사물 인터넷 환경에서 CoAP을 이용한 시맨틱 그룹 통신

연세대학교 대학원

컴퓨터과학과

조 건 희

사물 인터넷 환경에서 CoAP을 이용한 시맨틱 그룹 통신

지도 이 경 호 교수

이 논문을 석사 학위논문으로 제출함

2015년 8월 일

연세대학교 대학원
컴퓨터과학과
조 건 희

조건희의 석사 학위논문을 인준함

심사위원 이 경 호 인

심사위원 한 요 섭 인

심사위원 이 경 우 인

연세대학교 대학원

2015년 8월 일

감사의 글

어느덧 짧지도 길지도 않았던 시간이 지나 졸업을 앞두고 있습니다. 2년 전 연구실에 들어와 아낌없는 격려와 지도를 해주신 이경호 교수님께 진심으로 감사 드립니다. 또한, 저의 논문 심사를 맡아주신 한요섭 교수님과 이경우 교수님께 감사를 드립니다.

연구실 생활을 하면서 함께 연구를 하고 밤을 지새웠던 연구실 선배님들과 후배님들께도 깊은 감사 드립니다. 연구실의 가장 고학기이며 모든 일을 도맡아 하신 세진이 형, 국적은 캐나다이지만 한국 사람보다 더 한국 사람 같은 주익이 형, 장난이 많지만 때론 아주 가끔씩 조언을 해준 상진이 형, 많은 대화를 나누지 못해서 아쉬운 형진이 형, 입학을 같이하여 많은 도움을 주신 동기들 진호 형, 숭광이형, 웅남, 민재, 이번에 같이 논문 심사 및 취업 준비를 한 기용이 형, 연구실의 만형들로 알찬 정보들을 주신 강호형과 현석이 형, 다음 학기에 저와 같은 과정을 걷게 될 민정이, 마지막 학기에 버퍼 방에서 함께 지낸 승민이, 병국이, 원우, 승준이, 그리고 졸업해서 회사에 계신 필식이 형, 동갑내기 지훈이, 같이 취업을 하며 가끔 안부를 물었던 정우 형에게 감사하다는 말을 전하고 싶습니다.

학사 과정 때부터 8년이란 시간 동안 함께했던 대학교 친구들에게도 감사하다는 말을 전하고 싶습니다. 매일 시도 때도 없이 카카오톡 대화 창에서 떠드는 13명, 동하 형, 순정이 형, 승희 형, 근환이 형, 태환이 형, 정식이, 주성이, 남건, 경민이, 강영이, 준헌이, 승용이, 유희 형, 그리고 동기들 중 가장 먼저 시집 간 미정이, 모두 변함없는 모습으로 늘 함께 했으면 좋겠습니다.

끝으로, 지금까지 저를 낳아주시고 길러주신 부모님께 감사의 말을 전하고 싶습니다. 가족들을 위하여 매일 밖에서 힘써주시는 아버지, 집에서 가족들을 위하여 힘써주시는 어머니, 그리고 지금은 매형과 함께 따로 살고 있지만 부모님 만큼 저를 많이 챙겨준 누나에게 감사하다는 말을 전합니다.

목차

그림 목차

표 목차

국문 요약

사물 인터넷 환경에서 CoAP을 이용한 시맨틱 그룹 통신

사물 인터넷 환경은 모든 사물들이 IP 기반의 인터넷에 연결되어 필요한 서비스를 제공하거나 제공한다. 사물 인터넷 기술이 발전함에 따라 다양한 사물들의 수가 전세계적으로 늘어나게 되었다. 그에 따라 관리해야 할 사물들의 양과 통신 데이터 양이 크게 늘어나게 되어 소모되는 전력량도 또한 늘게 되었다. 사물들과 데이터들을 효과적으로 처리하고 생기는 전력 소모를 낮추기 위하여 많은 양의 사물들과 데이터를 관리하는 방법이 연구되고 있다. 그 중에서도 저전력 기반의 프로토콜인 CoAP(Constrained Application Protocol)에 대한 연구가 활발하게 이루어지고 있다.

CoAP은 웹 기반의 프로토콜인 HTTP의 대체로 HTTP 와는 달리 메시지 타입 및 크기가 달라 사물 인터넷 환경에 더욱 적합하다. 또한 CoAP은 멀티캐스트를 지원하여 다수의 사물들을 한 번에 관리 및 제어할 수 있다. 하지만, 현재 프록시 서버를 통한 멀티캐스팅 데이터 처리방법 기술은 미비한 실정이다. 프록시를 이용하여 멀티캐스팅을 할 때 요청을 보내는 사용자는 단일 응답을 기대하기 때문에 다수의 응답을 보내는 것은 맞지 않으며 다수의 응답을 보내게 되면 트래픽 오버헤드가 발생하게 된다.

이 문제를 해결하기 위하여 본 논문에서는 사물들의 매쉬업을 이용하여 프록시를 통하는 멀티캐스팅 데이터 처리 방법과 시맨틱 데이터로 가공하여 사용자들에게 제공하는 방법을 제안한다. 멀티캐스팅을 하기에 앞서 사물 매쉬업을 이용하여 시맨틱 기반의 멀티캐스팅 그룹을 생성하는 방법을 제안하여 CoAP 환경에 적용한다.

