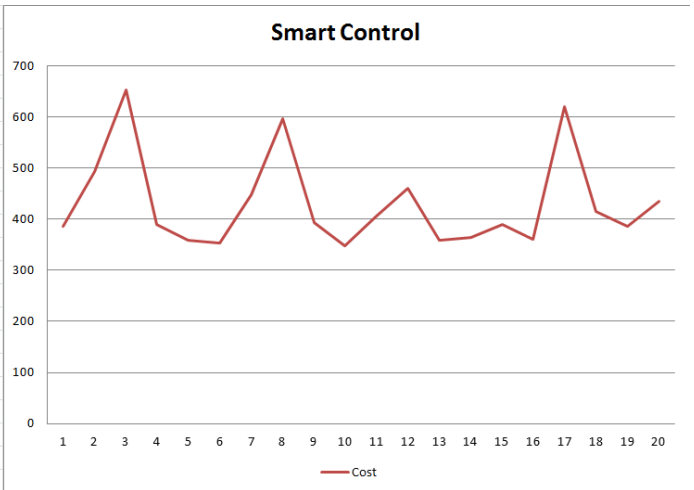

그림 150 지능 제어 소요 시간

그림 150은 응용 서비스가 지정 방에 대한 스마트 제어의 수행시간이다. 그래프의 가로축은 수행 횟수를 나타내고 세로축은 수행시간(ms)을 나타낸다. 각각의 수행횟수 별로 385.0390625, 493.0546875, 653.0703125, 389.046875, 358.046875, 353.03125, 447.0390625, 596.0546875, 393.0390625, 348.03125, 406.046875, 460.046875, 358.046875, 364.03125, 390.046875, 361.0390625, 620.0546875, 414.046875, 386.046875, 435.046875ms의 수행시간이 측정되었고 최소 수행시간은 348.03125ms이고, 최대 수행시간은 653.0703125ms이다. 그래서 응용서버는 한 방에 대해 스마트 제어하는데 걸리는 평균시간은 430.4953125ms로 분석된다.

VI. 저작도구를 기반의 실내 GIS 시스템

1. 실내 GIS 시스템 설계

본 논문에서 실내 GIS 시스템은 주로 클라이언트 및 응용서버 저작도구에게 지도 가시화 서비스 제공하기 위해 실외지도 정보, 빌딩 정보, 층 지도 정보, 방 정보를 생성, 공급 및 관리하는 시스템이다. 실내 GIS 시스템은 GIS 공급자와 GIS 서비스 저작도구로 구성한다.

1.1 실내 GIS 시스템 세부설계

1.1.1 GIS 공급자

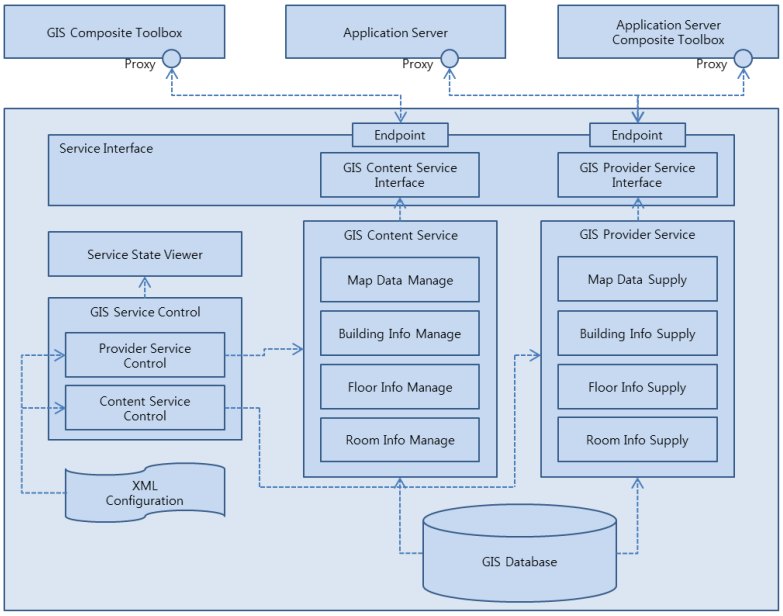

그림 151 실내 GIS 공급자 세부구조

그림 151은 실내 GIS 공급자의 세부구조이다. GIS 공급자는 GIS Content Service를 통해 GIS 서비스 저작도구가 생성하는 서비스 콘텐츠를 저장한다. 그리고 GIS 공급 서비스를 통해 응용서버 및 응용서버 저작도구에게 지도 서비스를

제공한다. GIS Content Service는 Map Data Manage, Building Info Manage, Floor Info Manage와 Room Info Manage로 구성된다. Map Data Manage는 지도 이미지를 관리하는 역할을, Building Info Manage는 지도상에 빌딩 관련 정보를 관리하는 역할을, Floor Info Manage는 빌딩내의 층 관련 정보를 관리하는 역할을, Room Info Manage는 층 지도상의 방 관련 정보를 관리하는 역할을 제공한다. GIS 공급 서비스는 Map Data Supply, Building Info Supply, Floor Info Supply와 Room Info Supply로 구성된다. Map Data Supply는 지도 데이터를, Building Info Supply는 지도상에 빌딩 관련 정보를, Floor Info Supply는 빌딩내의 층 관련 정보를, Room Info Supply는 층 지도상의 방 관련 정보를 공급하는 역할을 수행한다. GIS Service Control은 Provider Service Control과 Content Service Control로 구성되고 Provider Service 및 Content Service의 열림, 닫음을 제어한다. Service State Viewer는 서비스의 동작 상태를 관리자에게 보여준다. XML Configuration은 WCF Service의 동작 환경을 배치한다. GIS Database는 지도 서비스가 제공하는 서비스 콘텐츠를 저장한다.

