碩士學位論文

USN 기반의 구동체 제어 미들웨어 설계 및 구현

濟州大學校 大學院

컴퓨터工學科

金　龍　佑

2008年 12月

USN 기반의 구동체 제어 미들웨어 설계 및 구현

指導敎授 金 度 縣

金 龍 佑

이 論文을 工學 碩士學位 論文으로 提出함

2008年 12月

金珪利의 工學 碩士學位 論文을 認准함

審査委員長 안 기 동 印

委 員 송 왕 철 印

委 員 김 도 현 印

濟州大學校 大學院

2008年 12月

Design and Implementation of Actuator Control Middleware based on USN

Yong-Woo Kim

(Supervised by professor Do-Hyeun Kim)

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

2008. 12.

This thesis has been examined and approved.

Thesis director, KJ

Thesis director, Wang Cheol Song

Thesis director, Do Hyeun

December 2008

Department of Computer Engineering

Graduate School

Cheju National University

감사의 글

제가 대학원에 진학하는 것을 마음먹고 2년 동안 실력을 키우고 졸업하겠다는 다짐을 한 것이 엊그제 같습니다. 참 시간이 빠른 것 같습니다. 학부생을 졸업하고 공부를 더하고 싶은 욕심에 대학원에 진학 했지만 원하는 것을 전부 해내지 못한 것 같아 많이 아쉽습니다. 하지만 훌륭하신 교수님들께 많을 지식을 배웠고 더불어 간접적으로나마 사회생활도 겪을 수 있어서 저에게 큰 도움이 되는 시간이었습니다. 먼저 누구보다도 저를 믿고 뒤에서 뒷바라지 해주신 저의 부모님과 저에게 엄마 같고 애인 같은 누나, 멋지고 하나뿐인 형, 든든한 스승 같은 매형께 감사하다는 말을 전하고 싶습니다. 힘든 농사일을 하면서도 교육에 열정적이셨던 부모님의 사랑과 기대에 조금이나마 보답이 되었으면 좋겠습니다. 아직도 많이 죄송하고 좋은 아들이 되지 못한 듯싶어 후회스럽지만 이제 사회에 나가 그 큰 은혜에 보답할 수 있도록 노력하겠습니다. 아버지, 어머니, 누나, 형, 매형 모두 사랑합니다! 그리고 언제나 저에게 가족처럼 가르침을 주셨던 김도현 교수님, 교수님이 안 계셨다면 지금 저도 없었습니다. 교수님과 함께 많은 시간을 보내면서 힘들 때마다 위로해주시고 격려 해주셔서 감사합니다. 뿐만 아니라 저에게 관심을 가져주셨고 많은 가르침을 주셨던 김장형 교수님, 안기중 교수님, 곽호영 교수님, 이상준 교수님, 변상용 교수님, 송왕철 교수님, 변영철 교수님 정말 감사하고 존경합니다. 학부생활을 하면서부터 교수님들께서 저에게 큰 힘이 되어 주셔서 무사히 대학원 생활까지 마칠 수 있었습니다. 언제나 건강하시고 하시는 일 모두 잘되시길 바랍니다. 그리고 이정하 선생님, 정은경 선생님께 감사합니다. 선생님보다는 저에겐 친 누나처럼 잘해주셔서 제가 재밌는 대학생활을 할 수 있었습니다. 언제나 학생들 뒤에서 힘든 일을 하시지만 누나들의 그 백만 불짜리 미소는 언제나 기억날 거예요. 이 밖에도 대학원 생활을 잘 할 수 있도록 많은 조언을 해주신 김정희, 한경복, 강은철, 권 훈, 노영식, 김성수 선배님들, 대학원 동기인 이철식, 한지광, 양문석, 변지웅 형님을 비롯해 많은 분들께 고맙다는 말을 전합니다. 그리고 빼놓을 수 없는 저희 모바일 컴퓨팅 연구실 멤버들! 덕분에 많이 배우고 즐거운 대학원 생활을 한 것 같다. 절친한 친구 현진규, 강성철, 강경옥, 강문석 언제나 좋은 역만 내가 해서 미안했고 니들이 있어 든든했던 것 같아. '친구'라는 노래, 늙어서도 어깨동무하고 같이 부르자. 그리고 이창영, 라자니, 이영수, 박창홍, 최익준, 부준필, 고민정, 김영작, 김경탁, 강창봉, 황인철, 바야르 모두 고맙고 너희들은 나의 후배이자 친구요, 라이벌이란 것을 잊지 말고 교수님이 항상 말씀하시는 것처럼 모두 잘 되길 빈다. 연구실 생활을 하면서 나에게 가장 큰 보람은 너희들과 만난 것 같다. 연구실 멤버는 아니지만 저에게 많은 조언과 격려를 해주신 벗과 같은 우성이형, 민철이형, 우종이형, 수남이형 모두 고맙습니다. 마지막으로 은정아, 언제나 힘들 때마다 내 옆에서 힘이 되어줘서 고맙고 사랑한다. 모두 고맙습니다.

목 차

그림목차 ····· iii
국문초록 ····· v
영문초록 ····· vi
약어표 ····· vii

Ⅰ. 서 론 ····· 1
1. 연구배경 ····· 1
2. 연구 목적 및 방법 ····· 1
3. 논문 구성 ····· 2

Ⅱ. 관련연구 ····· 3
1. USN 미들웨어 기술 분석 ····· 3
2. 기존 구동체 제어 미들웨어 기술 분석 ····· 6

Ⅲ. USN 기반의 구동체 제어 미들웨어 구조 설계 ····· 9
1. 구동체 제어 미들웨어 구성 ····· 9
2. 데이터 처리 모듈 설계 ····· 11
2.1. 데이터 처리 모듈 구조 ····· 11
2.2. 구동체 제어 미들웨어에 필요한 XML 데이터 ····· 12
3. 구동체 제어 모듈 설계 ····· 14
3.1. 구동체 제어 모듈 구조 ····· 14
3.2. 구동체 제어 모듈에 필요한 기능 설명 ····· 15
3.2.1 센싱 데이터 인터페이스 ····· 17
3.2.2 구동체 스펙 구성 인터페이스 ····· 17
3.2.3 규칙 데이터 설정 인터페이스 ····· 18
3.2.4 제어 메시지 생성 컴포넌트 ····· 18
3.2.5 제어 메시지 관리 컴포넌트 ····· 19

3.2.6 구동체 제어 인터페이스 ····· 19
4. 구동체 제어를 위한 구동체 등록 정보 XML 구조 ····· 20
5. 제어 메시지 생성 과정 ····· 22

Ⅳ. USN 기반의 구동체 제어 미들웨어 구현 ····· 26
1. 구현 환경 및 테스트 환경 ····· 26
2. 구현 결과 ····· 27
3. 성능 분석 ····· 32
4. 평가 및 고찰 ····· 35

Ⅴ. 결론 ····· 36
참고문헌 ····· 37

그 림 목 차

그림 1. USN 미들웨어 개념 모델 ····· 3
그림 2. USN 미들웨어의 기본적인 시스템 구성 ····· 5
그림 3. USN 미들웨어에서 지원 가능한 질의 유형 ····· 6
그림 4. SANET 시스템 구조 ····· 6
그림 5. Sentire 프레임워크 데이터 흐름도 ····· 7
그림 6. Sentire 미들웨어의 쿼리 패킷과 데이터 패킷의 구조 ····· 8
그림 7. 구동체 제어 미들웨어 및 관련 시스템 구성도 ····· 9
그림 8. 구동체 제어 미들웨어의 모듈 구조도 ····· 10
그림 9. 데이터 처리 모듈의 구조도 ····· 11
그림 10. XML 데이터 파싱 및 환경 변수 저장 과정 ····· 13
그림 11. 데이터 분배 모듈의 센싱 데이터 처리 과정 ····· 13
그림 12. 구동체 제어 모듈의 구조도 ····· 14
그림 13. 구동체 제어 모듈의 데이터 흐름도 ····· 17
그림 14. 제어 메시지 패킷 구조 ····· 19
그림 15. 구동체 제어 모듈의 규칙 데이터 XML 구조 ····· 20
그림 16. 조건 질의와 조건 정보의 예 ····· 21
그림 17. 제어 메시지 생성 컴포넌트의 조건 검사와 규칙 연산 ····· 23
그림 18. 조건 정보의 규칙 연산 알고리즘 ····· 25
그림 19. 구동체 제어 미들웨어에서 사용하는 센서 하드웨어 ····· 26
그림 20. 구동체 제어 시스템의 테스트 환경 ····· 27
그림 21. 구동체 제어 응용의 규칙 데이터 등록 화면 ····· 28
그림 22. 규칙 데이터 XML 파일 ····· 29
그림 23. 구동체 제어 미들웨어 구현 화면 ····· 30
그림 24. 센싱 데이터 수집 미들웨어에서 수신한 센싱 데이터 로그 파일 ····· 33
그림 25. 메시지 큐에 저장되는 최종 처리 후 센싱 데이터 로그 파일 ····· 33
그림 26. 메시지 큐의 센싱 데이터 및 제어 메시지 목록 ····· 33

그림 27. 구동체 제어 미들웨어의 제어 메시지 처리 속도 그래프 ····· 33
그림 28. 5개의 구동체를 제어하는 경우 제어 메시지 처리 속도 그래프 ····· 33
그림 29. 구동체 제어 미들웨어의 평균 데이터 처리 속도 그래프 ····· 34

국문초록

USN 기반의 구동체 제어 미들웨어 설계 및 구현

컴퓨터공학과 김용우
지 도 교 수 김도현

현재 USN 기반의 미들웨어 분야를 보면 다양한 서비스 응용을 위해서 상황 인식 기능 및 구동체 제어 기능 등의 확장된 기능을 제공하는 미들웨어 개발 및 연구가 진행되고 있다. 이에 해외에서 센서와 구동체를 네트워크로 연결한 SANET 시스템에서 구동체를 제어하기 위하여 Sentire 미들웨어와 같은 구동체 제어 USN 미들웨어가 개발 되고 있다. 본 논문에서는 구동체 제어를 위한 USN 미들웨어가 제공하는 필요 기술에 대하여 연구하고 센싱 데이터를 분석하여 관리자가 등록한 규칙 데이터에 따라 제어 메시지를 생성하는 구동체 제어 미들웨어를 설계하고 구현한다. 이를 위해 제시한 USN 기반의 구동체 제어 미들웨어는 기존의 SANET 시스템의 Sentire 미들웨어에서 제공하는 기능을 포함하면서 사용자가 제어 메시지를 명령하지 않아도 등록된 규칙데이터에 따라 제어 메시지를 생성하여 처리하는 기능을 추가한다. 이를 통해 센싱 데이터를 처리하고 사용자의 요구에 따라 제어 메시지를 생성할 수 있는 방법을 제시함으로써 기존의 데이터 처리 중심 USN 미들웨어의 기능을 확장하고 구동체가 포함된 새로운 서비스 응용 개발에 도움이 될 것으로 사료된다.

ABSTRACT

Design and Implementation of Actuator Control Middleware Based on USN

KIM, Yong-Woo
Department of Computer Engineering
Graduate School
Cheju National University

Recently, In the field of middleware based on USN, research and development for middleware have been focused on context-aware module and actuator control for variety service application. An USN middleware including the actuator control has been developed in foreign countries as Sentire middleware connecting sensors and actuators through networks. In this thesis, we study the technical skills needed of USN middleware to control actuator and analyze sensing data. Further, proposed actuator control middleware creates control message as registered conditional information automatical. In this research proposed, actuator control middleware supports a comprised function of Sentire middleware on SANET system. A main function of proposed middleware is automatically created control message instead of command control message. Therefore, it supports a way to process sensing data, and create control message depending on the user. So, this thesis can help extending on the existing data processing USN middleware, and the new service development of applications.

약어표

USN	Ubiquitous Sensor Network
SANET	Sensor and actuator networks
QoI	Quality of information management
XML	Extensible Markup Language
DOM	Document Object Model
IM	Interface Manager
DM	Data Manager
RM	Resource Manager
SM	Sensor Manager
AM	Actuator Manager
WSN	Wireless Sensor Network
TCP/IP	Transmission Control Protocol/Internet Protocol
MSMQ	Microsoft Message Queue

Ⅰ. 서 론

1. 연구 배경

현재 유비쿼터스 기술의 발전에 더불어 USN 기반의 많은 서비스 및 응용들이 활발히 개발되고 있다. 국내에서는 이러한 유비쿼터스 컴퓨팅 기술을 미래 도시 환경의 구축에 적용하기 위한 u-City 사업이 구체적으로 추진되고 있다. 이러한 u-City 사업에 필요한 USN 요소 기술 중에서 이종의 센서 노드 H/W 인프라, 이종의 네트워크 환경, 그리고 다양한 응용 서비스들 간의 독립성을 보장하면서도 유연한 통합을 지원하는 USN 미들웨어 기술은 매우 중요한 핵심 기술이다. 지금까지 USN 미들웨어는 오류 없이 빠르고 정확하게 데이터를 필터링, 저장, 분산하는 데이터 처리 중심의 기술 중심으로 연구 및 개발되었으나, 앞으로 USN 미들웨어 기술은 수집되는 USN 데이터를 이용하여 상황 인지 및 행위제안, 나아가 상황에 따라 구동체(Actuator)와 연계되어 위급상황 시 대피유도, 구조 행위를 할 수 있는 기능이 포함된 USN 미들웨어로 발전해야 한다. 이미 해외에서 SANET 시스템의 Sentire 미들웨어와 같이 구동체를 제어 할 수 있는 미들웨어에 대한 연구가 진행되고 개발된 사례가 있다. 이에 본 논문에서는 구동체 제어를 위해 Sentire 미들웨어를 분석하여 필요한 USN 미들웨어 기능을 도출하고 제안하는 구동체 제어 모듈을 통해 관리자가 설정한 규칙 데이터에 따라 제어 메시지를 생성하는 USN 기반의 구동체 제어 미들웨어를 설계하고 구현한다.

2. 연구 목적 및 방법

본 논문에서는 기존 USN 기반의 데이터 처리 중심의 미들웨어 기능과 더불어 수집된 센싱 데이터를 사용자가 등록한 규칙 데이터와 비교·분석하여 제어 메시지를 생성하는 기능이 추가된 USN 미들웨어 모델을 제시한다. 본 논문에서 제

시하는 구동체 제어 미들웨어는 구동체를 제어하기 위해 필요한 컴포넌트에 대해 기존 관련 연구를 통해 분석하고, 다양한 구동체의 제어를 유연하게 적용하기 위한 방안을 제시한다. 이를 위해 구동체 제어 미들웨어의 핵심 모듈로 수집된 센싱 데이터를 처리하는 데이터 처리 모듈과 제어 메시지를 생성하는 구동체 제어 미들웨어의 프로토타입을 제시하고 구동체 제어 모듈을 설계하고 구현 한다.