핵심되는 말: 사물 인터넷, CoAP, 프록시, 멀티캐스트 통신, 시맨틱 조합

제 1 장 서론

최근 몇 년 동안, 건물이 사회에서 에너지 소모의 40%를 차지하며 지속적으로 증가하면서 주요 에너지 소비원이 되었다 [1]. 많은 상업 건물들은 HVAC (heating, ventilation and air conditioning) 시스템을 사용하여 건물의 온도와 습도를 제어하고 있다. 온도와 습도를 제어하며 환경을 쾌적하게 하는 것이 주된 목적이지만, 증가하는 에너지 소모량으로 인해 최근에는 에너지 최적화 및 소모율 감소를 목적으로 하고 있다. 예를 들어, 밤에 사람들이 퇴근하고 없는 건물에서는 자동으로 기계들을 끄고 사람들이 출근하는 아침에는 출근하기 전에 자동으로 미리 켜는 것으로 건물의 에너지 소모율을 관리한다.

스마트 홈, 스마트 빌딩 등이 생겨나면서 건물 내에서 각종 기기들을 관리하는 방법과 에너지를 절약하기 위한 많은 연구들이 제안되어 왔다. 기존 연구들은 건물의 자동화 시스템을 다루는 연구들로 Building Automation System(BAS), Building Automation Control network(BACnet)[2], KNX [3], EnOcean [4], Open Building Information Xchange(oBIX)가 연구되었다. 앞의 시스템과 네트워크들은 건물 자동화를 위한 목적으로 사용되고 있다. 하지만 이 시스템의 네트워크들은 각기 다른 네트워크 기술이기 때문에 다른 네트워크 기술을 쓰는 장비가 설치되었을 경우에 최적화 또는 특정 제약 문제가 생긴다.

최근에는 모든 사물들이 IP 기반 네트워크에 연결되는 사물 인터넷 [5]이 구축되면서 앞서 언급한 특정 네트워크를 사용할 필요가 없게 되며, 6LoWPAN 과 Wi-Fi 에 기반한 센서 네트워크를 건물 네트워크에 적용시키는 연구가 진행되고 있다. 또한 에너지 소모를 최소화 시키기 위해 저전력

기반의 프로토콜인 Constrained Application Protocol(CoAP) [6]이 연구 및 표준화 되고 있다. CoAP 은 저전력, 건물 자동화 그리고 machine-to-machine(M2M)을 응용하기 위한 목적으로 연구되고 있다. CoAP 의 기능 중에는 기능이 같거나 위치에 따라 여러 사물을 조합한 그룹을 이용한 그룹 통신이 지원되는데, 이는 멀티캐스트 기반의 통신으로 이뤄진다. 건물 자동화 시스템을 구축하는데 있어서 CoAP 의 그룹 통신을 사용하게 되면 다수의 사물들을 동시적으로 제어할 수가 있게 되어, 건물의 제어가 수월해지고 저전력 기반이기 때문에 에너지를 절감할 수가 있게 된다. 그러나 그룹 통신을 하기 위해서는 먼저 그룹을 생성해야 하는데, 그룹은 사용자가 직접 개입하여야 생성이 된다. 그룹의 속성은 그룹명과 그룹 경로만 있기 때문에 해당 그룹이 갖고 있는 의미를 알 수가 없으므로, 사람은 그룹이 어떤 그룹인지는 알 수 있지만 시스템은 알 수가 없다. 또한 HTTP 서버에서 CoAP 서버로 프록시를 통한 멀티캐스트 통신을 할 경우, 멀티캐스트 통신 이후 전송되는 다수의 응답이 HTTP 서버로 다시 프록시를 통해 전달이 될 때 부하가 생기게 된다. 부하가 생기지 않게 하기 위해 다수의 응답들을 조합하는 방법이 필요한 상황이다.

본 논문에서는 사물 인터넷 환경에서 이종의 네트워크간 멀티캐스트 요청을 전달하여 다수의 사물들을 관리 및 제어 방법을 제안한다. 프록시를 통하여 사용자의 요청을 CoAP 에 맞게 변환을 한 다음 멀티캐스트 통신을 수행하며, 멀티캐스트 그룹 생성은 기존의 방법처럼 사용자가 수동으로 그룹에 포함되는 사물들을 지정하지 않고 하나의 질의를 이용하여 자동 생성한다. 또한 멀티캐스팅 후 생성되는 다수의 응답을 조합하여 전송 부하를 줄인다.

본 논문의 구성은 다음과 같다. 2장에서는 기본적인 CoAP의 특징, CoAP 멀티캐스트 통신 기술, CoAP에 시맨틱 데이터의 적용, CoAP 프록시 환경에 대한

기존 연구를 기술한다. 3장에서는 본 논문에서 제안하는 CoAP 프록시 시스템의 전반적인 구조와 각 모듈에서 하는 작업들에 대하여 설명한다. 4장에서는 제안한 방법의 검증을 하기 위한 실험 및 결과 분석을 보여준다. 마지막으로 5장에서는 결론과 향후 연구를 기술한다.