1.1.2 GIS 서비스 저작도구

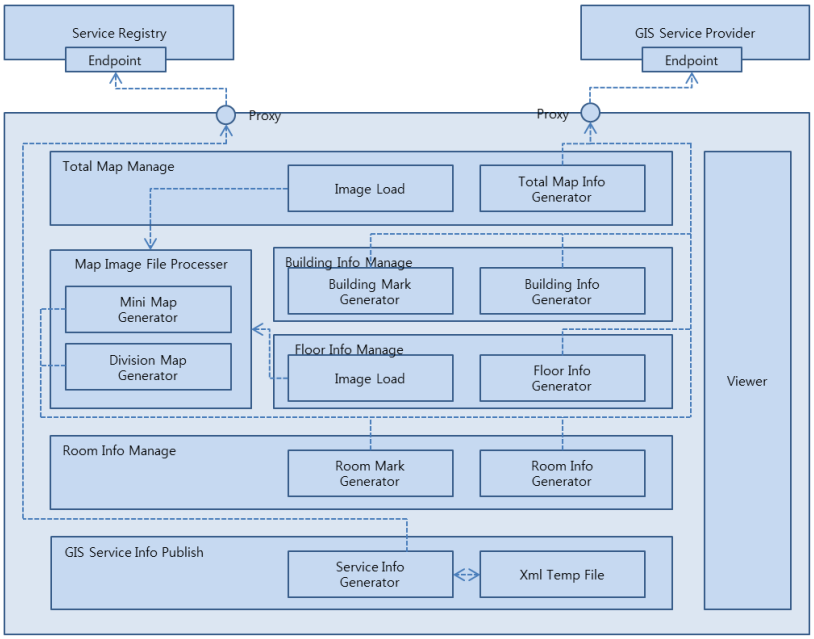

그림 152 GIS 서비스 저작도구 세부구조

그림 152는 GIS 서비스 저작도구의 세부구조이다. GIS 서비스 저작도구는 GIS 공급자의 콘텐츠 서비스를 통해 서비스 콘텐츠를 생성하여 관리하는 역할을 수행하고 서비스 등록자의 등록 서비스를 통해 GIS 서비스 정보를 등록한다. Total Map Manage는 Image Load, Total Map Info Generator로 구성된다. Image Load는 Map Image File Processer를 통해 Mini Map, Division Map을 생성하여 GIS 공급자를 저장한다. Total Map Info Generator는 전체 지도의 정보(지도 사이즈, 미니 지도 사이즈 등)를 생성하여 저장한다. Building Info Manage는 Building Mark Generator, Building Info Generator로 구성된다. Building Mark Generator는 관리자가 지도상에서 빌딩 구역 표기를 통해 위치정보를 생성하고 Building Info Generator는 빌딩 정보(빌딩 이름, 빌딩, 층 정보 등)를 생성한다. Floor Info Manage는 Image Load, Floor Info Generator로 구성된다. Image Load는 Floor Map Image를 Map Image File Processer를 통해 Mini Map, Division Map을 생성하여 GIS 공급자를 저장한다. Floor Info Generator는 층 지도의 정보(지도 사이즈, 미니 지도 사이즈 등)를 생성하여 저장한다. Floor Info Manage는 Floor Mark Generator, Floor Info Generator로 구성된다. Floor Mark Generator는 관리자가 지도상에서 방 구역의 표기를 통해 위치정보를 생성하고 Floor Info Generator는 빌딩 정보(방 이름, 번호, 설명 등)를 생성한다. GIS Service Info Publish는 Service Info Generator를 통해 Service정보(검색 키워드, 이름, 주소 등)를 생성하여 XML임시 파일에 저장하여 서비스 등록자에게 등록한다.

2. 실내 GIS 시스템 구현

2.1 GIS 공급자

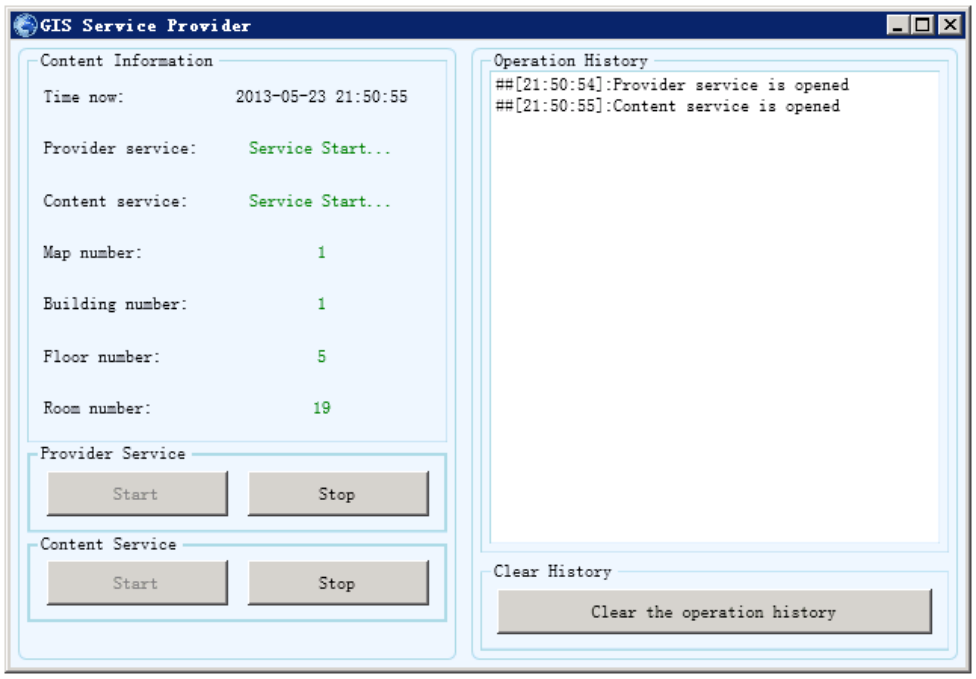

그림 153 GIS 공급자 실행화면

그림 153은 GIS 공급자의 실행화면이다. GIS 공급자가 공급 서비스 및 콘텐츠 서비스를 제공한다. Time Now는 GIS 공급자의 동작환경의 현재 시간이다. Provider Service는 GIS 공급자의 공급 서비스의 동작상태를 표시한다. Content Service는 GIS 서비스 콘텐츠 서비스 의 동작상태를 표시한다. Map Number는 DB에서 실외 지도의 개수를 표시한다. Building Number는 DB에서 저장하는 빌딩정보 개수를 의미한다. Floor Number는 DB에서 저장하는 층 정보 개수를 의미한다. Room Number는 DB에서 저장하는 방 정보 개수를 의미한다. Start버튼을 통해 해당 서비스를 켜진다. Stop버튼을 통해 해당 서비스를 꺼진다. Operation History는 GIS 서비스 제어작업을 기록한다.