3. 논문 구성

서론에 이어 2장 관련 연구에서는 기존의 USN 미들웨어의 주요 기능과 관련 요구 기술을 고찰하고, 기존에 연구된 구동체 제어 관련 미들웨어에 대해 알아본다. 3장에서는 본 논문에서 제시하는 USN 기반의 구동체 제어 미들웨어의 전체적인 시스템 구성과 핵심 모듈인 데이터 처리 모듈과 구동체 제어 모듈에 대해 설명한다. 이어서 4장에서는 센싱 데이터 수집 미들웨어 및 구동체 제어 미들웨어를 구현하고, 수집된 데이터의 전송 시간과 제어 메시지 전송 시간 비교를 통해 본 논문에서 제시한 구동체 제어 미들웨어의 성능을 평가하고 고찰한다. 끝으로 5장에서는 결론을 맺는다.

Ⅱ. 관련 연구

본 장에서는 USN 기반의 구동체 제어 미들웨어에 필요한 기능에 대해 연구된 USN 미들웨어의 기술 분석을 통해 고찰한다. 또한 기존의 구동체 제어 USN 미들웨어인 SANET 시스템의 Sentire 미들웨어와 본 논문에서 제시하는 미들웨어를 비교 분석하여 구동체를 제어하기 위해 필요한 기능에 대해 알아본다.

1. USN 미들웨어 기술 분석

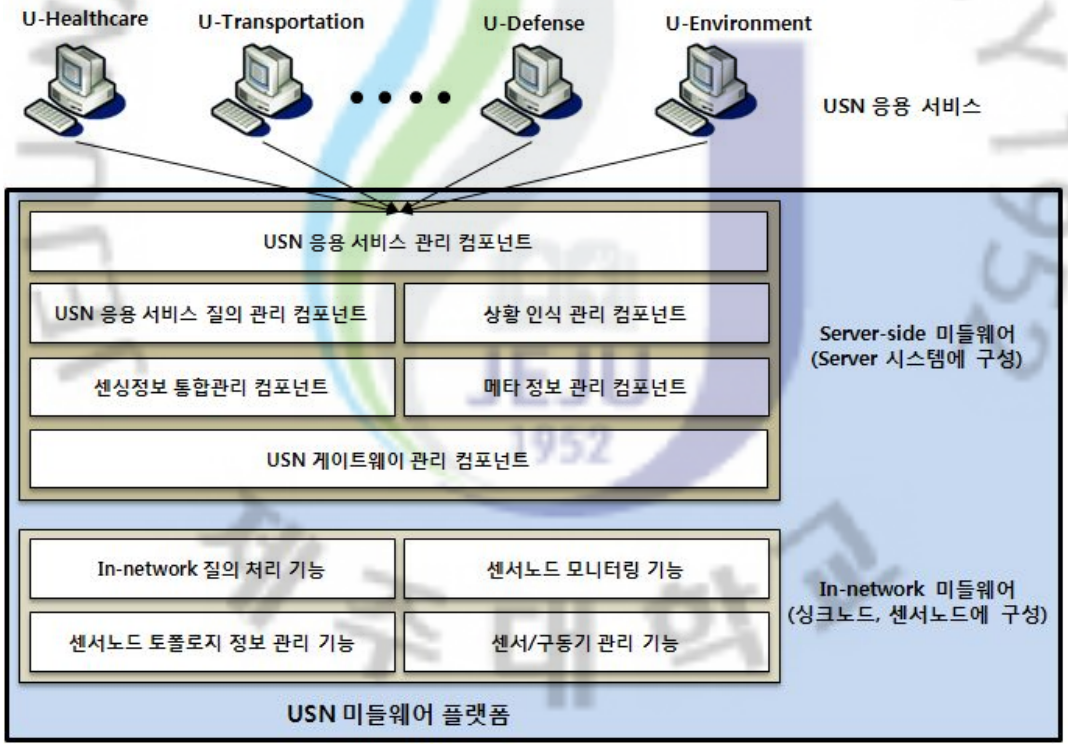

그림 1. USN 미들웨어 개념 모델[2]

USN 미들웨어는 물리적으로 응용 서비스와 센서 네트워크 인프라 중간에 위치하며, 응용 서비스를 효율적으로 지원하기 위하여 서버 시스템에 위치하는

Server-side 미들웨어와 센서 네트워크 내부에서 센싱 정보를 효과적으로 관리하기 위하여 센서 노드 및 싱크 노드들에 위치하는 In-network 미들웨어로 구성된다. 일반적으로 협의적 의미의 USN 미들웨어는 Server-side 미들웨어를 의미하며, 광의적 의미의 USN 미들웨어는 In-network 미들웨어를 포함하는 것을 의미한다. 이러한 광의적 의미의 USN 미들웨어의 핵심 기술은 다음과 같이 분류 될 수 있다. 첫째는 USN 응용 서비스에서 제공되는 다양한 형태의 다중 질의들에 대한 분석 및 최적화 기능이고 둘째는 센서 네트워크로부터 끊임없이 주어지는 센싱 데이터의 획득, 가공, 저장, 분석 및 관리 기능, 셋째는 센싱 데이터 마이닝을 통한 상황 정보 생성 및 관리 기능, 마지막 넷째는 센서 네트워크에 관한 실시간 모니터링 및 동적/정적 메타 정보 관리 기능이다[1] 또한 최근의 고급 USN 응용 서비스들은 센싱 정보를 수집하여 단순히 제공하는 수준이 아니라, 이들 정보들을 분석하고 마이닝하여 새로운 상황 정보를 생성할 수 있는 기능을 반드시 필요로 하고 있다. 예를 들어, u-Fire 와 u-Transportation 서비스는 온도, 습도, 자기장, 소음 등의 다양한 센싱 정보를 획득하고, 이들 정보들을 통합하여 분석하고, 미리 정의된 규칙을 이용하여 최종적으로 화재 및 교통사고 발생을 판단할 수 있는 기능을 필요로 하고 있다. 그러므로 USN 미들웨어는 상황 판단을 위하여 요구되는 센싱 데이터를 통합 분석하기 위한 기능과 규칙을 정의하여 새로운 상황 정보를 생성할 수 있는 기능들을 제공해야 한다[2]

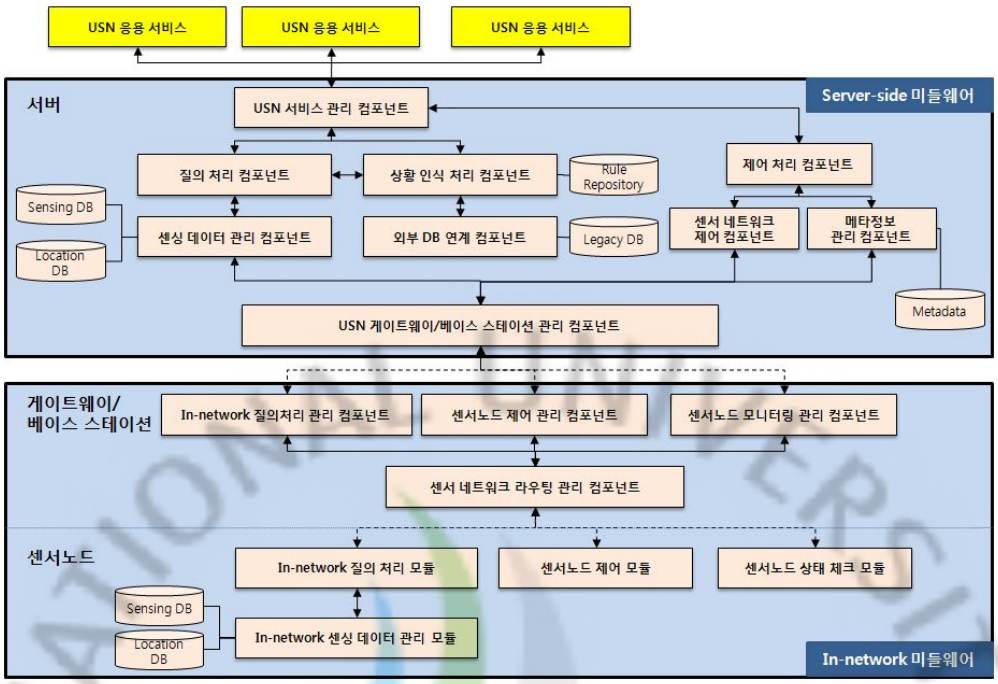

그림 2. USN 미들웨어의 기본적인 시스템 구성[2]

그림 2는 USN 미들웨어에서 필요한 컴포넌트와 모듈을 광범위하게 설명한다. 그림에서 보여주는 기능 중에 질의 처리 컴포넌트와 상황인식 컴포넌트, Rule Repository는 제어 메시지를 생성하기 위한 방법으로 사용될 수 있다. 질의 처리 컴포넌트는 USN 응용 서비스에서 요구하는 질의 정보를 나타낸다. 이는 사용자가 구동체가 동작하는 규칙 데이터를 정의하여 Rule Repository에 저장함을 의미한다. 저장된 규칙 데이터를 이용하여 수집되는 센싱 데이터와 비교하고 상황을 판단하는 상황 인식 처리 컴포넌트를 제공함으로써 구동체를 제어 하게 된다. 사용하는 규칙 데이터의 형식은 USN 미들웨어의 질의 유형 중에서 조건 질의(Conditional Query)를 이용한다. 조건 질의 형태의 규칙 데이터를 생성하는 이유는 센싱 데이터에 대한 속성 및 범위를 지정하기 편하기 때문이며 지속적 질의(Continuous Query)나 일시적 질의(Spatiotemporal Query)처럼 시간과 공간에 대해 규칙 데이터를 생성할 필요가 없기 때문이다. 아래 그림 3은 USN 미들웨어에서 사용하는 질의 유형을 나타내는 그림이다. 조건 질의를 보면 우리가 일반적으로 사용하는 조건문이 들어간 쿼리 형태를 알 수 있다. 조건 질의 형태를 변형하여 정의한 규칙 데이터 형식에 대해서는 뒤에 살펴본다.

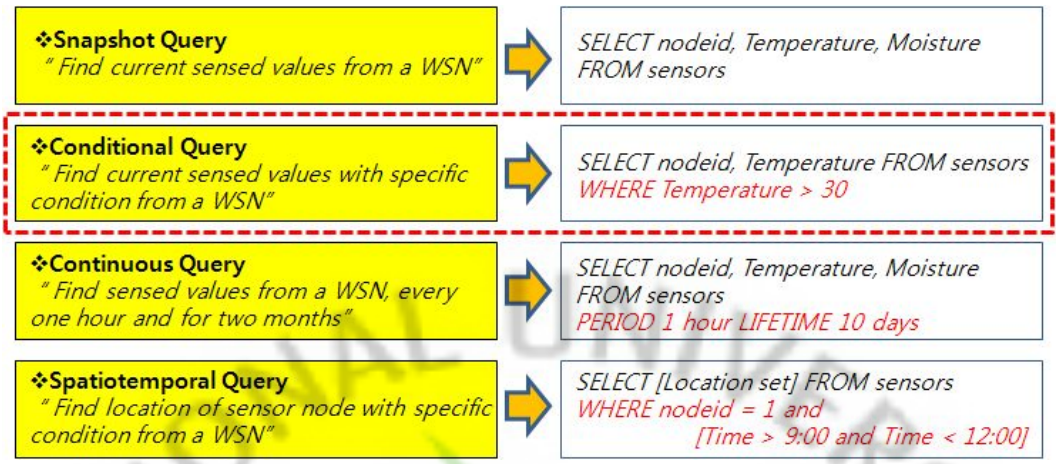

그림 3. USN 미들웨어에서 지원 가능한 질의 유형[2]

2. 기존 구동체 제어 미들웨어 기술 분석

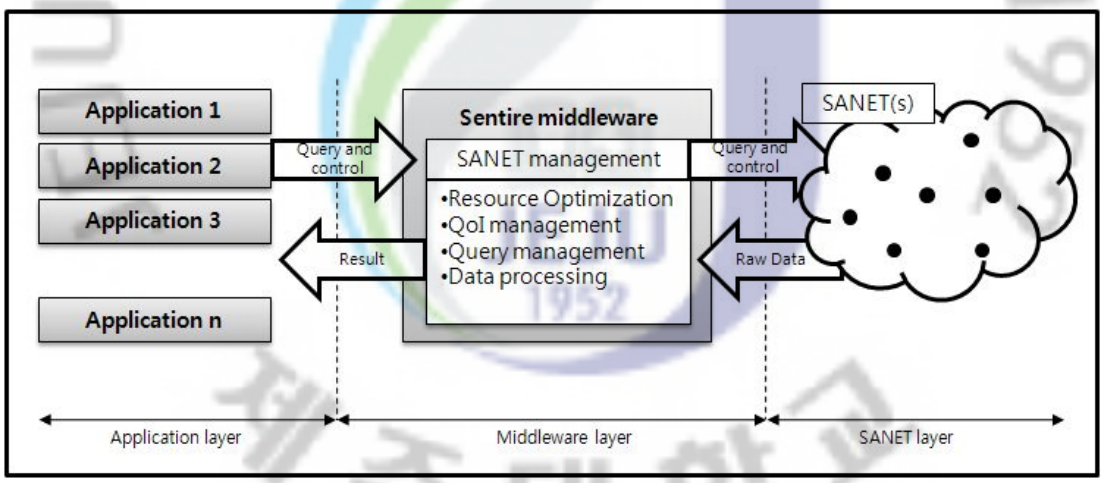

그림 4. SANET 시스템 구조[3]

그림 4는 센서와 구동체간의 네트워크인 SANET과 센싱 데이터 수집과 구동체 제어를 담당하는 미들웨어인 Sentire 미들웨어의 시스템 구조를 나타내고 있다. 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(Sensor data processing), 규정된 수준 이상의 데이터의 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 고려하고 있다[3] 이와 더불어 SANET 시스템의 미들웨어는 일반적인 컴퓨팅 패러다임인 상호 작동 가능성(Interoperability), 재활용성(Reusability), 확장성(Extensibility), 적응성(Adaptability)을 모두 가지고 있다[4]