제 2 장 관련 연구

CoAP [6]은 한정적인 자원을 갖고 있는 네트워크(Constrained network)에서 사용하는 전송 프로토콜이다. 6LoWPAN(IPv6 over Low power WPAN)과 같은 무선 네트워크 환경에서 사물들간 통신을 하기 위해 IPv6 패킷을 작게 분할하여 전송한다. 패킷 전송이 줄어들기 때문에 메시지 부하를 줄이게 되어 혼잡(Congestion)을 줄인다. CoAP 의 주요 목적은 저전력, 건물 자동화(Building Automation) 그리고 M2M 응용을 하기 위함이다. 그 외에 CoAP 의 특징은 다음과 같다. CoAP 의 메시지(message)는 UDP 를 이용하여 종단점(endpoint)간에 교환을 한다. CoAP 은 사물에 따라 세부적인 옵션을 지정할 수 있으며 메시지를 전송할 때 있어서 옵션 항목에 담아서 전송하게 된다. RESTful 서비스를 충족하여 기본적으로 GET, POST, PUT, DELETE 등의 HTTP 메소드를 사용한다.

CoAP 은 사물인터넷 환경에서 다수의 디바이스들과 통신할 수 있는 멀티캐스트 통신(multicast communication)을 지원한다. 멀티캐스트는 일대다수(one-to-many) 또는 다수대다수(many-to-many)로 통신을 할 수 있다. HTTP 같은 경우에는 멀티캐스트가 지원되지 않기 때문에 디바이스들과 통신하기 위해서는 단일통신인 유니캐스트(unicast) 통신으로 여러 요청 메시지를 전송하여 통신하게 된다. 이때 여러 요청 메시지와 응답 메시지를 받기 때문에 네트워크 혼잡이 생기게 된다.

반면에 CoAP 에서의 멀티캐스트 통신은 하나의 요청 메시지만 보내게 되고 Acknowledgement(ACK)가 없는 non-confirmable(NON) 메시지를 보내서 HTTP 환경에서 생기는 네트워크 혼잡을 줄일 수 있다. 다만 NON 메시지는 ACK 를

받지 않기 때문에 ACK 를 보내는 confirmable(CON) 메시지 에 비해 신뢰성이 떨어진다.

Rahman 등 [7]는 멀티캐스트를 지원하는 CoAP 그룹 통신에 대해 제안하였다. 한정적인 사물들은 그 숫자와 종류가 다수 존재하지만, 이 사물들을 기능이나 위치로 관계를 엮을 수 있다. 이 사물들을 기능이나 위치에 기반하여 그룹으로 조합할 수가 있다. 또한 그룹은 여러 개 만들 수가 있다. 예를 들어, 건물 안에 한 층에 있는 조명들이 1 번 그룹으로 조합되어 있을 수 있고, 온도계들은 2 번 그룹에 조합될 수가 있다. 이 그룹들은 멀티캐스팅하기 전에 조합이 되어있을 수도 있고 동적으로 조합이 될 수가 있다. 그룹에 속해 있는 사물들의 정보를 보내거나 받을 때, 그룹 통신은 통신에 걸리는 시간과 필요로 하는 대역폭을 줄일 수가 있다. 하지만, 그룹 통신이 지원되지 않는 HTTP 는 지연시간이 길고 상대적으로 큰 대역폭을 요구할 수 있다.

그룹통신을 하기 위해서는 원하는 사물들을 그룹으로 조합을 해야 한다. 각 그룹에는 프로필(profile)이 존재한다. 프로필에는 키와 값(key/value) 두 개가 존재한다. Key 는 그룹 구성원의 이름이며 value 는 그룹 구성원에 해당하는 값이다. 그룹명(name)은 사용자가 지정하는 텍스트 형태로 되어있으며, 값(value)은 그룹의 주소가 되며 URI 형태로 호스트 주소와 포트번호로 구성되고 마찬가지로 사용자가 지정할 수 있다. 그룹명과 주소가 구성이 되면 해당 주소를 통해 요청 메시지를 보낼 수가 있다. 생성된 그룹은 CoAP 의 리소스 디렉토리(resource directory) [8]에 추가되며 그룹에 속한 사물들의 리스트를 갖게 된다.

CoAP 명세에 있는 그룹은 name 과 path 만이 존재한다. 그 외에 그룹에 대한 옵션 정보는 없다. 그룹의 name 과 path 는 그룹을 생성할 때 사용자가 직접

URI 템플릿으로 작성을 해야 하며, 이는 단순 문자열이기 때문에 시맨틱 정보가 없다. 명세의 그룹통신은 name 과 path 에 대한 정보만 있는 것 때문에 Ishaq 등 [9][10]는 그룹에 대한 옵션 정보를 추가하였다. 엔티티 매니저(Entity Manager)를 이용하여 그룹을 CoAP 기반의 엔티티(entity)로 생성하였다. 엔티티는 CoAP 의 그룹 리소스들로 구성한 것으로 새로운 CoAP 자원이 된다. 다음 그림 1 은 제안하는 엔티티 매니저로 CoAP 명세보다 그룹을 만들 때보다 쉽게 등록, 조작, 저장을 수행한다. 이 논문은 그룹을 엔티티로 생성하면서 옵션정보를 추가하였다. 하지만 옵션 정보는 그룹에 있는 리소스를 기반으로 하여 그룹의 옵션을 프로필에 넣은 것이며, 이 리소스들의 옵션들은 리소스 생성시에 사용자가 직접 명시하여 넣게 된 것이다.