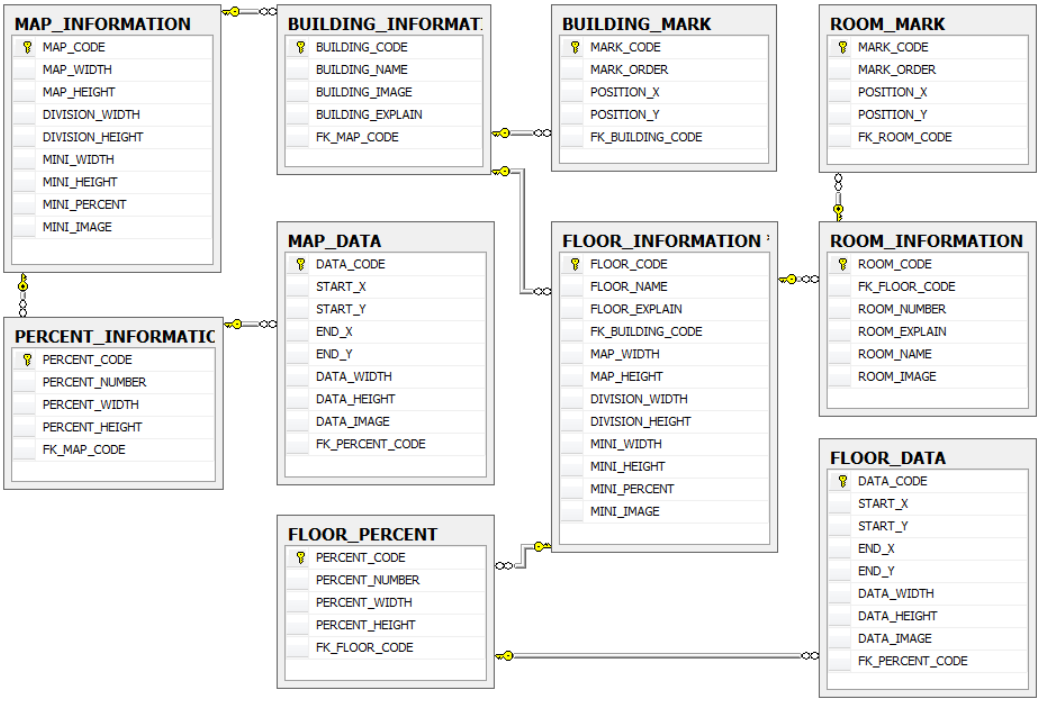

그림 154 GIS 데이터베이스 관계도

그림 154는 GIS 데이터베이스 관계도 이다. Map_Informaiton은 전테 지도 정보(지도 사이즈, 미니지도, 미니지도 사이즈 등)를 저장하는 테이블이다. Percent_Information은 10%~100% 축소한 전체 지도 이미지 사이즈 정보를 저장하는 테이블이다. Map_Data는 전체 지도의 분할 지도를 저장하는 테이블이다. Building_Information은 빌딩 정보를 저장하는 테이블이다. Building_Mark는 빌딩의 구역 정보를 저장하는 테이블이다. Floor_Information은 층 지도 (지도 사이즈, 미니지도, 미니지도 사이즈 등)를 저장하는 테이블이다. Floor_Percent는 10%~100% 축소하는 층 지도 이미지 사이즈 정보를 저장하는 테이블이다. Floor_Data는 층 지도의 분할 지도를 저장하는 테이블이다. Room_Information은 방 정보를 저장하는 테이블이다. Room_Mark는 방의 구역 정보를 저장하는 테이블이다.

그림 155 GIS 공급자 클래스 다이어그램

그림 155는 GIS 공급자의 클래스 다이어그램이다. IGISProviderService 및 IGISContentService는 공급 서비스 및 콘텐츠 서비스를 외부에 노출하기 위한 인터페이스이다. GISProviderService는 지도 정보를 공급하는 역할을 제공한다. GISContentService는 지도 정보를 관리하는 역할을 제공한다. MainWindow는 서비스를 제어하고 서비스 상태의 뷰어를 제공한다. stFloorInformation은 층 지도 정보를, stMark는 마크 정보를, stMapInformation은 전체 지도 정보를 저장하는 구조체이다. stRoomInformation은 방 정보를, stMiniMap은 미니지도 정보를, stMapData는 분할 지도 정보를, stSize는 지도 사이즈를, stBuildingInformation은 빌딩 정보를 저장하는 구조체이다.

2.2 GIS 서비스 저작도구

그림 156 GIS 서비스 저작도구 실행화면

그림 156은 GIS 서비스 저작도구 실행화면이다. 서비스 콘텐츠 도구는 서비스 콘텐츠의 생성관리를 위한 편의성을 제공하기 위한 도구이다. 관리자는 이를 이용하여 서비스 공급자나 등록자에게 서비스 콘텐츠 정보를 생성하고 저장하는 기능을 제공한다. 서비스 콘텐츠 도구는 지도정보 관리, 빌딩정보 관리, 층 정보 관리, 서비스 정보 관리, 방 정보 관리를 제공한다. 각 관리 모듈은 사용자의 사용 편의성을 제공하기 위해 지도 가시화를 기반으로 제어를 제공한다.