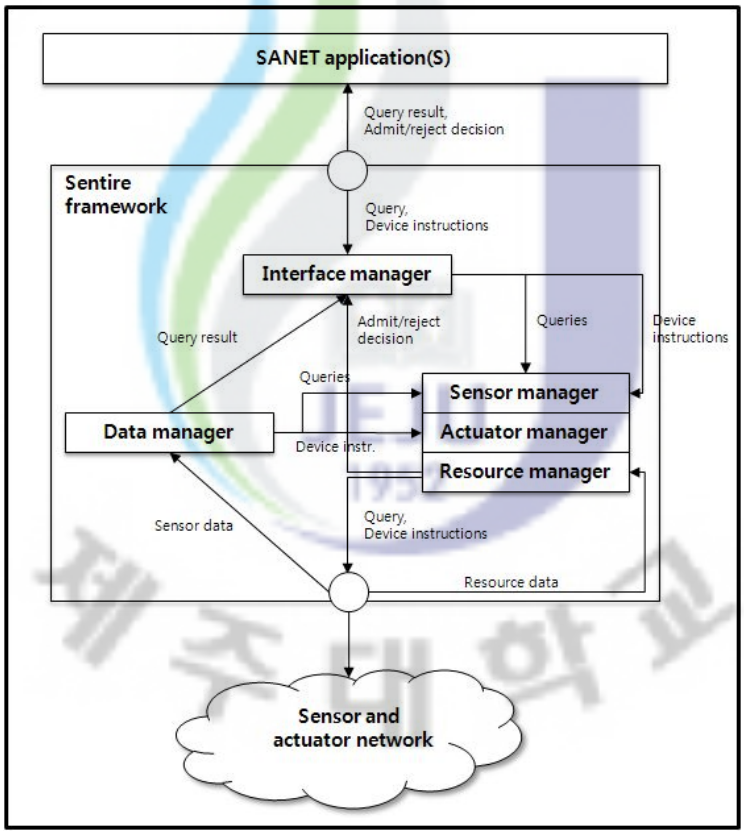

그림 5. Sentire 프레임워크 데이터 흐름도[3]

그림 5는 센서와 구동체 네트워크와 SANET 응용 사이에서 Sentire 프레임워

크의 주요 관리 기능과 데이터의 흐름을 나타낸다. SANET 응용과의 미들웨어의 연결을 담당하는 인터페이스 관리자(IM : Interface manager), 개발자가 정해 놓은 루틴으로 데이터 처리 기능을 수행하는 데이터 관리자(DM : Data manager), 센서와 Actuator의 자원 관리를 담당하는 자원 관리자(RM : Resourse manager), 센서, 구동체들의 구성과 동작 방식을 제어하는 센서 관리자(SM : Sensor manager)와 구동체 관리자(AM : Actuator manager) 컴포넌트는 Sentire 프레임워크에서 핵심적인 역할을 한다. 이중 구동체 관리자는 SANET 응용이나 데이터의 관리자로부터 보내지는 장치 명령어(Device Instruction)를 받아 들여 Actuator를 제어한다. Sentire 미들웨어에서 각 컴포넌트 간 데이터 통신을 위해 사용하는 메시지의 헤더에는 그림 6과 같이 효율적인 처리를 위해 priority, time stamp, sending manager 정보가 들어 있다. 또한 SANET 응용에서 보내는 Sentire 질의 메시지와 데이터 메시지는 비슷한 구조를 가지며 모두 표준 방식인 XML 문서 객체 모델(DOM : Document Object Model)[5]를 사용하여 만든다.

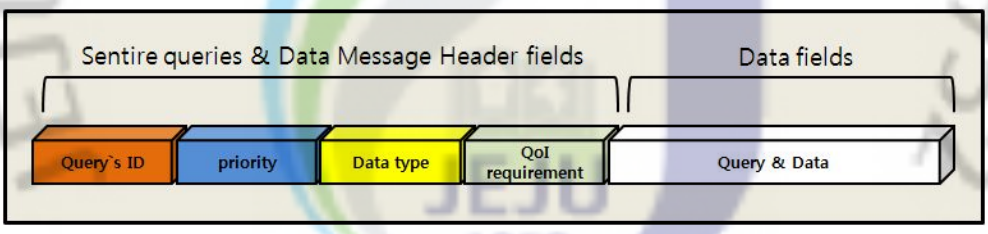

그림 6. Sentire 미들웨어의 질의 패킷과 데이터 패킷의 구조

SANET 시스템에서 동작하는 Sentire 미들웨어는 구동체 제어 방식에서 사용자에 의한 제어 메시지 전송 방식을 이용한다. 이는 화재와 같은 긴급한 상황에서 구동체 제어가 사용자나 응용에 의존하기 때문에 실시간적으로 처리되기 힘들다. 또한 Sentire 미들웨어는 구동체에 대한 확장성을 제공하지 않는다. 즉 제어하고자 하는 구동체를 등록하는 기능이 없기 때문에 새로운 구동체를 추가하여 제어할 방안을 제시하지 않는다. 이에 제안하는 구동체 제어 미들웨어는 자동화된 제어 메시지 생성 기능과 새로운 구동체에 대한 확장성을 보장하고자 구동체 등록 기능을 추가한다.

Ⅲ. USN 기반의 구동체 제어 미들웨어 구조 설계

본 장에서는 USN 기반의 구동체 제어 미들웨어의 전체적인 시스템 구성과 핵심 모듈인 데이터 처리 모듈과 구동체 제어 모듈에 대해 설명하고, 사용자 응용에서 구동체를 제어하기 위해 규칙 데이터인 조건 정보와 규칙 연산 정보를 등록하는 방법에 대해 설명한다. 또한 규칙 데이터를 이용하여 제어 메시지를 생성하는 과정에 대해 기술한다.

1. 구동체 제어 미들웨어 구성

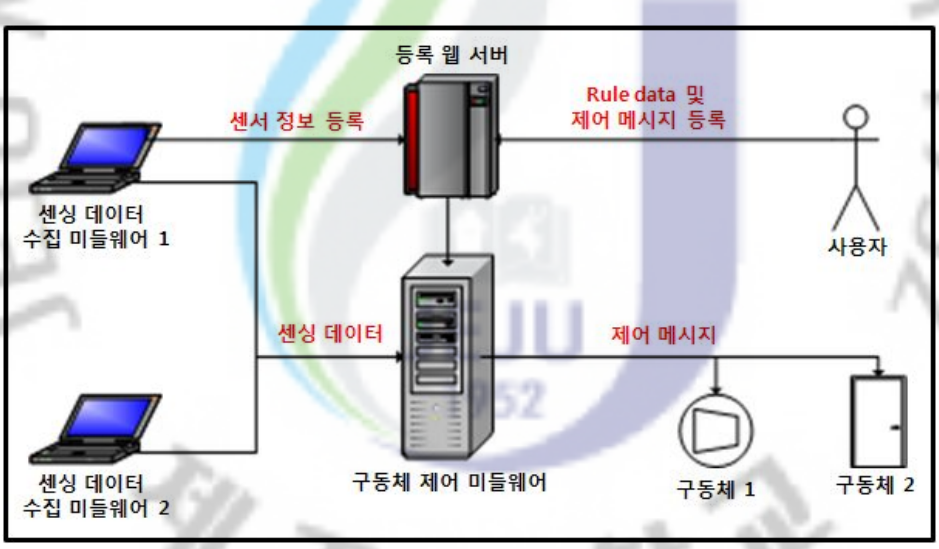

그림 7. 구동체 제어 미들웨어 및 관련 시스템 구성도

제안하는 구동체 제어 미들웨어는 그림 7의 시스템 구성도와 같이 구동체를 제어하는 시스템에서 핵심적인 역할을 한다. 구동체 제어 미들웨어는 등록된 구동체로 제어 메시지를 전송하고 관리하는 기능을 갖는다. 이를 위해 사용자는 제어 메시지 생성에 필요한 규칙 데이터를 구동체 제어 미들웨어에 설정해주어야

한다. 이 기능을 담당하는 등록 웹 서버는 WSN(Wireless Sensor Network)에서 동작하는 센싱 데이터 수집 미들웨어의 센서 하드웨어 정보를 사용자에게 제공함으로써 원하는 위치의 센서가 수집하는 데이터에 대하여 규칙 데이터를 생성할 수 있도록 한다. 구동체 제어 미들웨어는 수집되는 센싱 데이터를 등록된 규칙 데이터 중에서 조건 정보와 비교하고, 조건 정보간의 논리 연산인 규칙 연산 과정을 거쳐 구동체 제어 응용 관리자가 요구한 제어 메시지를 생성하고 전송한다. 제안된 구동체 제어 미들웨어의 핵심 모듈로는 센싱 데이터 처리와 분배를 담당하는 데이터 처리 모듈(Data Process Module)과 제어 메시지를 생성하는 구동체 제어 모듈(Actuator Control Module)이 있으며, 센싱 데이터 인터페이스로는 센싱 데이터 수집 미들웨어에서 데이터를 수집하기 위한 데이터 수집 인터페이스(Data Collection Interface)와 서비스 응용으로 센싱 데이터를 제공하기 위한 사용자 인터페이스(User App Interface)가 있다. 그림 8은 구동체 제어 미들웨어의 모듈 구조도이다.

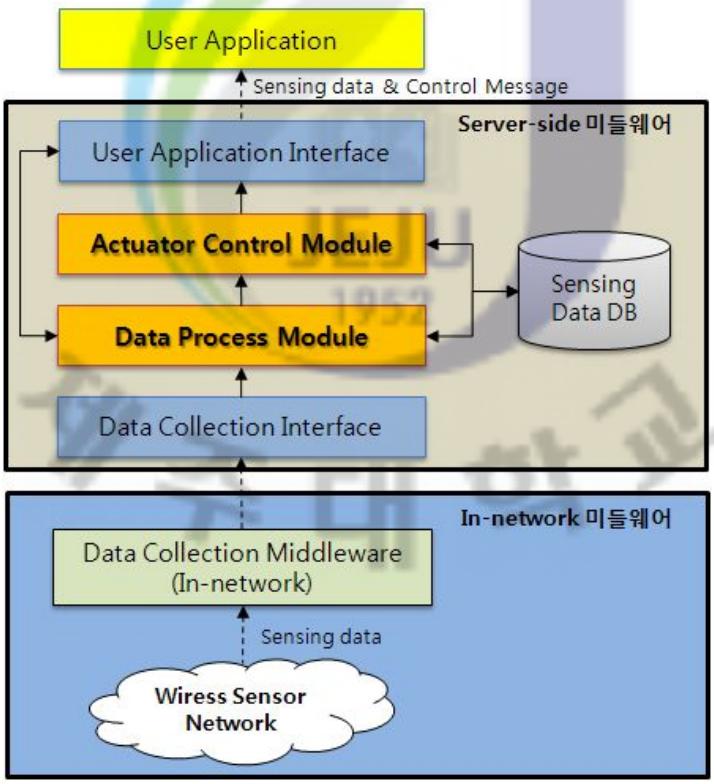

그림 8. 구동체 제어 미들웨어의 모듈 구조도

2. 데이터 처리 모듈 설계

2.1 데이터 처리 모듈 구조

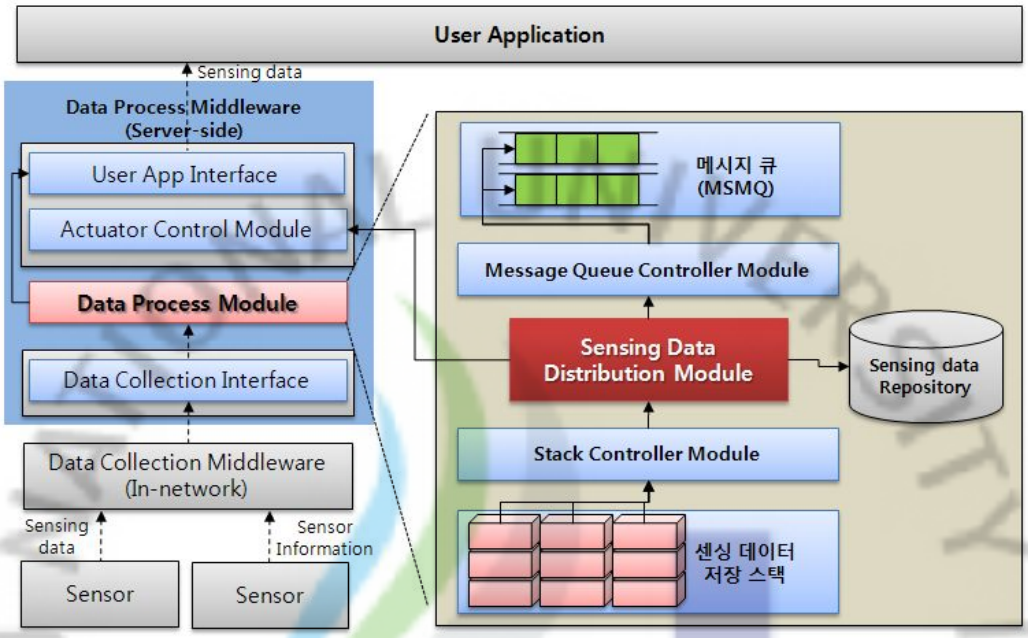

그림 9. 데이터 처리 모듈의 구조도

데이터 처리 모듈은 위의 그림 9와 같은 구조를 갖는다. 수집된 데이터는 데이터 수집 미들웨어에게 하나씩 할당되는 센싱 데이터 저장 스택에 보관된다. 스택을 사용하면 실시간 센싱 데이터를 최우선적으로 처리할 수 있으며, 생성되는 스택은 센싱 데이터 수집 미들웨어 정보 등록 웹 서버의 결과물인 XML 정보를 파싱하여 센싱 데이터 수집 미들웨어의 개 수 만큼 미리 생성하게 된다. 데이터 분배 모듈에서는 병렬 처리를 위해 멀티 스레드 작업으로 각각의 스택에서 센싱 데이터를 하나씩 꺼내어 구동체 제어 응용의 데이터 처리 요구 정보에 따라서 구동체 제어 모듈이나 메시지 큐 컨트롤러로 분배된다. 메시지 큐 컨트롤러는 자신이 관리하는 메시지 큐들 중에 처리된 최종 센싱 데이터가 저장되어야할 메시지 큐를 찾아내어 저장하는 역할을 한다. 메시지 큐에 저장된 데이터는 순서적으로 웹 서비스를 통해 사용자 응용에게 제공된다. 여기서 중요한 점은 항상 실시간 센싱 데이터를 제공하기 위해 가장 나중에 저장된 센싱 데이터를 남기고, 나

머지 센싱 데이터는 데이터베이스 저장 모듈로 하여금 데이터베이스에 백업 데이터로 저장된다. 즉, 데이터베이스 저장 모듈은 메시지 큐를 감시하면서 가장 최신 데이터를 남기고, 나머지만 큐로부터 데이터를 읽어 저장하는 것이다. 데이터 처리 모듈은 다수의 센싱 데이터 수집 미들웨어로부터의 데이터 수집, 구동체 제어 응용에 요구에 맞는 데이터 처리, 정확한 데이터 분배를 주 기능으로 동작한다.