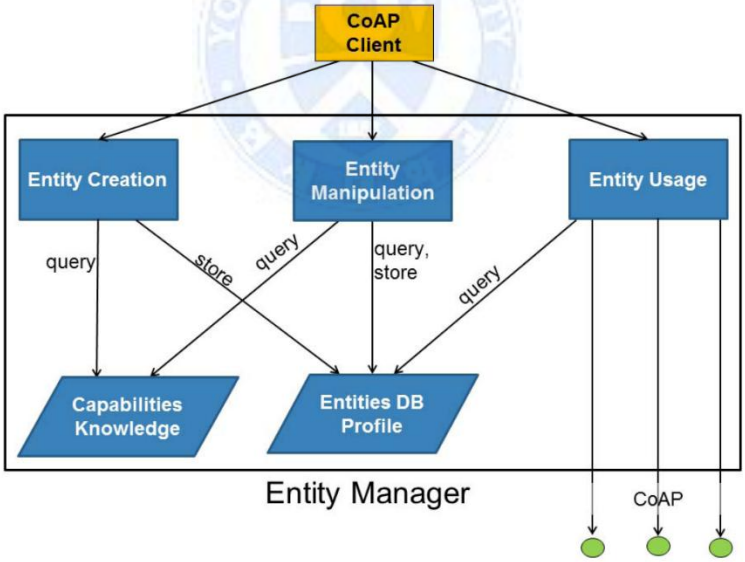

그림 1. Entity Manager 구조 [9][10]

사용자가 리소스 정보 및 옵션을 사용자 임의대로 넣을 경우에는 해당 시스템에서 용어들을 다른 형태로 이해할 수가 있다. 예를 들어, 온도 센서 또는 그룹의 정보에 temp 로 넣을 경우에 temp 의 의미를 temperature 가 아닌 temporal feature 로 이해할 가능성이 있다. 용어 이해 문제를 해결하고 각 도메인들의 정보를 공통의 용어로 정의하기 위한 연구도 진행되었다. W3C Semantic Sensor Network Incubator Group 에서는 Semantic Sensor Network(SSN) 온톨로지 [11]를 연구 및 개발하여 각종 사물들의 개념을 정의하여 용어 정리를 하였다.

SSN 온톨로지 외에 Stephan 등 [12]이 사물 인터넷 환경을 위한 도메인 모델을 이용하여 주요 개념들과 관계를 정의하여 사물 인터넷 환경을 위한 공통의 용어 및 사물의 분류 체계를 제안하였다. 하지만 도메인들이 이 공통 모델을 사용하지 않고 자체 도메인을 사용하게 된다면 이종의 도메인들의 혼용을 위한 방법이 필요하다. 각 도메인들의 사물 정보를 의미적으로 이해하는 방법으로는 온톨로지에 기반한 시맨틱 검색이 있다. Paolucci 등 [13] 은 DAML-S[1] 에 기반하여 웹 서비스 프로필들의 시맨틱 매칭 방법을 제안하였다. 매칭 엔진을 개발하여 사용자가 요청하는 서비스를 키워드 매칭을 통해 정교하게 탐색한다. Ruta 등 [14]에서는 CoAP 환경에서 시맨틱 정보를 이용한 사물 탐색을 수행한다. 자체 온톨로지를 이용하여 사물들을 등록하고 시맨틱 정보로 구성되어 있는 사용자 질의에 따라 탐색을 수행하게 된다. 탐색을 할 때는 시맨틱 매칭을 이용하여 질의에 가장 부합하는 센서를 탐색하여 순위를 정한 다음 가장 높은 순위의 센서를 선택하게 된다. 이

[1] http://www.daml.org/

논문에서는 자체 온톨로지를 사용했기 때문에 다른 온톨로지 또는 도메인으로 등록이 되어있는 사물을 탐색하기에는 어려움이 있다.

CoAP 의 네트워크 환경은 사용자들이 주로 쓰는 HTTP 네트워크 환경과 차이점이 있다. 기본적으로 CoAP 은 이더넷(Ethernet) 네트워크가 아닌 6LoWPAN 환경으로 TCP/IP 계층 구조가 다르며 전송하는 메시지 포맷도 다르다. 기본적인 구조가 다르기 때문에 HTTP 네트워크에서 CoAP 네트워크와 통신을 할 경우에는 다음 그림 2 처럼 프록시 게이트웨이를 거쳐야 한다.