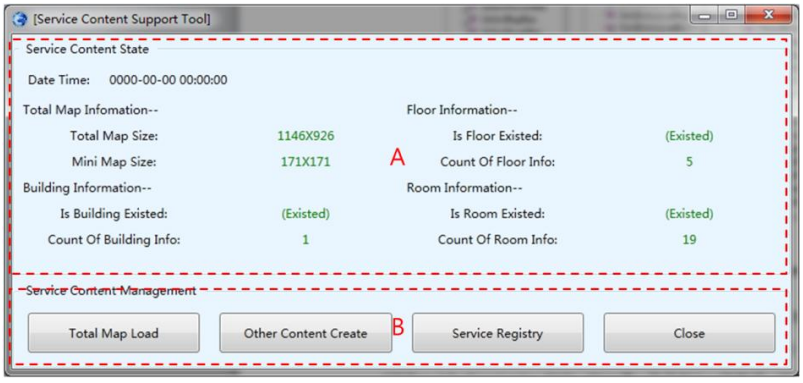

그림 157 GIS 서비스 저작도구 메뉴 실행화면

그림 157은 GIS 서비스 저작도구 메뉴 실행화면이다. 서비스 콘텐츠 도구 창은 서비스 공급자단의 콘텐츠 상황을 관리자에게 직관적인 정보로 보여준다. 그리고 다른 정보 관리 창의 호출을 통하여 다른 정보를 관리하는 기능을 제공한다. A는 콘텐츠의 상태 정보를 표시한다. 현재 시간, 전체 지도 정보 상태, 빌딩정보 상태, 층 정보 상태, 방 정보 상태 등 상태 정보를 서비스 공급자의 공급 서비스로부터 요청하고 출력한다. Data Time는 GIS 서비스 저작도구의 실행환경 현재시간이다. Total Map Information는 실외 지도 이미지와 미니 지도 이미지 사이즈를 표시한다. Building Information는 등록되는 빌딩 정보의 개수를 표시한다. Floor Information는 등록되는 층 정보의 개수를 표시한다. Room Information는 등록되는 방 정보를 표시한다. B는 다른 정보 관리 버튼을 제공한다. 여기서는 전체 지도 관리 창을 호출하고, 기타 콘텐츠 정보 관리 창을 호출하며 서비스 정보 등록기능을 수행한다.

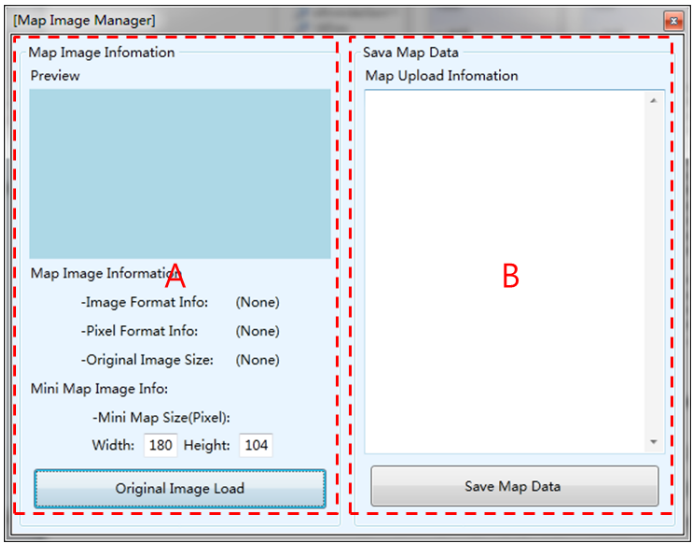

그림 158 지도 이미지 관리 실행화면

그림 158은 지도 이미지 관리의 실행화면이다. 지도 이미지 관리자는 전체 지도 이미지를 로드하고 분할 처리하여 서비스 공급자에게 보내는 역할을 수행한다. 학교 캠퍼스와 같은 경우 전체 학교 지도의 화상도가 일정 수준이상을 되면

그 이미지 파일의 사이즈는 아주 크다. 클라이언트가 이 데이터를 다 다운로드 받고 출력하려면 지연시간은 과도해진다. 그렇다고 이미지 파일을 처리 없이 단순히 저장해서는 안 된다. 이를 위해 이미지 파일을 분할하고 분할 정보와 분할 이미지 데이터를 같이 저장하면 이 문제를 해결할 수 있다.

A구역은 지도 이미지 로드 버튼을 통해 로컬의 지도 이미지 파일을 메모리에 저장시킨다. 그리고 이 이미지의 포맷, 사이즈 등 정보를 표시한다. 미니지도의 사이즈 조정을 통하여 미니지도 사이즈를 지정한다. B구역은 도입된 지도 이미지를 분할처리하고 저장하는 과정을 표시하는 구역이다.

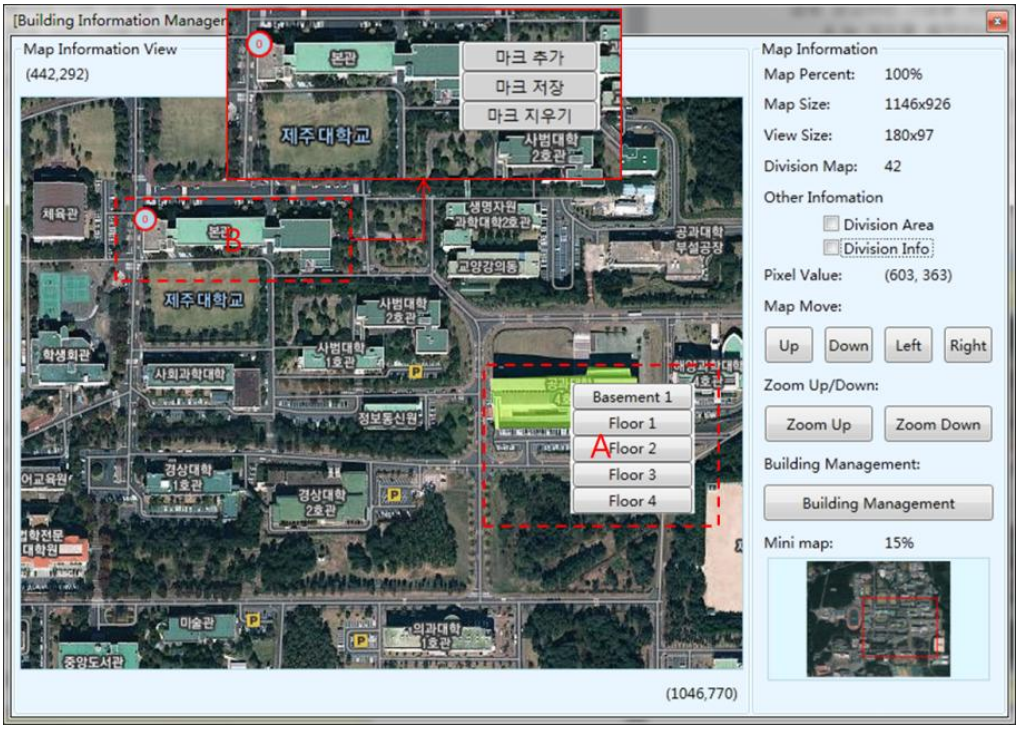

그림 159 실외 지도 관리 실행화면

그림 159는 실외 지도 관리 실행화면이다. 실외 지도는 큰 범위의 지도 뷰어이다. 최소의 가시화 단위는 빌딩이다. 전체 지도 뷰어는 전체 지도상에서 빌딩정보의 출력 및 생성하는 기능을 수행한다. A는 빌딩의 층 옵션이고 B는 빌딩의 마크를 생성하는 기능을 제공한다. Map Percent는 현재 지도의 퍼센트를 표시한다.