2.2 구동체 제어 미들웨어에 필요한 XML 데이터

본 절에서는 등록 웹 서버를 통해 생성된 센싱 데이터 수집 미들웨어 정보와 사용자 응용의 요구 정보를 저장한 두 개의 XML 데이터를 데이터 처리 모듈에서 사용하는 방법에 대해 설명한다. 구동체 제어 미들웨어에서 XML 데이터를 사용하는 이유는 XML은 플랫폼에 독립적이고 이 기종 간 정보교환이 가능하며 논리적 의미를 부여한 정보의 다양한 구조를 정의하는데 적합하기 때문이다. 이는 구동체 제어 미들웨어의 독립성과 확장성을 위해 필요하다. 데이터 처리 모듈은 구동체 제어 미들웨어의 동작에 필요한 두 종류의 XML 데이터를 분석하여 자신이 생성할 스택과 메시지 큐의 개 수, 데이터를 수집하는 방법, 데이터 처리 요구 정보, 조건 정보 및 규칙 연산 정보로 이루어진 규칙 데이터를 먼저 파악해야한다. 그림 10은 이러한 XML 파싱 작업 과정을 설명한다. 두 개의 XML 데이터를 파싱하는 과정을 거쳐 데이터 처리 모듈의 환경 변수 클래스에 저장한다. 센싱 데이터 수집 인터페이스와 센싱 데이터 저장 스택에서는 센싱 데이터 수집 미들웨어의 정보를 사용하고 센싱 데이터 분배 모듈과 메시지 큐 컨트롤러에서는 사용자 응용의 요구정보를 사용한다.

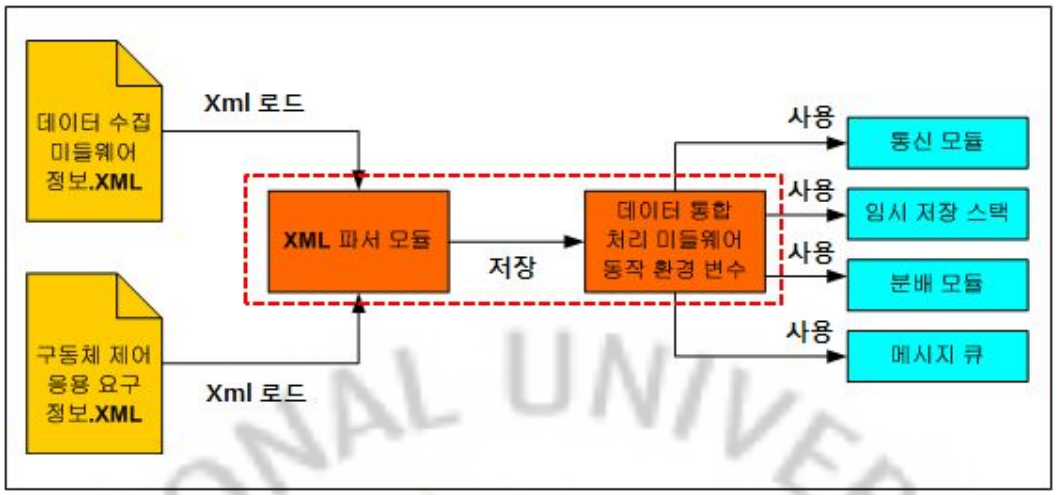

그림 10. XML 데이터 파싱 및 환경 변수 저장과정

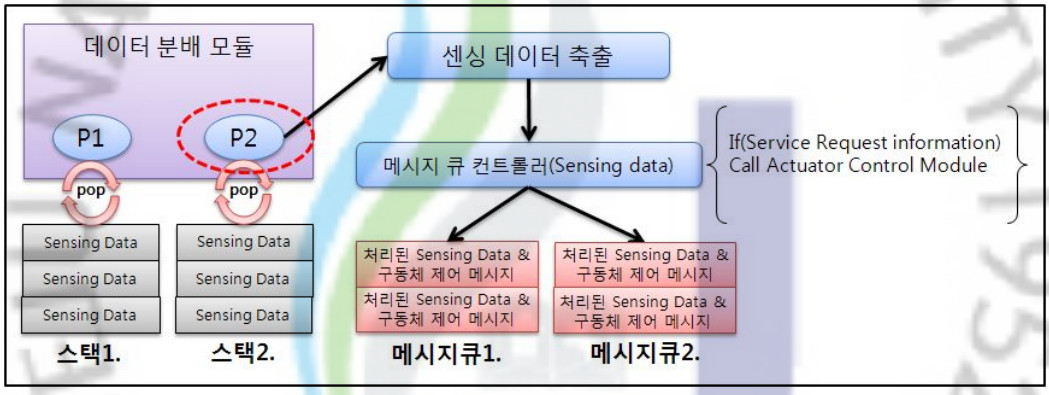

그림 11. 데이터 분배 모듈의 센싱 데이터 처리 과정

데이터 처리 모듈의 데이터 분배 모듈은 데이터 처리 모듈의 가장 핵심적인 부분으로 그림 11과 같이 센싱 데이터 수집 미들웨어 전송된 센싱 데이터를 축출하고 구동체 제어 모듈로 전달하여 사용자 응용이 요구하는 처리 과정을 거쳐 메시지 큐 컨트롤러에 의해 저장된다. 데이터 분배 모듈이 필요한 이유는 다양한 센서 네트워크에서 동작하는 센서 데이터 수집 미들웨어의 다양한 데이터를 효과적으로 처리할 수 있으며, 서비스 응용에 원하는 센싱 데이터를 적절하게 분배하기 위함이다. 그림 11에서 보듯이 P1, P2는 이러한 처리 과정을 담당하는 하나의 프로세스로 데이터 분배 모듈은 센싱 데이터 수집 미들웨어의 개수, 즉 처리할 센싱 데이터 저장 스택의 수만큼 생성되어 하나의 스택을 분배 받고 작업을

처리하게 된다.

3. 구동체 제어 모듈 설계

3.1 구동체 제어 모듈 구조

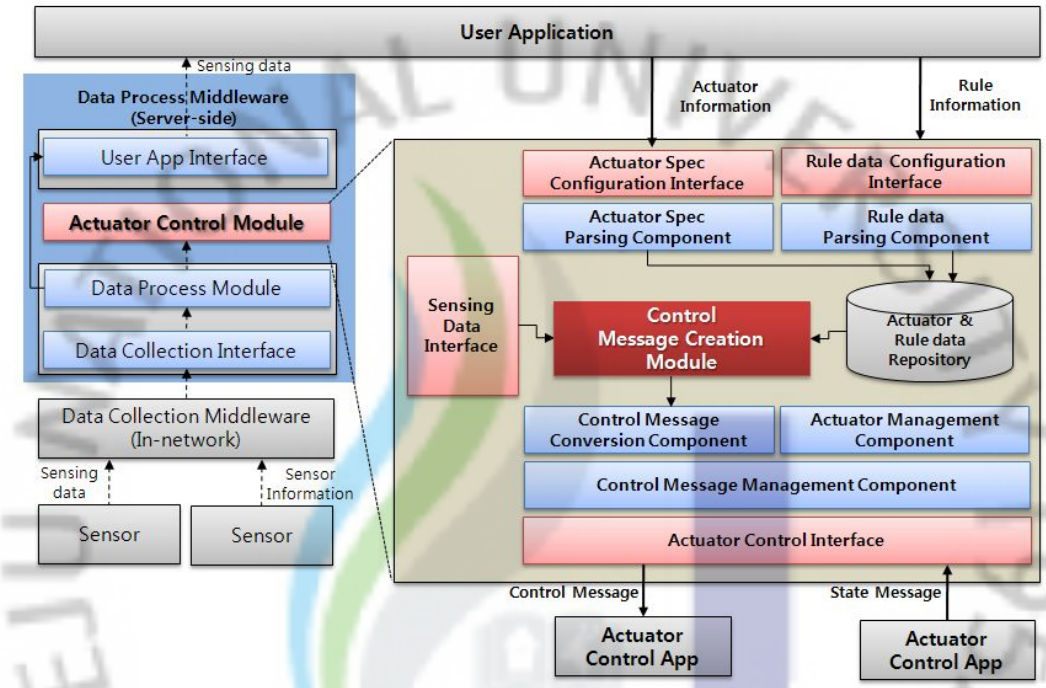

그림 12. 구동체 제어 모듈의 구조도

구동체 제어 미들웨어에서 구동체 제어 모듈은 수집되는 센싱 데이터를 분석하고 등록한 규칙데이터와 비교하여 긴급 상황 혹은 정해진 상황에 따라 등록된 구동체를 동작시키기 위한 제어 메시지를 생성하는 역할을 담당한다. 구동체 제어 모듈의 각 컴포넌트 구조는 그림 12와 같다. 구동체 제어 모듈의 핵심 부분은 제어 메시지 생성 컴포넌트로서, 관리자가 등록한 규칙 데이터(조건정보, 규칙 연산 정보)에 따라 수집되는 센싱 데이터를 비교하여 제어 메시지를 생성하는 부분이다. 이밖에도 구동체 제어 모듈은 생성된 제어 메시지를 해당하는 구동체의 하드웨어 특성에 따라 알맞은 제어 신호나 값으로 변환하는 기능, 구동체의 현재 동작 유무를 판별하는 상태 관리 기능, 구동체 제어 응용과 데이터를 주고받기

위한 인터페이스, 규칙 데이터 설정 인터페이스를 제공한다. 다음 절에서 각 기능에 대해 자세하게 설명한다.

3.2 구동체 제어 모듈에 필요한 기능 설명

제안하는 구동체 제어 미들웨어는 Sentire 미들웨어의 구동체 제어 컴포넌트 기능을 제공한다. 반면 Sentire 미들웨어에서 구동체를 제어하기 위해 응용에서 장치 명령어를 전달해야 하지만 제안하는 구동체 제어 미들웨어는 규칙데이터인 조건 정보들과 규칙 연산에 의해 제어 메시지를 스스로 생성하여 구동체를 제어한다. 또한 다양한 구동체를 제어하기 위해 구동체 스펙 구성 인터페이스를 통해 제어메시지와 통신 방법을 등록받게 된다. 이는 여러 구동체에 유연하게 적용되기 위해 필요한 기능이다. 아래 표 1은 Sentire 미들웨어와 본 논문에서 제안하는 구동체 제어 미들웨어의 컴포넌트 비교와 구동체 제어 방식을 비교한 표이다.

표 1. Sentire 미들웨어와 제안된 구동체 제어 미들웨어의 기능 비교

기 능	Sentire 미들웨어	구동체 제어 미들웨어
응용과의 연결 인터페이스	• Interface manager	• User App Interface • Rule data Configuration Interface • Actuator Spec Configuration Interface
센서 제어	• Sensor manager	• Data Collection Middleware(In-network)
구동체 제어	• Actuator manager	• Actuator Control Module
하드웨어 자원 관리	• Resource manager	• Sensor HW & Actuator HW Repository
구동체 제어 방식	• 쿼리, 장치 명령어를 SANET 응용에서 제공하여 Actuator 제어	• 조건 정보 및 조건들의 규칙 연산에 의한 Actuator 제어 메시지를 통해 Actuator 제어
구동체 등록 방식	• 제공 하지 않음	• Actuator Spec Configuration Interface를 통해 제어 메시지 및 통신 방법을 등록

표 1을 보면 Sentire 미들웨어에서 인터페이스 관리자 기능을 구동체 제어 미

들웨어서는 3개의 인터페이스로 확장하고 있다. 사용자 응용과 데이터를 주고받는 인터페이스 기능을 지원하면서 규칙데이터를 설정하고 구동체의 하드웨어 정보를 구성하는 인터페이스를 추가한다. 또한 센서의 제어 및 데이터 수집을 담당하는 Sensor manager의 기능을 In-network 미들웨어인 센싱 데이터 수집 미들웨어로 담당을 한다. 이렇게 하여 WSN(Wireless Sensor Network)에 대한 관리를 전적으로 센싱 데이터 수집 미들웨어에서 담당하며 구동체 제어 미들웨어는 데이터 처리와 구동체 제어 및 관리 기능을 담당하게 하여 효율적인 분산 처리를 제공할 수 있다. 그리고 Sentire 미들웨어에서 구동체 제어를 위해 SANET 응용을 통해 쿼리나 장치 명령어를 직접 전달하는 반면 구동체 제어 미들웨어는 규칙 데이터와 수집되는 센싱 데이터의 비교 연산을 통해 자동으로 제어 메시지를 생성하는 차이점을 가진다. 또한 구동체 스펙 구성 인터페이스를 통해 통신 방법과 하드웨어 특성을 등록하여 다양한 구동체와 호환할 수 있는 장점을 갖는다. 그림 13은 구동체 제어 응용의 각 컴포넌트간의 데이터 흐름을 나타낸다. 구동체 제어 모듈이 동작하기 위해 우선 규칙 데이터와 구동체 스펙 정보를 구동체 제어 응용의 관리자나 사용자가 등록해주어야 한다. 이 정보들은 모두 데이터베이스와 같은 저장소에 기록된다. 구동체 제어 미들웨어가 동작하여 실시간으로 센싱 데이터를 전송받게 되면 데이터 처리 모듈에서 구동체 제어 모듈의 연결을 담당하는 센싱 데이터 인터페이스를 통해 전달된다. 이 센싱 데이터와 저장소에 저장된 규칙 데이터의 비교 및 규칙 연산을 제어 메시지 생성 컴포넌트가 수행하여 제어 메시지를 생성하고 제어 메시지 변환 컴포넌트로 전송한다. 전송된 제어 메시지는 구동체를 제어하기 위해 등록된 하드웨어 정보에 따라 제어 신호나 값으로 변환되고, 이를 패키징하여 구동체 제어 인터페이스를 통해 구동체 제어 응용으로 전송된다.