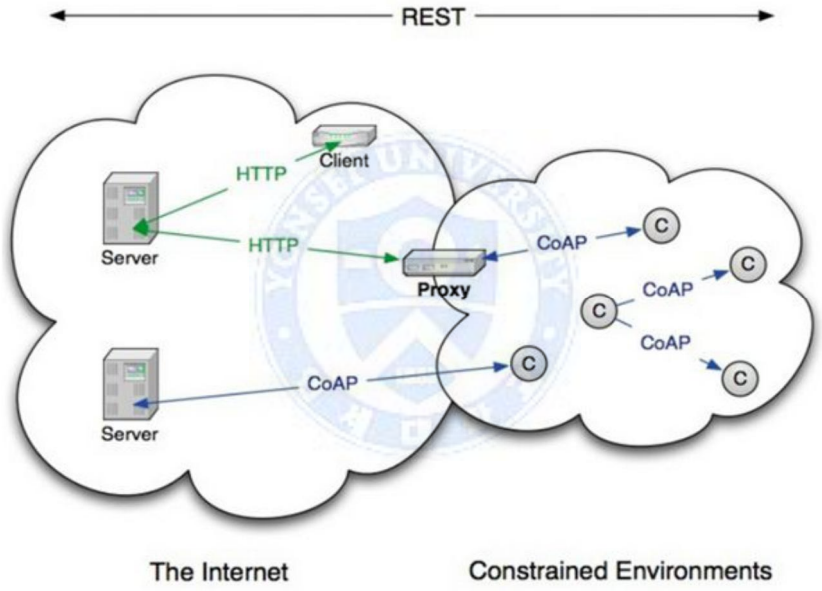

그림 2. CoAP Proxy (Core WG, 2010)

프록시 게이트웨이는 클라이언트로부터 요청된 URL 을 다른 네트워크 서버에 전달하게 되고 그에 대한 응답을 되돌려 받는다. 프록시가 필요한 상황은 여러 가지가 있다. 첫 번째 예로 스마트 폰에서 가정에 배치되어 있는 CoAP 네트워크 기반의 센서를 확인할 때, 스마트 폰의 HTTP 요청을 CoAP 으로

변환시켜주는 프록시가 필요하게 된다. 두 번째는 웹 브라우저와 같은 애플리케이션이 CoAP 은 지원하지 않고 HTTP 만 지원할 경우에 프록시가 필요하게 된다.

Mäenpää 등 [15]에서는 무선 센서 네트워크 환경에서 CoAP 노드를 이용하여 2G 망, 3G 망과 같은 다른 네트워크와 통신할 수 있는 환경을 제안하였다. CoAP 망과 다른 네트워크 망들과 네트워킹을 하기 위하여 게이트웨이와 프록시 노드를 네트워크에 배치하였다. 게이트웨이와 프록시 노드를 다른 네트워크와 연결을 하면서 동시에 REsource Location And Discovery (RELOAD)라는 오버레이 네트워크(Overlay Network)를 이용하여 근처의 센서들을 등록하고 탐색한다고 한다. 하지만 통신방법은 P2P(peer to peer) 방식으로 다른 센서 노드에 대한 정보를 미리 알아야 하며 네트워크 자체의 부하를 해결해야 하는 문제점이 있다.

Dijk 등 [16]에서는 HTTP 의 URL 을 CoAP URL 로 사상(mapping)하는 방법을 제안하였고 [17], [18], [19], [20], [21]에서 프록시 기능만이 아닌 사물의 식별 및 탐색 기능과 함께 웹 API 를 포함하고 있는 프레임워크들을 개발하였다. CoAP 과 HTTP 클라이언트/서버 간의 정확한 상호교환을 하기 위하여 프록시를 실행하는 방법을 보여준다. 요청을 전달할 때 forward proxy 를 통하여 HTTP 요청을 CoAP 요청으로 변환을 하게 되고, reverse proxy 를 통하여 CoAP 응답을 HTTP 응답으로 변환하게 된다. 그 외에도 CoAP 서버 및 클라이언트 개발이 많이 진행되었으며, 다음 표 1 은 CoAP 관련 개발을 정리한 표이다.

표 1. CoAP 구현 요약 정리

이름	개발 언어	CoAP 구현 정도	구현 특징
libcoap	C	Client & Server	Observe, Blockwise Transfers
iCoAP	Objective-C	Client	Observe, Blockwise Transfers
nCoAP	Java	Client & Server	Observe
jCoAP	Java	Client & Server	Observe, Blockwise Transfers
CoAPython	Python	Client & Server & Proxy	Observe, Multicast Server discovery, CoRE Link Format parsing
txThings	Python	Client & Server	Observe, Blockwise Transfers
TinyOS CoapBlip	nesC/C	Client & Server	Observe, Blockwise Transfers
Eribium for Contiki	C	Client & Server	Observe, Blockwise Transfers
Californium	Java	Client & Server	Observe, Blockwise Transfers, DTLS
Copper	JavaScript	Client	Observe, Blockwise Transfers
Microcoap	C	Client & Server	Core, Observe, Block, RD
ETRI CoAP	C	Client & Server	Core, Observe, Block,

하지만, 프록시의 방법은 멀티캐스팅 방식으로 요청을 보낼 경우에는 문제가 없지만, 응답을 받는 것에 있어서 문제점이 존재한다. 요청을 보낼 때는 HTTP 클라이언트에서 유니캐스트 방식으로 단 하나의 요청으로 각 CoAP 서버로 멀티캐스트 요청을 보내게 된다. 이때, HTTP 클라이언트에서는 유니캐스트 요청을 보냈기 때문에 한 개의 응답 받는 것을 예상한다. 하지만, 각 CoAP 서버에서는 멀티캐스트 요청으로 보내졌기 때문에 응답을 여러 개 받게 된다. 응답을 여러 개 받게 됨으로 네트워크 혼잡이 일어나며 네트워크 부하가 생기게 되며 시간이 걸리게 된다. 애플리케이션 계층에서 응답들을 합쳐서 처리하는 방법이 가능하나 이를 해결하기 위한 방법은 아직 미비한 실정이며 CoAP 명세 [19]에서 다루지 않고 있다.