Map Size는 현재 지도의 전체 사이즈를 의미한다. View Size는 현재 지도의 가시화 범위이다. Division Map는 뷰어 범위내의 분할 지도 개수를 표시한다. Other Information는 분할 구역 정보와 분할 구역을 지도 뷰어에서 표시한다. Pixel Value는 마우스가 지정하는 위치이다. Map Move는 지도 이동조정을 제공한다. Zoom Up/Down는 지도 확대/축소 조정을 제공한다. Building Management는 방 정보 관리자를 호출한다.

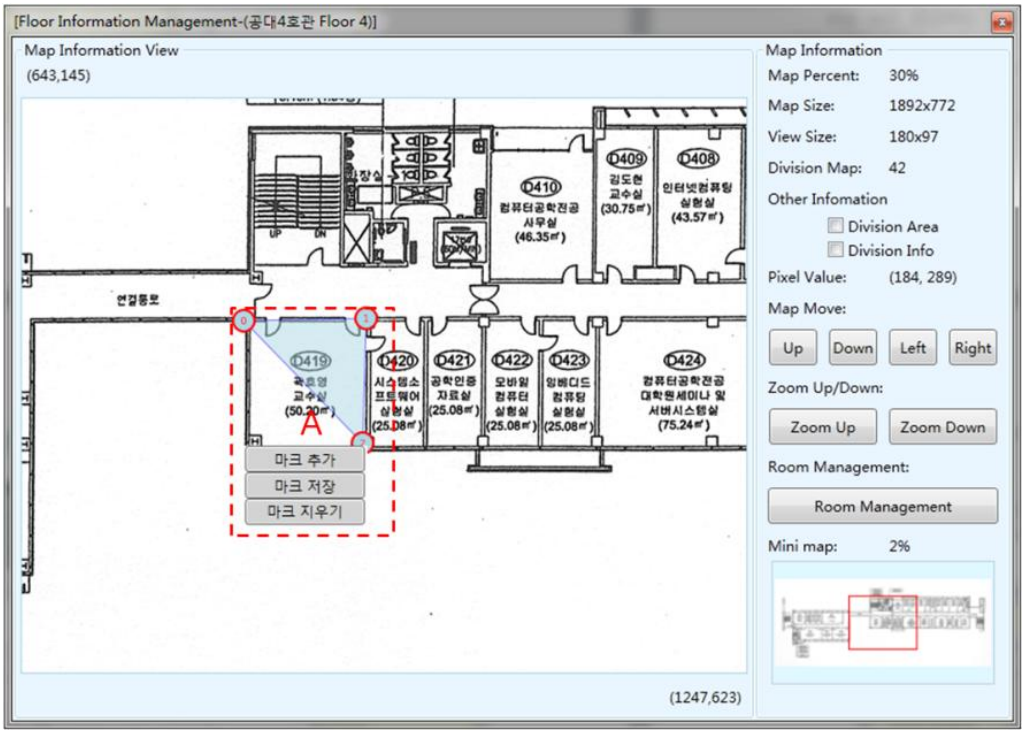

그림 160 실내 지도 관리 실행화면

그림 160은 실내 지도 관리 실행화면이다. A는 층 지도에서 방 구역을 생성하는 역할을 담당한다. Map Percent는 현재 지도의 퍼센트를 표시한다. Map Size는 현재 지도의 전체 사이즈를 의미한다. View Size는 현재 지도의 가시화 범위이다. Division Map는 뷰어 범위내의 분할 지도 개수를 표시한다. Other Information는 분할 구역 정보와 분할 구역을 지도 뷰어에서 표시한다. Pixel Value는 마우스가 지정하는 위치이다. Map Move는 지도 이동조정을 제공한다. Zoom Up/Down는 지도 확대/축소 조정을 제공한다.