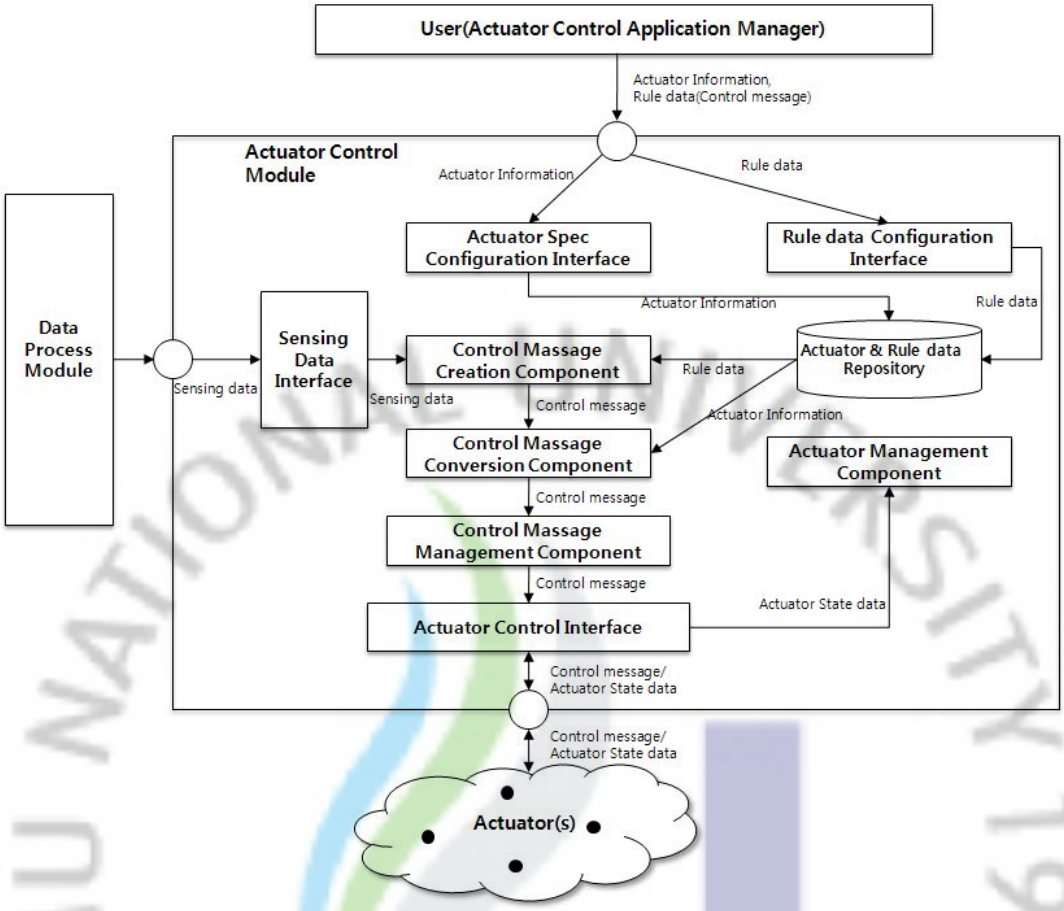

그림 13. 구동체 제어 모듈의 데이터 흐름도

3.2.1. 센싱 데이터 인터페이스(Sensing Data Interface)

센싱 데이터 인터페이스(Sensing Data Interface)는 데이터 처리 모듈과 구동체 제어 모듈간의 센싱 데이터의 흐름을 담당한다. 데이터 처리 모듈에서 제공되는 센싱 데이터를 전달받고 제어 메시지를 생성하기 위해 조건 검사와 규칙 연산을 수행하는 모듈인 제어 메시지 생성 컴포넌트로 전달하여야 한다. 센싱 데이터 인터페이스가 센서 데이터의 전달 기능을 수행하며, 센싱 데이터의 특성상 실시간으로 수집되는 데이터이므로 최신 데이터를 보장하고 흐름제어를 수행하는 중간 버퍼 역할을 하는 스택을 두고 LIFO(Last in first out)방식으로 처리한다.

3.2.2. 구동체 스펙 구성 인터페이스(Actuator Spec Configuration Interface)

구동체 스펙 구성 인터페이스(Actuator Spec Configuration)는 구동체 제어 모듈에서 관리해야하는 구동체의 하드웨어 정보를 등록하는 역할을 수행한다. 하드웨어의 통신 방식, 변환 되어야 할 제어 신호나 값, 동작 방식 등을 구동체 제어 응용의 관리자가 이 인터페이스를 통해 등록함으로써 제어 메시지 생성 컴포넌트와 제어 메시지 변환 컴포넌트 과정에 필요한 정보를 제공한다. 구동체 스펙 구성 인터페이스를 통해 등록된 구동체 정보는 데이터베이스와 같은 Actuator Spec Data Repository에 저장되며 여러 가지 구동체 하드웨어 특성에 따라 확장될 수 있는 기능을 갖추어야 한다.

3.2.3. 규칙 데이터 설정 인터페이스(Rule Data Configuration Interface)

규칙 데이터 설정 인터페이스(Rule Data Configuration Interface)는 구동체 제어 모듈에서 제어 메시지를 생성하기 위한 조건 정보, 규칙 연산 정보를 설정한다. 구동체 제어 응용의 관리자가 정해진 형식으로 구동체를 제어하기 위한 조건 정보와 조건 정보간의 연산을 나타내는 규칙 연산 정보를 설정하면 이 인터페이스를 통해 XML 형식으로 기록된다. 이 정보는 구동체 스펙 구성 인터페이스에서와 같이 데이터베이스와 같은 Rule Data Repository에 저장된다. 구동체 제어 응용 관리자는 언제든지 규칙 데이터를 추가, 수정, 삭제 할 수 있어야 하며 웹을 통해 설정할 수 있도록 한다.

3.2.4. 제어 메시지 생성 컴포넌트(Control Message Creation Component)

제어 메시지 생성 컴포넌트(Control Message Creation Component)가 동작하기 위해 먼저 등록된 규칙 데이터와 데이터 처리 모듈로부터 실시간으로 수집되는 센싱 데이터가 필요하다. 이를 위해 제어 메시지 생성 컴포넌트는 Rule Data Repository에 저장된 규칙 데이터, 즉 조건정보와 규칙연산 정보를 구동체 응용에 따라 검색하여 알맞은 규칙 데이터를 검색한다. 그 후 수집되는 센싱 데이터와 조건에 해당하는지 검사하고, 조건정보의 연산 정보인 규칙 연산 과정을 거쳐 관리자가 등록한 제어 메시지를 생성한다. 조건 정보의 검사와 규칙 연산의 자세한 과정에 대해서는 5절에서 자세히 살펴본다.

3.2.5. 제어 메시지 관리 컴포넌트(Control Message Management Component)

제어 메시지 관리 컴포넌트(Control Message Management Component)는 생성된 제어 메시지를 구동체 제어 응용으로 전송하기 위한 패키징 작업을 수행한다. 패키징 작업은 생성된 제어 메시지를 실제 전송하기 위한 데이터 메시지로 만드는 과정이다. 기존 연구에서는 구동체를 제어하기 위한 메시지 프로토콜에 대한 표준 형식이 없기 때문에 구동체로 전송되는 제어 메시지 패킷은 Sentire 미들웨어의 쿼리 메시지와 비슷한 구조로 헤더부분과 데이터부분으로 정의한다. 헤더부분에는 타겟 구동체 제어 응용의 고유 ID 값 필드, 제어 메시지인지 상태 메시지인지를 구별하기 위한 데이터 타입 정보 필드, 구동체가 동작할 시간을 지정 할 수 있는 Actuating Time 필드, 제어 메시지가 생성된 시간을 나타내는 Massage Creation Time 필드로 구성되며 실제 데이터 부분에는 제어 메시지가 구동체에 따라 변환된 제어 신호 및 값이 포함된다.

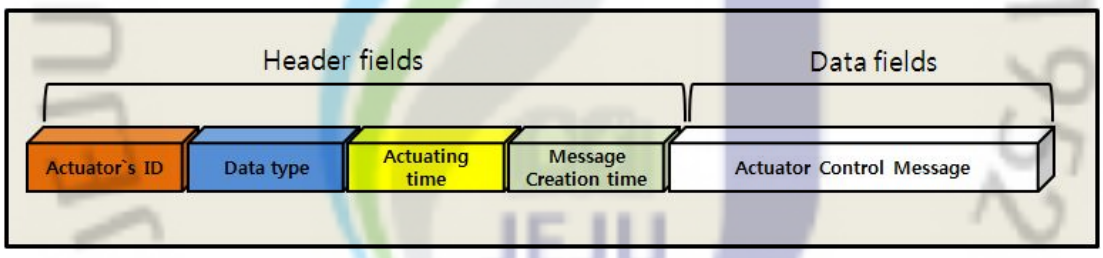

그림 14. 제어 메시지 패킷 구조

3.2.6. 구동체 제어 인터페이스(Actuator Control Interface)

구동체 제어 인터페이스(Actuator Contorl Interface)는 구동체 제어 모듈에서 구동체로 제어 메시지를 전송하는 기능을 담당한다. 상위 모듈에서 패키징된 메시지를 해당하는 구동체로 전송하기 때문에 구동체의 통신 방식에 따라 제어 메시지를 전송해야 한다. 또한 구동체의 응답 신호를 받아서 구동체의 상태 정보를 구동체 관리 컴포넌트로 전송하는 기능도 포함되어야 한다. 이 인터페이스를 웹 서비스를 이용하여 구동체 응용이 데이터를 요청하는 방식을 구현하나 직접적인 TCP/IP 통신이나 여러 가지 통신 방법을 선택할 수 있어야 한다.

4. 구동체 제어를 위한 구동체 등록 정보 XML 구조

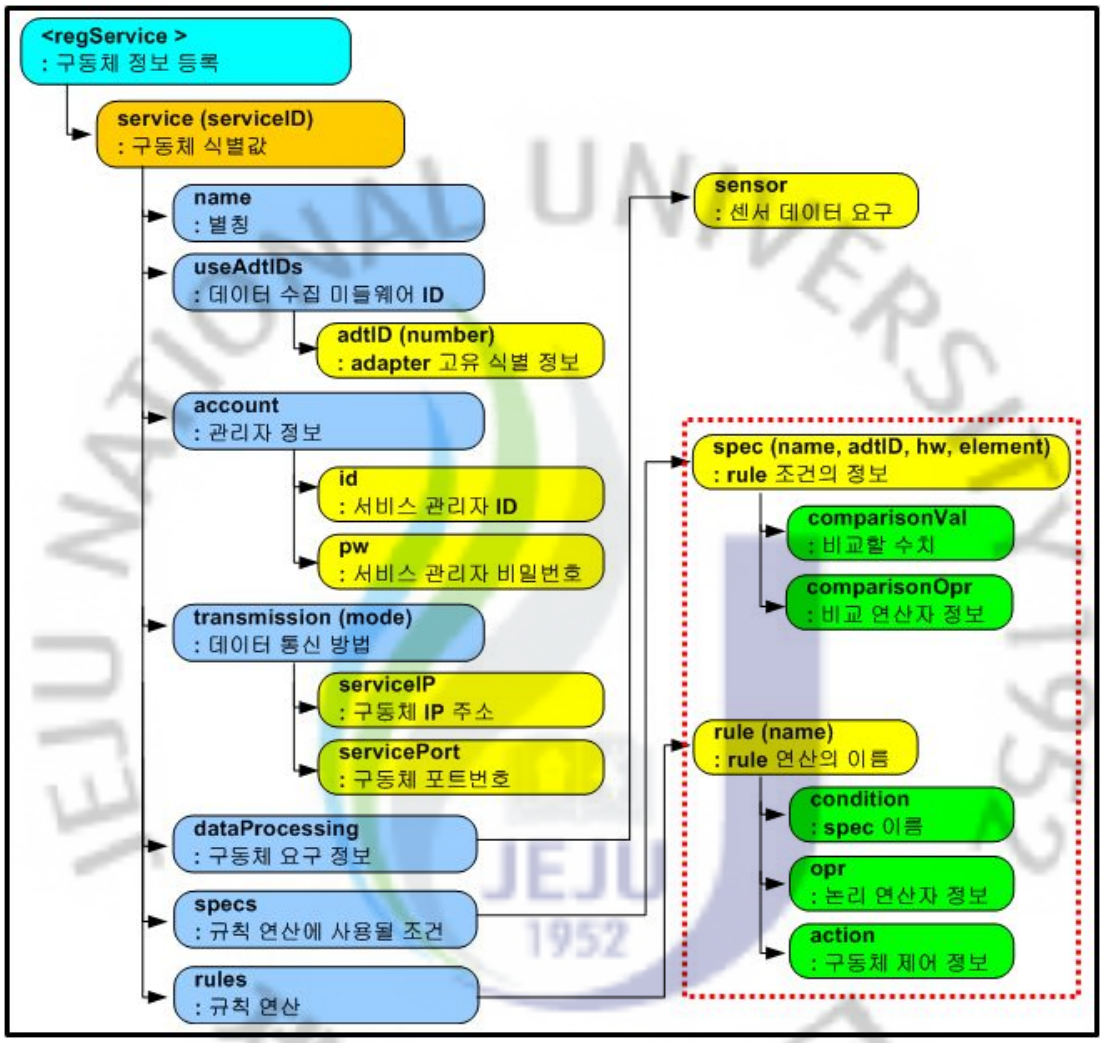

그림 15. 구동체 등록 정보 XML 구조

그림 15는 구동체 등록 정보 XML 구조이다. 이는 제안한 구동체 제어 미들웨어에서 새로운 구동체의 확장성을 위해 제공하는 구동체 스펙 구성 인터페이스와 규칙 데이터 설정 인터페이스의 결과물이다. 그림을 보면 우선 구동체의 식별값, 제어 메시지 생성에 필요한 센싱 데이터를 수집하는 센싱 데이터 수집 미들웨어 ID, 구동체와 구동체 제어 미들웨어 간 통신 방법이 정의된다. 점선으로 표

시된 부분은 구동체 제어 모듈에서 제어 메시지를 생성하기 위한 조건 정보와 규칙 연산 정보를 나타내는 규칙 데이터 XML 구조이다. 먼저 조건 정보 element의 속성들을 살펴보면 <specs> element는 센싱 데이터에 대한 조건 검사를 나타내는 정보로 <spec>이라는 자식 element를 가지며 속성으로 이름, 검사할 데이터 수집 미들웨어 고유 ID, 하드웨어 정보, 속성(온도, 조도, 습도 등)을 가지며, 다시 자식 노드로 비교할 수치 값을 나타내는 <comparisonVal> element와 비교연산자 정보인 <comparisonOpr> element를 가진다. 다음으로 이 조건 정보간의 규칙 연산 정보를 보면 <rules> element가 나타낸다. <rules> element는 규칙 연산의 이름을 나타내는 <rule> element와 그 자식 노드로 <condition>, <opr>, <action> element를 갖는다. 이 조건 정보는 USN 미들웨어의 질의유형 중에서 조건 질의 형태와 비슷하다. 이는 관련연구를 통해 4가지의 쿼리 유형 중에 실시간으로 수집되는 모든 센싱 데이터에 대해 조건 검사를 수행하기 위해 적합한 쿼리 유형이 조건 질의 유형이기 때문이다.

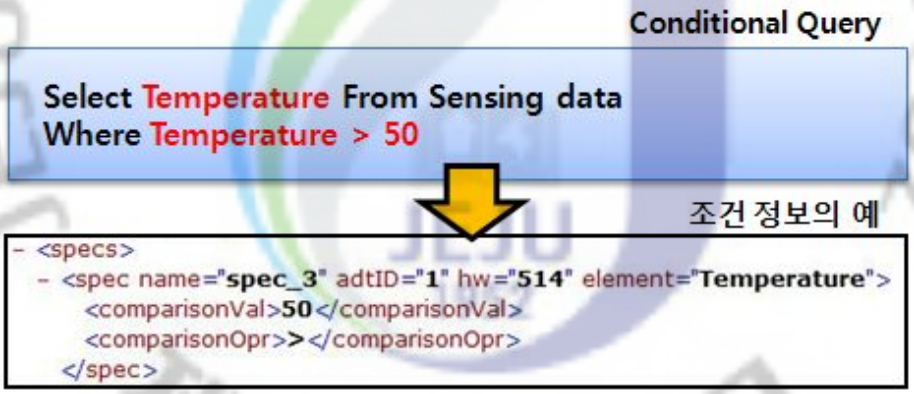

그림 16. 조건 질의와 규칙 데이터(조건 정보)의 예

그림 16을 보면 조건 질의 문을 조건 정보로 변환한 모습을 볼 수 있다. Temperature 값을 조건 정보의 'element' 속성으로 지정하고 조건문 "Where Temperature > 50"을 'comparisonVal', 'comparisonOpr' element로 설정한다. 그리고 센싱 데이터 수집 미들웨어의 고유 ID 값과 센서 하드웨어 번호를 지정함으로써 조건 정보를 완성한다. 규칙 연산의 경우는 위에서 생성한 조건 정보간의 논리 연산을 지정한다. 위의 조건 질의 예에서 만약 "Where Temperature > 50

and Intencity <100″이라는 조건이 있을 경우 조건 정보로 전부 나타낼 수 없기 때문에 조건 정보를 2개 생성(spec1, spec2)하고 “spec1 AND spec2″의 형태로 규칙 연산을 지정 한다.