Bovet 등 [22]은 IP 기반의 사물들이 존재하는 스마트 빌딩에서 건물의 자동화를 위한 시스템을 제안하였다. DNS 서버를 이용하여 각 구역의 건물과 건물 안의 층 그리고 방의 주소 체계를 자동으로 지정하였다. 네임 스페이스(name space)를 자동으로 지정하면서 각 구역의 건물들을 건물 별로 그룹으로 구성하였다. 멀티캐스트 방식을 사용하게 되는데 각 그룹의 업데이트 또는 탐색을 한다. 이 방식으로 인해 통신 값을 줄이게 되어 에너지 효율성을 높였다. 하지만 구체적인 그룹 조합에 대한 설명이 부족하고 도메인 모델과 온톨로지를 사용하지 않으므로 이 논문 또한 각각의 네임 스페이스에 대한 의미를 알 수가 없다.

Bovet 등 [23]은 기존의 자동화 시스템이 갖고 있는 서브 시스템 간의 이질적인 문제를 해결하기 위하여 웹 기반의 프로토콜을 제안하였다. 각기 다른 네트워크들과 인터페이스들을 통합하기 위하여 멀티 프로토콜 아키텍처를 제안하였다. 이 아키텍처는 CoAP, oBIX 등과 같은 여러 종류의 프로토콜이 사용 가능하며, 리소스의 식별, 연결(linking), 속성 기술, REST 서비스를 지원한

다. 또한 리소스들을 기술하는데 있어서 의미를 부여하기 위하여 온톨로지를 사용하였다. SSN 온톨로지를 사용하여 메타 데이터를 기술하였으며, RDF 를 이용하여 각 리소스들을 SPO (subject, predicate, object) 구조로 기술하였다. 하지만 온톨로지로 기술한 센서 정보를 구체적으로 활용하는 방법이 부족하며, 건물 자동화 시스템에서의 활용 방법도 제시되어 있지 않다.

제 3 장 CoAP 프록시 시스템

기본적으로 서로 다른 웹 서버간 통신하기 위해서는 클라이언트에서 목적지 주소를 알고 있거나 서버의 탐색 과정을 거쳐서 통신을 해야 한다. 마찬가지로 HTTP 네트워크 환경에서 CoAP 네트워크 환경으로 통신을 하기 위해서는 CoAP 서버의 목적지 주소를 알아야 한다.

Copper(Cu) [24]에서는 CoAP 리소스들과 상호 교환할 수 있는 웹 브라우저를 개발하였다. 사물 인터넷을 위한 웹 브라우저에서는 사용자들이 사물과 상호 정보를 교환을 하기 위한 서비스를 제공한다. CoAP 리소스 디렉토리인 /.well-known/core에 존재하는 리소스들을 브라우저 창에서 확인할 수 있으며 REST 기능들을 통하여 리소스들의 조회, 등록, 수정 및 삭제가 가능하다. 하지만 다른 도메인의 CoAP 서버 탐색이 불가능하므로 다른 CoAP 서버의 위치 정보에 대해서는 알 수가 없다. 또한 CoAP 리소스들만 탐색을 하고 CoAP 리소스 정보를 제공하기 때문에 HTTP 서버 또는 클라이언트와의 상호교환이 불가능하다. 이를 해결하기 위하여 본 논문에서는 HTTP 클라이언트에서도 CoAP 리소스들의 조회가 가능한 시스템을 제안한다.

본 논문은 사물의 리소스 정보들을 도메인 온톨로지 [25]와 SSN 온톨로지[2]를 기반으로 기술하는 시맨틱 표기법을 적용한다. 온톨로지를 사용하여 사물 정보를 기술하며 사물을 탐색 및 조합할 때 이용하게 된다. U-Campus 환경의 도메인 지식을 표현한 온톨로지를 이용하여 캠퍼스 내의 건물 실내와 실외의 위치를 사상한다. 도메인 온톨로지는 그림 3과 같이 표현되며, SSN 온톨로지와 관계가 형성이 되어 사물과 연결이 된다.