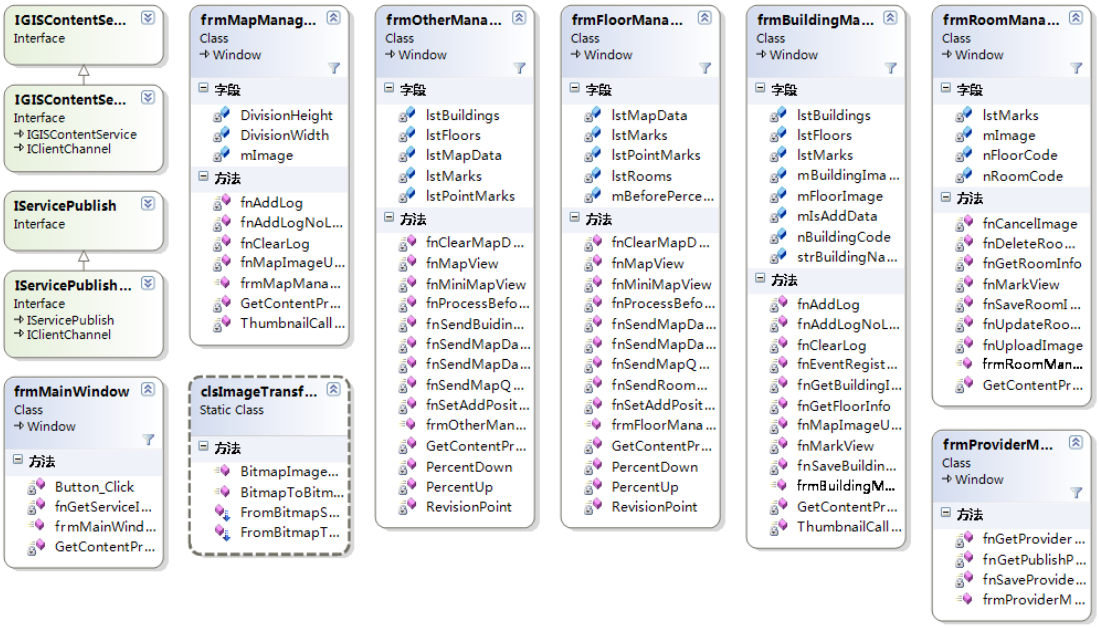

그림 161 GIS 서비스 저작도구 클래스 다이어그램

그림 161은 GIS 서비스 저작도구 클래스 다이어그램이다. IGISContentService 및 IServicePublish는 지도 콘텐츠 서비스, 등록 서비스를 접속하기 위한 인터페이스이다. frmMainWindow는 지도 서비스 콘텐츠 상태를 도시하며 다른 콘텐츠 관리 도구의 호출 기능을 제공한다. frmMapManagement는 전체 지도 이미지를 도입하여 분할 처리하는 역할을 수행한다. frmOtherManagement는 전체 지도상에서 빌딩 정보를 생성 및 관리하는 역할을 수행한다. frmFloorManagement는 층 지도상에서 방 정보를 생성 및 관리하는 역할을 수행한다. frmBuildingManagement는 빌딩 정보를, frmRoomManagement는 방 정보를 생성하는 도구이다. frmProviderMange는 서비스 정보를 관리하는 기능을 담당한다. clsImageTransform은 WPF에서 각종 이미지 유형을 변환하는 메소드를 제공한다.

2.3 지도 가시화 우화성능

“센서웹 기반의 COST 설계 및 구현”의 논문에서 제시하는 GIS시스템에서

기존 지도 이미지를 분할하고 블록형태로 DB에서 저장하고, 클라이언트가 도시하는 구역 정보에 의해 분할 지도 이미지를 요청한다. 이 방법을 사용하여 클라이언트가 전체 지도 이미지의 요청 안 해도 원하는 구역의 지도를 볼 수 있다. 그런데 사용자가 지도 이동, 확대나 축소작업 할 때 도시 구역의 모두 이미지 블록을 다시 요청하고 도시해야 한다.

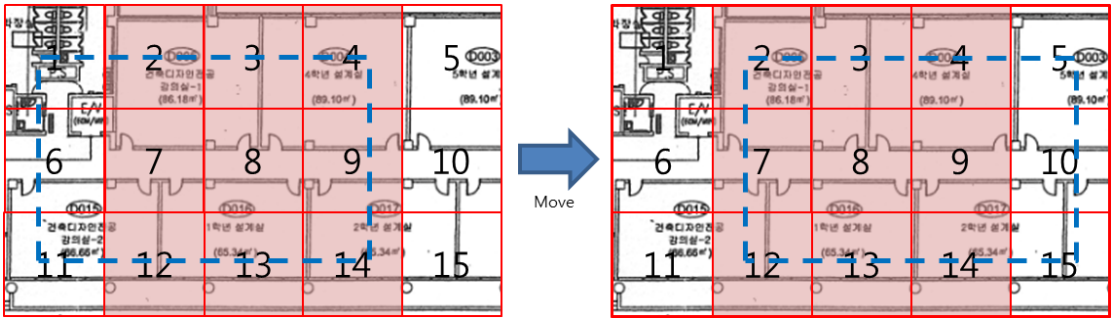

그림 162 지도 이미지 이동 처리원리

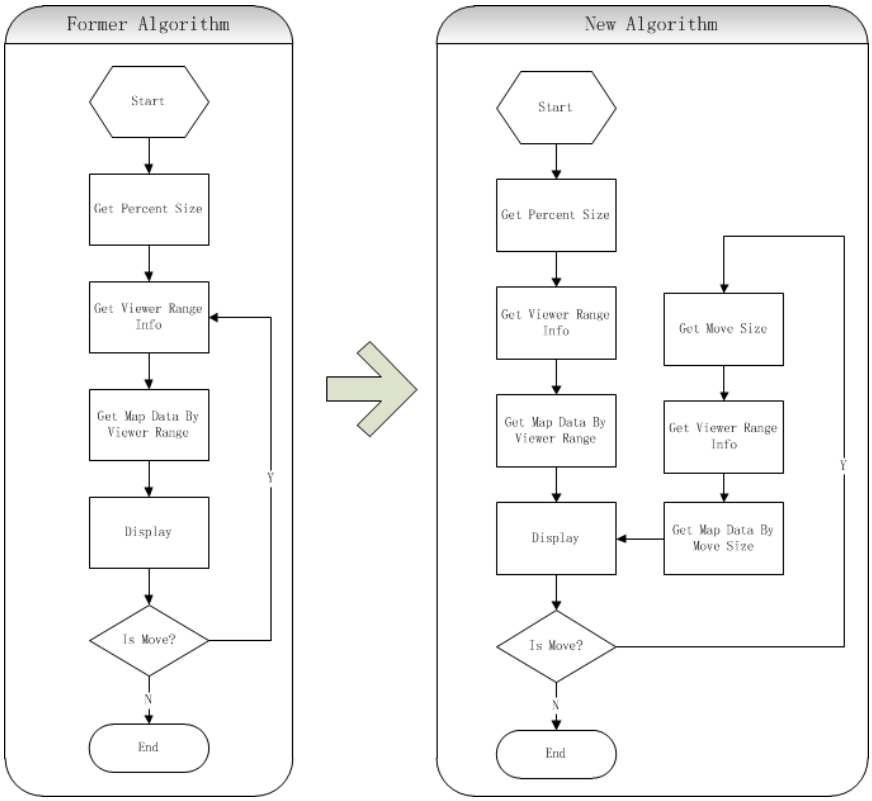

그림 163 분할 지도 요청 알고리즘 우화

그림 162는 지도 이동에 따라서 분할 지도를 요청하는 원리이다. 그림에서 점선은 클라이언트가 도시하는 구역이다. 이동하기 전에 도시 구역정보에 의해 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13과 14번 분할 지도를 요청하여 사용자 원하는 구역을 제대로 볼 수 있다. 다음에 사용자가 지도를 오른쪽으로 이동한다. 그러면 전의 지도를 다 지우고 다시 2, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14와 15번 분할 지도를 요청하고 출력하면 이동하는 효과를 나타난다. 그런데 2, 3, 4, 7, 8, 9, 12, 13과 14번 분할 지도 중복 요청하는 일을 발생한다. 본 논문에서 우화 방안을 사용하여 이 문제를 해결한다.