5. 제어 메시지 생성 과정

구동체 제어 메시지 모듈은 규칙 연산이라는 과정을 통해 제어 메시지 생성 과정을 수행한다. 먼저 규칙 연산을 하기 위해 필요한 조건 정보들은 구동체 제어 응용의 관리자가 구동체 제어 모듈의 규칙 데이터 설정 인터페이스(Rule Data Configuration Interface)를 통해 정의한다. 이 조건 정보들은 센싱 데이터 수집 미들웨어가 등록한 자신의 센서 하드웨어 정보를 토대로 등록한다. 구동체 제어 메시지 모듈은 조건 정보와 규칙 연산을 통해 제어 메시지 삽입 여부를 판단한다. 이 처리 과정은 그림 17과 같이 조건 정보 찾기, 조건 정보 결과 판단, 규칙 연산 찾기, 규칙 연산의 4가지 과정을 거친다. 처음 단계는 구동체 제어 응용에 맞는 조건 정보를 찾아내는 단계로써 이때 구동체 제어 응용의 고유 ID 값을 이용하여 조건 정보 리스트를 생성한다. 조건 정보 리스트가 생성되면 다음 단계로 조건 정보 리스트의 길이 만큼 루프를 돌면서 조건 하나를 검사하면서 True 혹은 False 값을 수집된 센싱 데이터와 비교하여 결정한다. 조건 정보 리스트의 결과 값이 모두 결정 된다면 다음 단계로 구동체 제어 응용의 ID를 이용하여 해당하는 규칙 연산을 찾아낸다. 규칙 연산은 조건과 조건간의 연산이기 때문에 이미 Ture, False 값이 결정된 조건을 AND, OR 연산을 통해 True일 경우 제어 메시지를 추가하고 False 일 경우 Null 값을 추가한다.

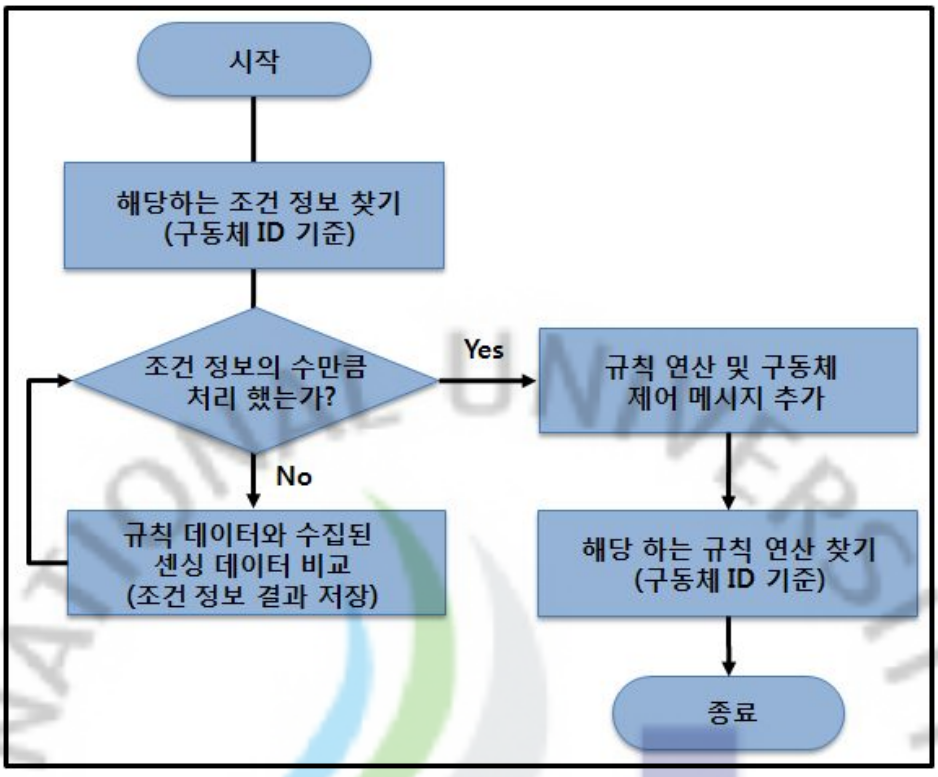

그림 17. 제어 메시지 생성 컴포넌트의 조건 검사와 규칙 연산 과정

그림 18은 구동체 제어 메시지 모듈에서 제어 메시지를 생성하는 규칙 연산의 방법과 과정을 나타내는 순서도이다. 규칙연산에 사용되는 규칙 연산 문자열은 “%조건정보:%조건정보:@연산자:%조건정보:@연산자:#제어 메시지” 형태를 가진다. 여기서 파싱에 필요한 특수문자 ‘%’ 는 조건정보 앞에 사용되며, ‘@’ 는 연산자, ‘#’은 제어 메시지 앞에 사용된다. 이는 조건간의 연산이 원하는 만큼 연산이 가능하도록 정해진 규칙이며 이 규칙연산 문자열을 먼저 파싱하는 것이 규칙연산의 처음 과정이다. 파싱된 문자열은 연산 순서 배열에 들어가며 최종 규칙연산의 결과를 나타내는 규칙 연산 결과 변수와 조건 정보간의 연산 결과를 저장할 조건_1, 2, 3을 선언한다. 조건 연산이 많을 경우에도 3개의 변수를 가지고 처리할 수 있으므로 불필요한 변수는 선언하지 않는다. 전체적인 루프는 파싱된 연산 순서 배열의 길이만큼 진행되며 배열의 위치와 루프의 반복횟수를 결정할 인덱스 1과 인덱스 2 변수를 선언한다. 규칙연산은 기본적으로 이미 그 결과를 처리한 조건 정보간의 AND, OR 연산을 취한다. 먼저 연산 순서 배열의 처음 인덱스

에는 '%'를 가진 조건 정보가 들어 있다. 만약 없다면 이것은 잘못된 규칙 연산 문자열이거나 바로 제어 메시지가 있는 경우이기 때문에 '#' 문자가 있는지 확인하고 제어 메시지 혹은 에러 메시지를 추가한다. '%' 문자일 경우 조건_1에 연산 순서 배열의 첫 인덱스(인덱스 2)의 조건 정보의 결과 값, 즉 True 혹은 False 값을 저장하고 인덱스 2를 하나 증가하여 다음 연산 순서 배열의 정보를 가져온다. 다음 문자열이 '%'를 가지고 있다면 그것은 두 번째 조건정보를 의미하므로 조건_2에 두 번째 조건정보의 결과 값을 저장하고 인덱스 2를 증가한다. 여기서 만약 두 번째가 조건 정보가 아닌 다른 문자열이라면 그 문자열에 '@'가 있는지 판단하지만 조건정보가 하나만 나오고 연산이 이루어지는 경우는 없으므로 제어 메시지에 에러 메시지가 저장된다. 정상적인 규칙 연산 문자열인 경우 다음 연산 순서 배열의 문자열이 '@'를 가지고 있는 연산 정보이며, 인덱스 1에 현재 인덱스 2의 값을 저장하고 규칙 연산 결과 변수 값을 조건_1과 조건_2, 연산자 정보를 이용하여 결정한다. 여기서 규칙 연산 결과는 조건들의 연산 결과를 저장하는 변수이고 연산 순서 배열의 문자열이 '#'을 가지고 있을 경우에만 현재 규칙 연산 결과 값을 판단하여 제어 메시지를 저장하고 종료한다. 이 과정은 연산 순서 배열의 모든 문자열에 대해 검사가 완료되면 제어 메시지를 반환하고 종료된다.

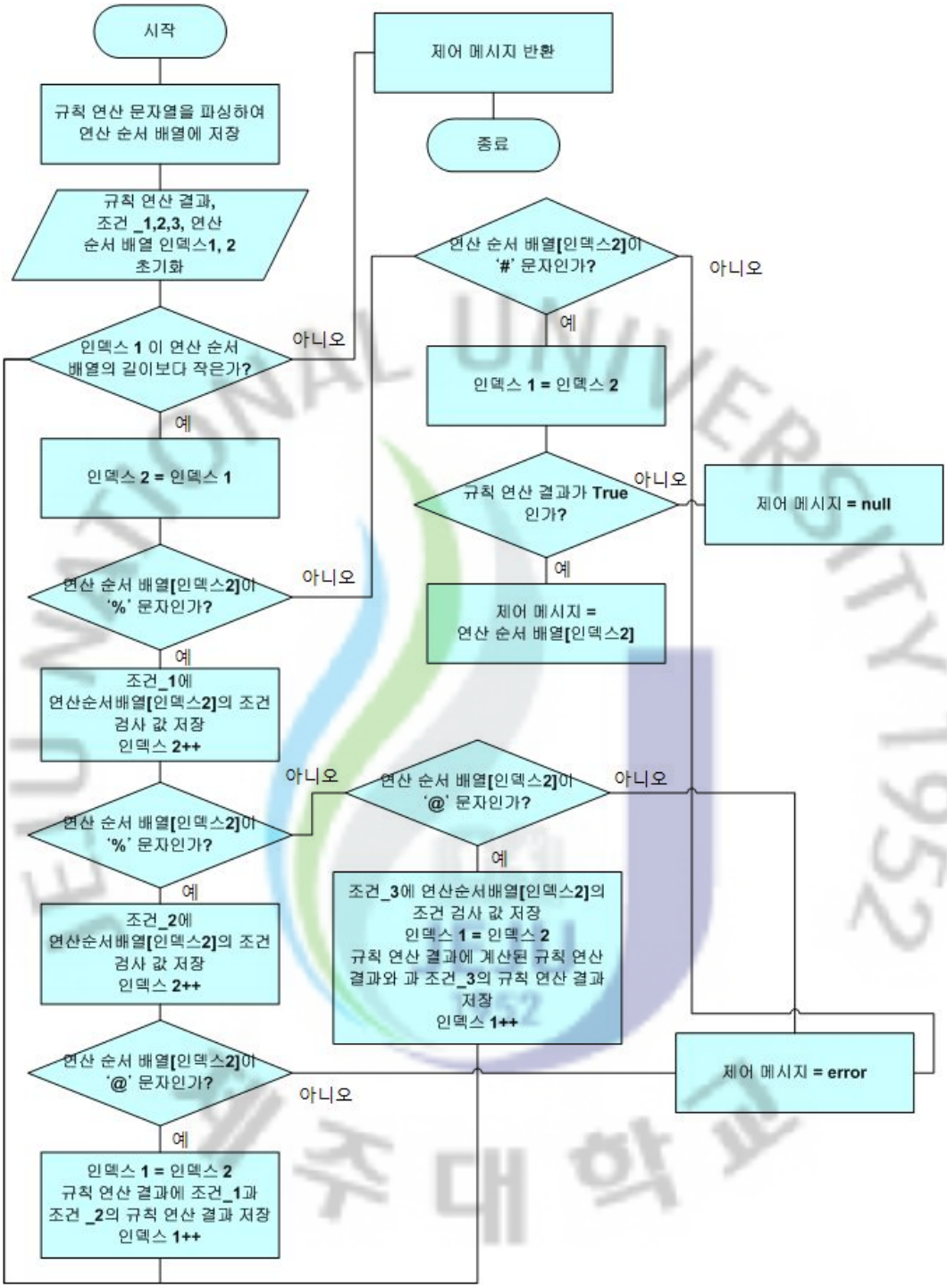

그림 18. 조건 정보의 규칙 연산 알고리즘

Ⅳ. USN 기반의 구동체 제어 미들웨어 구현

본 장에서는 위에서 설계한 구동체 제어 미들웨어의 데이터 처리 모듈과 구동체 제어 모듈을 구현하고 테스트 환경을 구축하여 가상의 구동체 제어 응용을 통해 본 논문에서 제시한 구동체 제어 미들웨어의 성능을 평가한다.

1. 구현 환경 및 테스트 환경

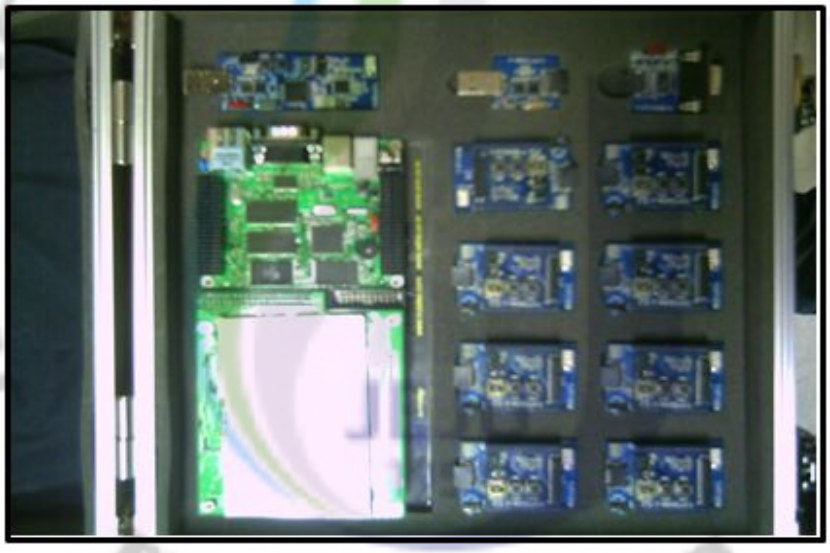

그림 19. 구동체 제어 미들웨어에서 사용하는 센서 하드웨어

센서 하드웨어로 Maxfor 사의 TIP700시리즈를 사용한다. .NET Framework 기반의 C#을 이용하여 구동체 제어 미들웨어와 가상의 구동체를 구현하였다. 데이터 백업을 위한 데이터베이스로는 MS-SQL 2005를 사용하였고 메시지 큐는 Windows XP Professional에서 제공하는 MSMQ(Microsoft Message Queue)를 사용한다. 테스트 환경으로는 가상으로 구현한 구동체 제어 응용 N개와 센싱 데이터 수집 미들웨어, 구동체 제어 미들웨어를 구축하고 수집되는 센싱 데이터의

온도 값에 따라 구동체에 제어 메시지를 생성하여 구동체로 전송한다.