[2] http://purl.oclc.org/NET/ssnx/ssn

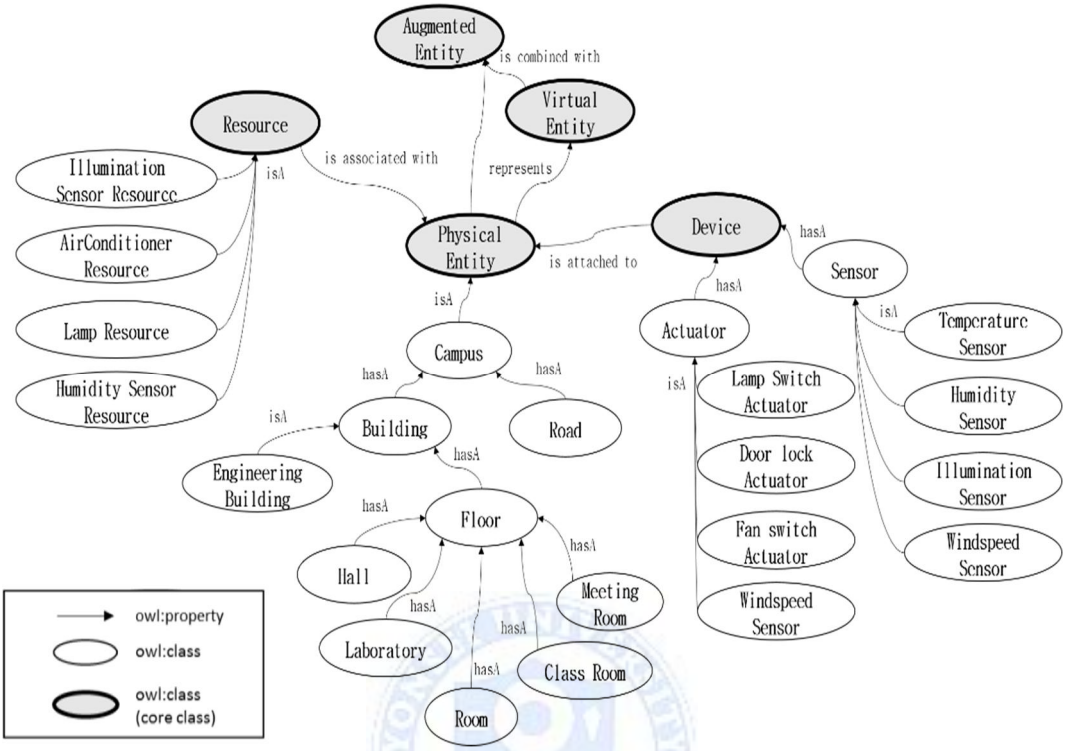

그림 3. 도메인 온톨로지 일부 예시

위의 도메인 온톨로지와 SSN 온톨로지의 관계를 생성하게 되면 그림 4와 같다. 그림 4는 SSN 온톨로지의 풍속 센서에 대한 예시이며 그림 3의 도메인 온톨로지를 SSN 온톨로지와 관계를 연결하여 확장시킨 시킨 예제이다. 온톨로지들은 CoAP 사물 정보들을 명세 하는데 이용이 되고, 탐색 및 그룹을 생성하는 과정에 사용한다.