그림 163은 분할 지도를 요청의 우화 알고리즘이다. 매번 이동할 때 도시 구역정보를 대신해 이동간격을 계산하여 이를 이용해 새 분할지도(5, 10, 15번 분할지도)를 요청하여 출력하면 된다. 다음에 실험을 통해 두 알고리즘의 성능에 대해 분석한다.

3. 성능 분석 및 평가

3.1 실험환경

실험환경은 표 28과 같다. 실험은 GIS 시스템의 주요 기능에서 성능 분석을 한다. 모든 성능분석은 해당 주요모듈 별로 20회에 걸쳐 수행시간을 측정하고 분석한다.

표 28 실험환경

Division	Detail
Operating System	Microsoft Windows Server 2008(X64)
Development Environment	.Net Framework 3.5, 4.0
Development Tool	Visual Studio .Net 2010
Programming Language	C#, XMAL, XML
DBMS	Microsoft SQL Server 2008 R2
Hardware	CPU: Intel® Core™ i3-3220 @ 3.30GHz RAM: 4GB Graphics: AMD Radeon HD 6570

GIS 시스템은 주로 지도 데이터를 요청하여 응답하는 수행시간을 측정한다. GIS 공급자의 데이터베이스에 저장된 분할 지도 데이터를 읽어나서 요청자에게 전송하는 시간을 측정한다.

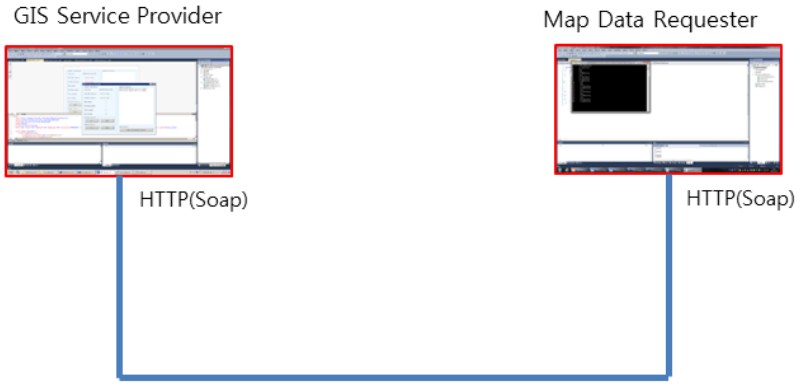

그림 164 GIS 서비스의 실험 네트워크 구성

실험환경의 네트워크 구성은 그림 164는 같다. 실험을 위해 GIS 공급자를 구동하기 위한 데스크탑 서버 1대, GIS 서비스 지도 정보 요청자를 구동하기 위한 노트북 1대를 사용한다.

표 29 GIS 서비스의 실험 네트워크 환경

Module Name	Address
GIS 공급자	http://117.17.102.128/Design_Time_Addresses/G_GIS_Service_Provider/GISProviderService/
클라이언트	117.17.102.197

3.2 지도 데이터 질의 성능평가

표 30 실험 결과(지도정보 요청)

Num	Respond Time		Respond Data Size		Move	
	Former Algorithm	New Algorithm	Former Algorithm	Former Algorithm	X	Y
1	222ms	93ms	1290341byte	366758byte	100pixel	100pixel
2	258ms	142ms	1347999byte	697208byte	200pixel	200pixel
3	241ms	182ms	1408094byte	993872byte	300pixel	300pixel
4	241ms	217ms	1418586byte	1187596byte	400pixel	400pixel

표 30은 이상의 두 알고리즘을 사용하여 실험 결과이다. 4번의 실험의 통해 같은 이동작업을 수행할 때 두 알고리즘을 이용하고 분할 지도 요청하는 소요시간과 요청하는 지도 데이터 사이즈에 대해 측정하여 분석한다.

본 실험은 제주대학교 실외지도를 대상으로 클라이언트가 지도 이동할 때 지도 화면 갱신의 소요시간과 요청에 대한 데이터의 량을 비교한다

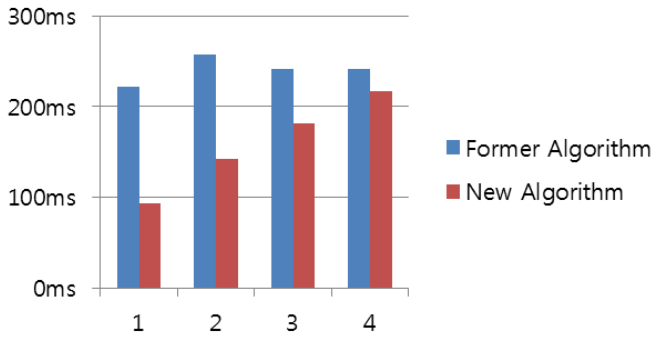

그림 165 소요 시간 비교

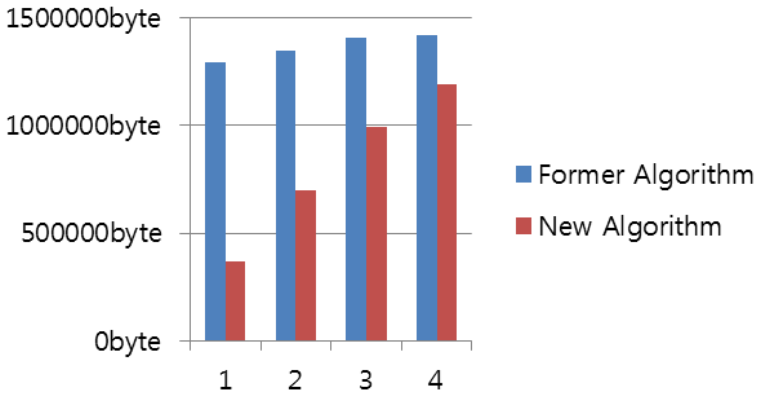

그림 166 요청 데이터 비교

그림 165와 그림 166은 표 30에 대한 실험 결과 그래프이다. 그림 165에서 1번 실험에 X축과 Y축 100Pixel 이동하는 경우에는 우화 전의 소요시간 222ms 이며, 우화 후의 소요시간 93ms이다. 우화 전의 데이터 사이즈는 1290341byte 이며, 우화 후의 데이터 사이즈는 366758byte이다. 2번 실험에 X축과 Y축 200Pixel 이동하는 경우에는 우화 전의 소요시간 258ms이며, 우화 후의 소요시간 142ms이다. 우화 전의 데이터 사이즈는 1347999byte이며, 우화 후의 데이터 사이즈는 697208byte이다. 3번 실험에 X축과 Y축 300Pixel 이동하는 경우에는 우화 전의 소요시간 241ms이며, 우화 후의 소요시간 182ms이다. 우화 전의 데이터 사이즈는 1408094byte이며, 우화 후의 데이터 사이즈는 993872byte이다. 4번 실험에 X축과 Y축 400Pixel 이동하는 경우에는 우화 전의 소요시간 241ms 이며, 우화 후의 소요시간 217ms 이다. 우화 전의 데이터 사이즈는 1418586byte이며, 우화 후의 데이터 사이즈는 1187596byte이다.