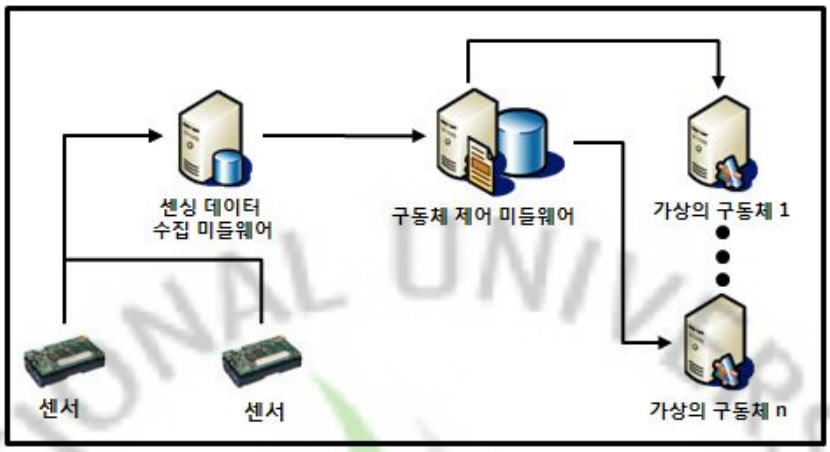

그림 20. 구동체 제어 시스템의 테스트 환경

2. 구현 결과

구동체 제어 미들웨어가 동작하기 위해서는 구동체 제어 응용의 관리자가 규칙 데이터를 먼저 설정한다. 그림 21은 구동체의 규칙 데이터의 등록을 담당하는 웹페이지 화면이다. 먼저 데이터 수집 미들웨어의 등록 과정으로 생성된 정보를 보여줌으로써 구동체 제어 응용에서 필요한 센싱 데이터를 얻을 수 있는 데이터 수집 미들웨어를 선택하고 제어 메시지를 생성할 수 있는 조건 정보와 규칙 연산을 정의한다.

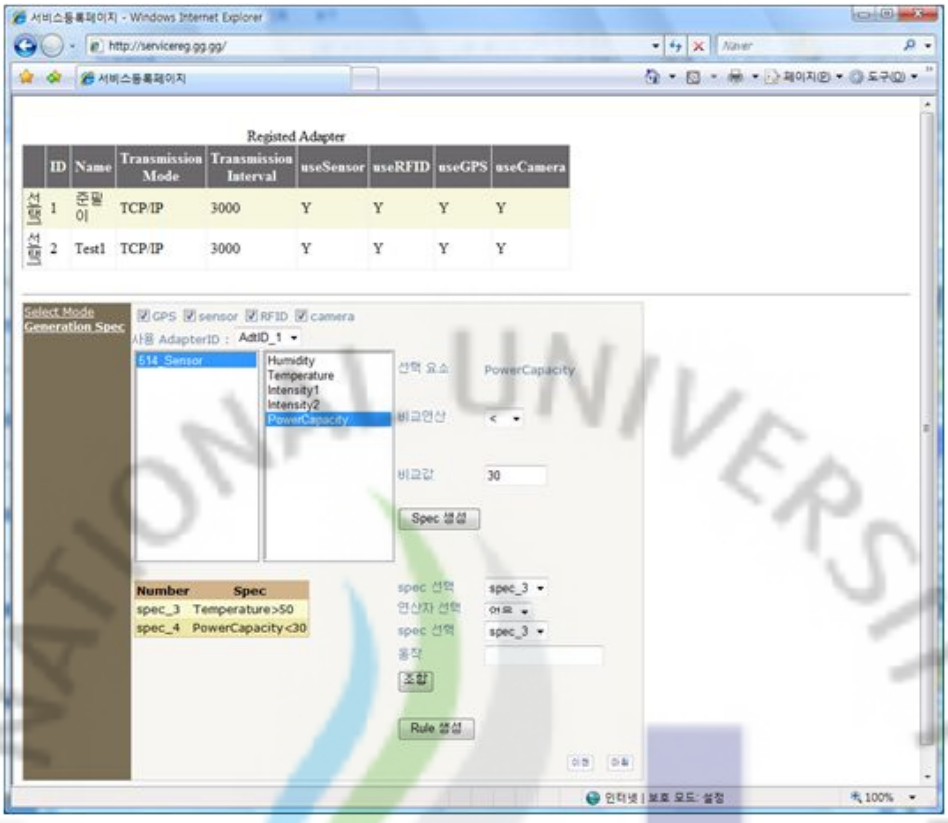

그림 21. 구동체 제어 응용의 규칙 데이터 등록 화면

구동체 제어 응용의 규칙 데이터 등록 페이지를 통해 생성된 XML 파일은 그림 22와 같다. 그림에서 보면 구동체 제어 응용의 데이터 처리 요구 정보와 조건 정보, 조건 정보의 규칙 연산 정보가 정의된다. 이 정보들은 데이터 수집 미들웨어의 등록 정보를 토대로 생성되며 최종적으로 규칙연산이 True 일 경우 <action> 엘리먼트로 되어 있는 부분의 값이 제어 메시지가 된다. 그림에서 구동체 ID가 2번인 구동체 제어 응용의 조건 3(spec_3)은 센서 514(하드웨어 코드 정보)가 수집하는 온도 데이터에 대해 50도보다 이상일 경우라는 정보이며 조건 4(spec_4)는 514번 센서의 전력량이 30 이하일 경우라는 정보이다. 이를 아래 부분의 규칙 연산에서 조건 3과 조건 4를 OR 연산을 하여 True 일 경우 Warning 이라는 제어 메시지를 생성하라고 정의한다.

그림 22. 규칙 데이터 XML 파일

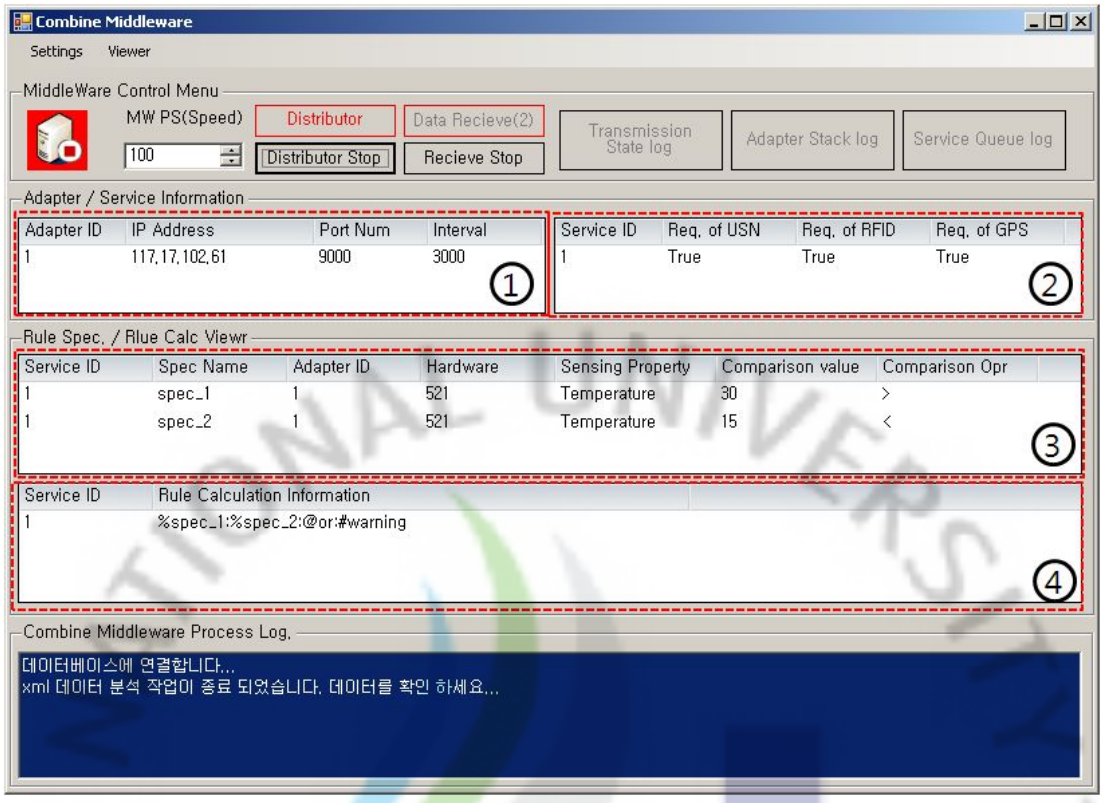

그림 23. 구동체 제어 미들웨어 구현 화면

그림 23은 구동체 제어 미들웨어의 구현 화면이다. 그림에서 1번은 센싱 데이터 수집 미들웨어의 등록 정보로써, XML 데이터를 파싱하여 센싱 데이터 수집 미들웨어의 고유 ID값, IP 주소, 포트번호, 데이터 통신 주기를 환경변수에 저장한 모습이다. 2번은 구동체 제어 응용의 데이터 처리 요구 정보이며, 3번은 규칙 데이터 중에서 조건 정보를 나타낸다. 그림에서 조건 정보1(spec_1)은 센싱 데이터 수집 미들웨어 ID 1의 센싱 데이터 중 521번 센서 하드웨어의 온도가 30도보다 이상일 경우이고 조건 정보2(spec_2)는 센싱 데이터 수집 미들웨어 ID 1의 센싱 데이터 중 521번 센서 하드웨어의 온도가 15도 이하일 경우라는 조건이다. 이 조건 정보는 4번의 규칙연산에 의해 OR 연산을 취할 경우 해당하는 온도 데이터가 수집되면 warning이라는 제어 메시지를 보내달라고 정의하고 있다.

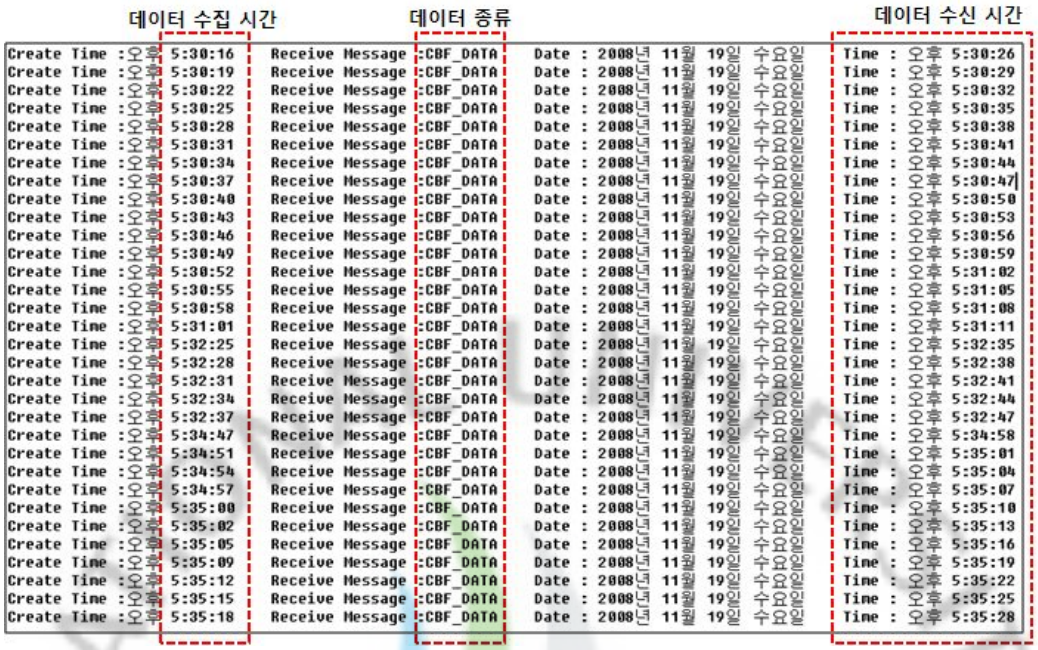

그림 24. 센싱 데이터 수집 미들웨어에서 수신한 센싱 데이터 로그 파일

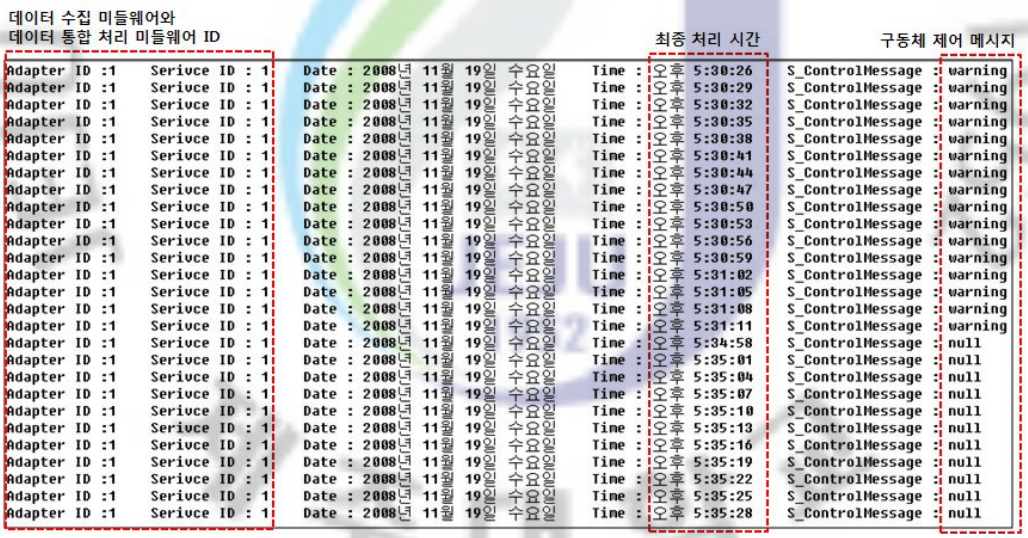

그림 25. 메시지 큐에 저장되는 최종 처리 후 센싱 데이터 로그 파일

위 그림 24와 25는 구동체 제어 미들웨어에서 처리하는 센싱 데이터를 수신 시점과 처리 완료 시점의 로그파일로 생성하여 분석한 결과이다. 그림 24는 센싱 데이터 수집 미들웨어에서 전송된 센싱 데이터를 받는 즉시 생성한 로그파일로써 데이터 종류는 CBF_DATA, 즉 센싱 데이터가 존재한다는 의미이며 데이터

수신 시간이 기록된다. 그림 25는 구동체 제어 모듈에서 처리가 완료되어 해당하는 메시지 큐에 저장되기 전 생성되는 로그파일로써 센싱 데이터 수집 미들웨어와 구동체 제어 응용의 고유 ID 값과 저장 시간, 제어 메시지가 기록된다. 그림에서 보듯이 제어 메시지가 warning으로 나타나다가 null로 기록되는 것을 볼 수 있다. 이는 센서의 온도를 높여서 30도 이상인 상태에서 측정을 시작하여 온도가 내려감에 따라 제어 메시지가 바뀜을 알 수 있다. 또 한 가지 알 수 있는 것은 데이터 수신 시간과 데이터 처리 후 저장 시간이 차이가 없음을 볼 수 있다. 이는 한 개의 데이터 수집 미들웨어에서 보내오는 데이터 처리 시 거의 성능에 지장이 없음을 보여준다. 그림 26은 메시지 큐에 저장되는 데이터이다. 구동체 제어 응용을 5개로 설정 했을 경우의 1번 구동체 제어 응용에게 제공될 메시지 큐의 상황을 보여주며 웹서비스로 요청할 경우 XML 형식의 데이터로 전송된다.

그림 26. 메시지 큐의 센싱 데이터 및 제어 메시지 목록

3. 성능 분석

구동체 제어 미들웨어에서 구동체를 제어하는 방법은 먼저 센싱 데이터 수집 미들웨어에서 수집하는 센싱 데이터를 구동체 제어 미들웨어로 전송하고 데이터 처리 모듈과 구동체 처리 모듈을 거쳐 가상의 구동체 제어 응용으로 제어 메시지를 전송하는 시간을 측정함으로써 제안한 구동체 제어 미들웨어의 성능을 분석 한다. 이 때 가상의 구동체 제어 응용의 수를 늘려가면서 비교하여 이를 그래

프로 나타내면 아래 그림들과 같다.

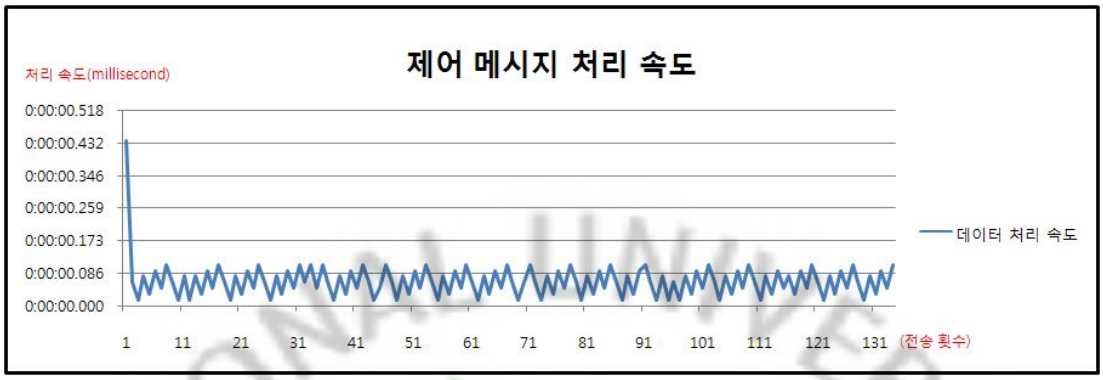

그림 27. 구동체 제어 미들웨어의 제어 메시지 처리 속도 그래프

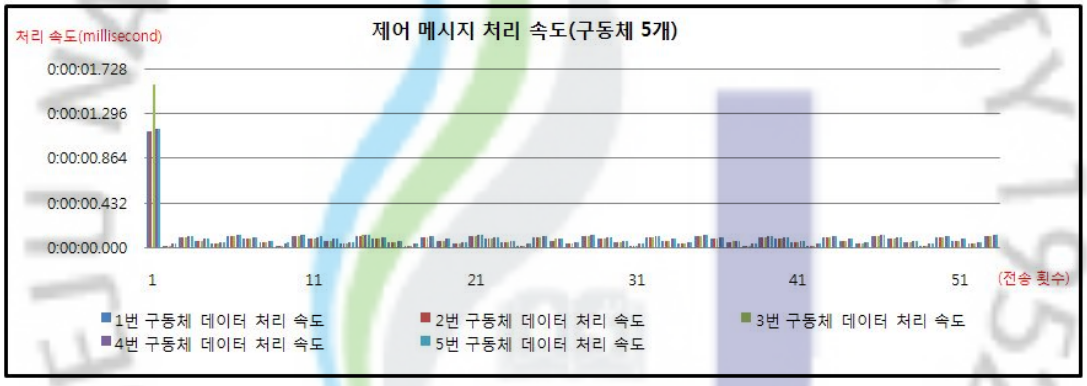

그림 28. 5개의 구동체를 제어하는 경우 제어 메시지 처리 속도 그래프

먼저 그림 27은 구동체 1개를 제어하는 경우 제어 메시지의 처리 속도를 나타낸 그래프이다. 구동체 제어 미들웨어가 센싱 데이터 수집 미들웨어로부터 센싱 데이터를 전송받은 시간과 데이터 처리 모듈, 구동체 제어 모듈 과정을 수행하고 구동체에게 제어 메시지를 전송한다. 이때, 가상의 구동체에서 데이터를 받은 시간을 측정하여 차이를 비교 하였다. 그래프를 보면 맨 처음 1회 전송 시 약 400 밀리세컨드가 소요되는 것을 알 수 있다. 이는 초기 구동체와 제어 메시지를 주고 받기위해 소요된 연결 시간이 포함되는 것이다. 이후 제어 메시지를 전송하는 시간이 약 85 밀리세컨드 아래에서 유지됨을 볼 수 있다. 그림 28은 가상의 구동체를 5개로 설정하고 제어 메시지를 처리하는 속도를 측정한 그래프이다. 이 그

래프를 보면 초기 연결 시간은 1개의 구동체를 제어하는 경우보다 약 2배 이상이 소요되는 것을 알 수 있다. 이는 다수의 연결을 멀티 스레드로 처리함에서 나타나는 소요시간이 추가된 경우이다. 하지만 연결이 이루어진 후 제어메시지 전송에 있어서는 1개의 구동체 제어 시 나타나는 시간과 거의 차이가 없음을 알 수 있다.

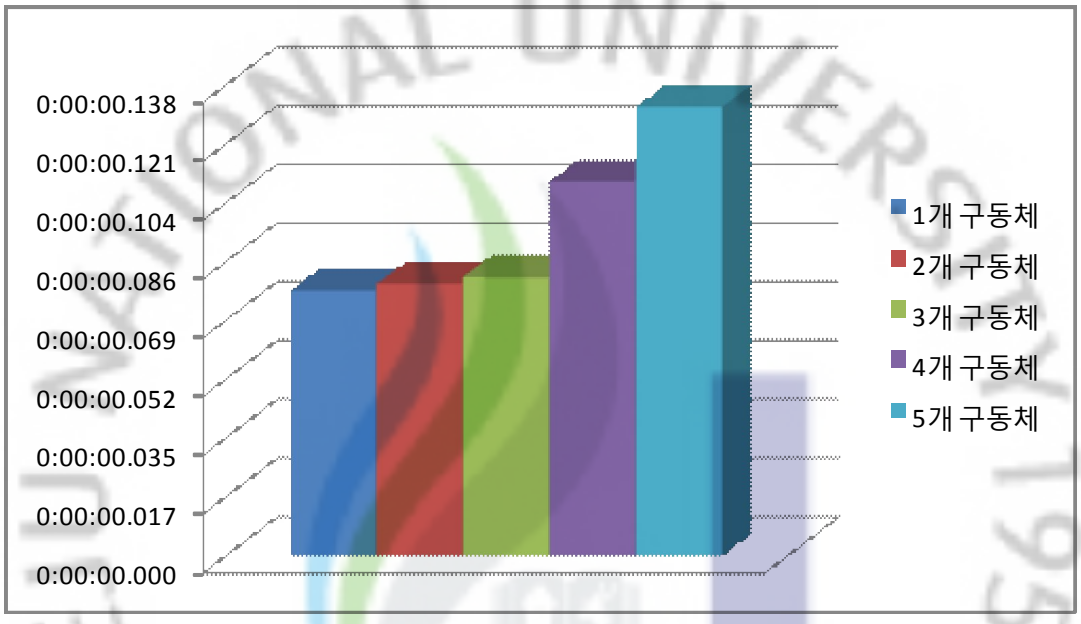

그림 29. 구동체 제어 미들웨어의 평균 데이터 처리 속도 그래프

그림 29는 구동체 제어 미들웨어에서 구동체의 수를 1개에서 5개로 추가하면서 처리 속도를 측정한 평균 처리 속도를 나타낸다. 하나의 구동체를 제어 할 경우보다 다수의 구동체를 제어하는 시간이 늘어나는 것을 볼 수 있다. 이는 구동체 제어 모듈에서 제어 메시지를 생성하는 소요시간이 늘어나는데 따른 것으로 추측된다. 하지만 대부분의 처리 속도가 0.138초 미만으로 구동체의 수가 증가하더라도 제어메시지 처리 속도에 따르는 성능은 매우 빠른 것으로 판단된다.

4. 평가 및 고찰

본 논문에서는 구동체 제어 미들웨어를 설계하고 수집되는 센싱 데이터의 상

황에 따라 사용자가 등록한 규칙데이터와 비교하여 제어 메시지를 생성하는 알고리즘을 제안한다. 이는 기존에 연구된 Sentire 미들웨어에서 구동체를 제안하기 위해 제어 메시지를 전송하는 과정에 비해 보다 실시간적인 처리가 가능하다. 또한 구동체 스펙 구성 인터페이스를 설계하여 새로운 구동체를 제어할 수 있는 기능을 제공한다. 제안한 구동체 제어 미들웨어와 가상의 구동체를 이용하여 테스트 환경을 구축하고 처리 속도를 분석하여 성능을 검증하였다. 이를 통해 다수의 구동체 제어 시 초기 연결시간, 즉 TCP/IP 통신을 사용한 경우의 연결에 소비된 시간이 구동체가 늘어날수록 증가함을 알 수 있었다. 하지만 이 후의 제어 메시지 처리에는 크게 차이가 나지 않는 일정한 처리 속도를 갖는 것으로 보인다. 실제 구동체를 제어하기 위해서는 제어 메시지를 받아서 동작하는 하드웨어가 필요하다. 또한 구동체의 특성에 따라 TCP/IP 통신 이외의 다른 통신 방법도 고려해야 할 것이다. 무엇보다 중요한 점은 화재와 같이 긴급 상황 시나 정확한 처리 동작을 요구하는 구동체일 경우 처리 속도는 제시한 구동체 제어 미들웨어의 처리 속도보다 더욱 향상되어야 할 것이다.

Ⅴ. 결 론

앞으로 USN 미들웨어는 데이터 처리·저장·분배 기능뿐만 아니라 수집된 데이터를 분석하여 상황에 맞게 구동체를 동작시키는 기능이 포함되어야 할 것으로 사료된다. 이에 본 논문은 기존에 연구 되었던 SANET 시스템의 Sentire 미들웨어의 기능을 분석하고 확장하여 USN 기반의 구동체 제어 미들웨어를 설계하고 구현한다. 이를 위해 USN 미들웨어의 Server-side 미들웨어의 기능 중에서 센싱 데이터를 처리하는 데이터 처리 모듈과 구동체 제어를 담당하는 구동체 제어 모듈을 설계하고 구현하였다. 또한 테스트 환경을 구축하고 실제 센싱 데이터를 수집하고 설정한 규칙 데이터에 따라 제어 메시지를 생성하는 모습을 볼 수 있었다. 비록 실제 다양한 구동체를 제어하기 위한 표준 구동체 스펙 구성 방법과 여러 가지 통신방식에 대한 적용을 고려해보지는 못했으나 표준 인터페이스인 웹서비스를 통해 구동체 제어 응용으로 제어 메시지를 전송 할 수 있었다. 본 논문에서는 다양한 USN 미들웨어의 기능 중 상황 인식 기능 및 구동체 제어 기능을 실제 구현하고 테스트를 통해 성능 분석을 하였다. 앞으로 새로운 유비쿼터스 사회의 발전을 위해 제시한 구동체 제어 모듈이 여러 서비스 응용에 필요한 모듈로 발전 될 수 있도록 표준 제어 메시지 구조와 통신 방법에 대한 연구를 진행 하겠다.

참고문헌

[1] 김대영, 김재언, 성종우, 이강우, “센서 네트워크 운영체제/미들웨어 기술동향,” IITA 주간기술동향 통권 1221 호, 200,11

[2] 김민수, 김광수, 이용준 "USN 미들웨어의 특징 및 기술개발 동향", 주간기술동향 통권 1284호 2007, 2, 21

[3] Joel W. Branch, John S. Davis II, Daby M. Sow, Chatschik Bisdikian, “A Framework for Building Middleware for Sensor and Actuator Networks”, Proceedings of the 3rd Int’l Conf. on Pervasive Computing and Communications Workshops, 2005

[4] E. Niemelä and T. Vaskivuo, “"Agile Middleware of Pervasive Computing Environments"", Proc. 2nd IEEE Annual Conf. on Pervasive Comp. and Comm. Workshops, Orlando, Florida, Mar. 2004, p. 192.

[5] Y. Yu, B. Krishnamachari, and V. K. Prasanna, “Issues in designing middleware for wireless sensor networks," IEEE Network, vol. 18, no. 1, Jan. 2004, pp. 15-21.

[6] W3C Document Object Model, http://www.w3.org/DOM/

[7] Andrew Troelsen, “c# and the .NET Platrorm Second Edition” (주)사이텍미디어, 2005. 6. 20, p 572-572.

[8] Tim Wark, Chris Crossman, Wen Hu, Wing Guo, Philip Valencia, Pavan Sikka, Peter Corke, Caroline Lee, John Henshall, Kishore Prayaga, Julian O'Grady, Matt Reed, Andrew Fisher "The Design and Evaluation of a Mobile Sensor/Actuator Network for Autonomous Animal Control", Proceedings of the 6th international conference on Information processing in sensor networks, 2007, p 206-215

[9] 부준필, 고민정, 정주영, 김도현, 이상준 "다양한 상황 데이터 처리를 위한 통합 인터페이스 설계 및 구현", 한국멀티미디어학회 추계학술발표대회 논문집 p.143 2008.11.21

[10] 고민정, 부준필, 정주영, 김도현, 이상준 "GPS, Sensor, RFID, Camera 구성 관리 모듈 설계 및 구현", 한국 멀티미디어학회 추계학술발표대회 논문집 p.142 2008.11.21

[11] 부준필, 양현규, 김도현 "TinyOS 기반의 센서 제어 모듈 설계 및 구현" 대한임베디드공학회 추계학술대회 논문집 p.278 2007.11.9

[12] 정순원, 차홍규, 김상일, 김대근, 박미정, 심준환 "Zigbee를 이용한 자동 가스차단 및 환기 시스템" 한국마린엔지니어링학회 공동학술대회 논문집, 2008

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

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

[15] 김정현, 김대영, 성종우, 송형주, 김수현, "센서 네트워크에서 컨텍스트 인지 서비스 개발을 위한 미들웨어", 한국컴퓨터종합학술대회 2005 논문집 제 32권 제 1호, 2005.7