그림을 보면 같은 구역에서 지도 이동할 때 새 알고리즘은 옛 알고리즘을 보다 요청 소요시간과 데이터 사이즈는 적다.

VII. 결론

본 논문은 기존의 COST 기반의 센서웹 시스템 구조에서 GIS 서비스를 분리하고, 분리후의 센서웹 시스템 구조를 참조하여 구동체웹을 설계 및 구현한다. 그리고 두 시스템의 GIS 서비스 공용하는 특정에 의해 센서웹 및 구동체웹 서비스를 같이 연동하여 지능적인 실내 공간의 상태를 조정하는 통합 서비스를 구현한다.

기존의 COST 기반의 센서웹 시스템은 서비스 콘텐츠 생성하고 서비스 콘텐츠 공급 기능을 분리하여 서비스 공급자의 부담을 줄이지만 센서를 생산하는 회사 입장에서 센서 노드 정보와 지도 정보를 관리하기 어렵다. 그리고 기존의 센서웹 시스템을 이용하여 다른 GIS를 사용하는 시스템과 연동하려면 거의 불 가능하다. 그래서 본 논문에서 기존의 COST 기반의 센서웹 시스템을 고찰하여 참조하여 실내 GIS 시스템이 분리되는 COST 기반의 센서웹 및 Actuator Web 시스템을 설계하여 구현한다. 그리고 3개 시스템을 연동하기 위한 응용서버를 통해 최적 실내 환경을 제어하는 통합 서비스를 설계하여 구현한다. Silverlight 구현 기술을 사용하여 실내 정보를 가시화 기능을 수행하는 웹 응용 클라이언트를 구현한다.

본 논문에서 제시하는 COST 기반의 GIS를 분리 공용하는 시스템 구조를 사용하여 다양한 서비스를 한 지도에서 서비스 제공이 가능하고 센서웹 및 구동체웹의 서비스 공급자 입장에서는 서비스 공급에 대한 부담이 줄어들어 서비스 제공에 전념할 수 있으므로 보다 안정적으로 서비스를 제공할 수 있는 환경이 제공된다. 클라이언트가 다양한 응용서버를 검색하여 다양한 서비스를 받을 수 있다. 또한 미래에 기존의 센서웹이 응용서버를 통해 다른 시스템을 구동하여 인간의 생활의 편리성을 제공할 것으로 전망된다.

참고문헌

[1] http://developer.51cto.com/art/201004/195100.htm

[2] F. Curbera, M.Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL and UDDI", IEEE Internet Computing, Vol. 6, No. 2, pp. 86-93, March 2002.

[3] R. Enns, Ed et al., "네트워크 Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[4] Aiko Pras, Thomas Drevers, Remco van de Meent, and Dick Quartel, "Comparing the Performance of SNMP and Web Services-Based Management", eTransactions on 네트워크 and Service Management, Vol. 1, No. 2, pp. 72-82, Dec. 2004.

[5] 박유미, 문애경, 유현경, 정유철, 김상기, "SOAP 기반웹서비스와 RESTful 웹 서비스 기술비교", 전자통신동향분석, 제25권, 제2호, pp. 112-120, 2010년 4월.

[6] Linnyer Beatrys Ruiz, J. Marcos Nogueira, and Antonio A. F. Loureiro, "MANNA: A Management Architecture for Wireless 센서 네트워크s", IEEE Communications Magazine, Vol. 41, No. 2, pp. 116-125, Feb. 2003.

[7] W. L. Lee, A. Datta, and R. Cardell-Oliver, "WinMS: Wireless 센서 네트워크-management system", CSSE Technical Report, UWA-CSSE-06-001, pp. 1-21, June 2006.

[8] M. Turon, "Mote-view: A 센서 네트워크 monitoring and management tool", In t he Second IEEE Workshop on Embedded 네트워크ed 센서s, pp. 11-18, March 2005.

[9] N. Ramanathan, L. Girod, E. Kohler, and D. Estrin "Sympathy: A Debugging System for 센서 네트워크s", Proceedings of The First IEEE Workshop on Embedded Networked 센서s, pp. 554-555, Nov. 2004.

[10] Samuel Madden, Joe Hellerstein, and Wei Hong, "TinyDB: In-네트워크 Query Processing in TinyOS", IRB-TR-02-014, pp. 1-41. Sep. 2003.

[11] 최재원, 고영탁, 김한경, 이광휘, "웹 인터페이스를 사용하는 SNMP 기반의 무선 센서네트워크 통합관리시스템", 전자공학회논문지, 제46권 TC편, 제10호, pp. 43-49, 2009년10월.

[12] Douglas Mauro and Kevin Schmidt, "Essential SNMP", O'Reilly Media, Inc., Sep. 2005.

[13] "Ergonomics of the thermal environment — Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices

and local thermal comfort criteria" ISO-7730 INTERNATIONAL STANDARD, Nov. 2005

[14] http://baike.baidu.com/view/3787595.htm

[15] http://keisan.casio.jp/has10/SpecExec.cgi?path=01500000.%8A%C2%8B%AB%82%CC%8Cv_%8EZ%2F05000000.%8BC%8F%DB%2F11000300.%91%CC%8A%B4%89%B7%93x%2Fdefault.xml

[16] 김용우, "USN 기반의 구동체 제어 미들웨어 설계 및 구현" 석사학위논문 2008년 12월

[17] 김대영, 김성훈, 하민근, 김태홍, 이요한 "Internet of Things 기술 및 발전 방향" 한국통신학회지(정보와통신) 28(9), 2011.8, 49-57 (9 pages)

[18] 안연준, "센서 웹 기반의 COST 설계 및 구현" 석사학위논문 2011년 8월