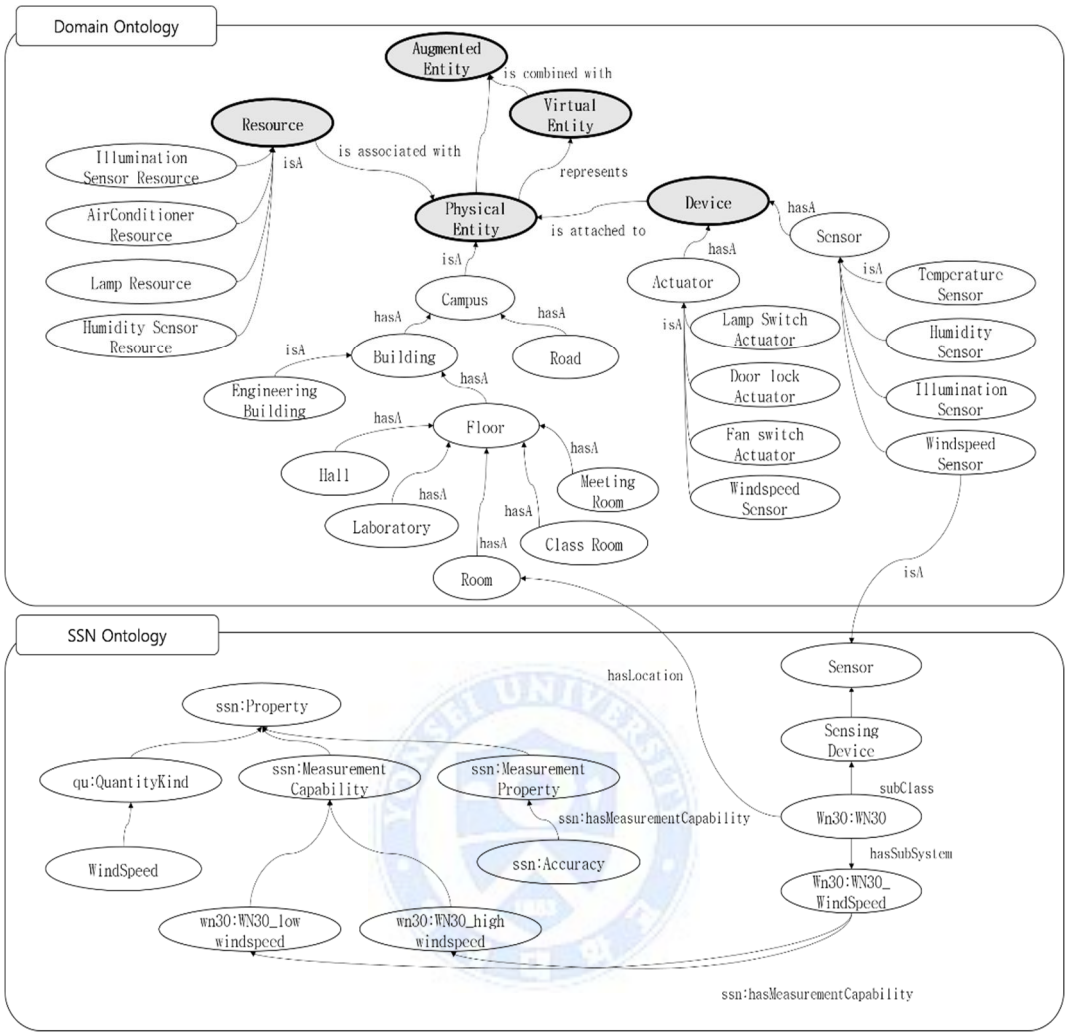

그림 4. 풍속 센서의 전체적인 온톨로지

본 논문에서 제안하는 시스템 구조는 그림 5와 같다. HTTP 네트워크 환경은 가장 많이 사용하고 있는 이더넷 망이 기본 층이 되어 TCP 통신을 하며, CoAP 네트워크 환경은 무선 센서 네트워크인 IEEE 802.15.4e망과 6LoWPAN이 기본 층이 되어 UDP 통신을 한다. 이 두 네트워크는 중간에 다리 역할을 하는 게이트웨이를 통해 상호교환을 한다. 이 시스템 구조의 주요 구성물은 3가지로 다음과 같다.

- 첫 번째는 사용자 질의를 생성하는 HTTP Client이며 웹 브라우저에서 질의를 생성하게 된다.

- 두 번째는 HTTP Server로 사용자 질의를 처리하여 CoAP 네트워크 환경으로 전송준비를 한다.

- 세 번째는 CoAP Proxy와 CoAP Server로 HTTP Server로부터 받은 요청을 변환하여 전달한 다음 CoAP Server에서 각 CoAP 센서로 전송한다.

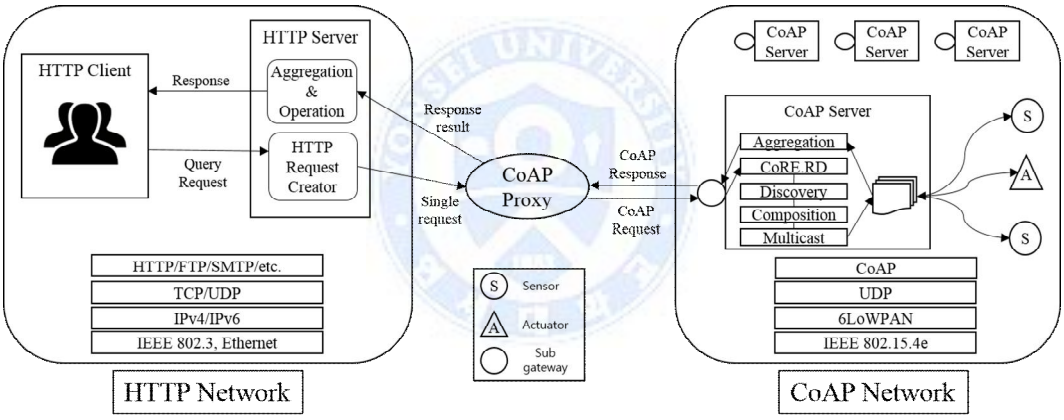

그림 5. 전체 시스템 구조

다음 절들은 시스템을 구성하는 주요 요소인 HTTP 클라이언트, HTTP SERVER, CoAP Proxy, CoAP Server 들에 대한 자세한 설명이다.

3.1. 시맨틱 메시지 생성

사용자가 사물을 탐색하거나 정보를 조회할 때, HTTP Client에서 첫 번째 과정이 수행된다. 첫 번째 과정은 시스템에 질의를 보내는 것으로 사용자가 탐색하고자 하는 위치와 사물을 입력한다. 사물이 구성되어 있는 위치와 사물의 종류를 입력하게 되며 사물의 특정한 기능을 수행할 때 추가적으로 기능을 입력한다. 사물 기능의 예로는 평균값 호출, 최대/최소값 호출 등이 있다.

본 논문에서는 사용자가 탐색하고자 하는 사물을 시스템이 이해할 수 있게 하기 위하여 시맨틱을 사용한다. 시맨틱 요청을 하기 위하여 요청 메시지의 포맷은 JSON[3] 을 사용한다. 기본적으로 JSON은 자바스크립트 객체를 표현하는 문자열 포맷 방식이다. JSON 포맷에 시맨틱을 강화하여 시맨틱 데이터를 확장시킨 포맷이 본 논문에서 사용할 JSON-LD가 된다. JSON-LD는 RDF (Resource Description Framework) 데이터를 표현할 수 있는 포맷으로 사람 또는 시스템이 이해할 수 있는 포맷이 된다. 따라서 일반적인 사용자 질의보단 더 정교한 질의를 시스템으로 요청할 수가 있게 된다.

```
1:  {
2:   “@context” :
3:   {
4:     physical entity: “http://icl.yonsei.ac.kr/uCampus#physicalEntity”,
5:     resource type: “http://icl.yonsei.ac.kr/uCampus#resourceType”,
6:     operation: “http://icl.yonsei.ac.kr/operation#”,
7:    },
8:   “location” : “engineering building 5th floor”,
9:   “resource type” : “temperature”,
10:  “operation” : “average”
11: }
```

그림 6. 사용자 요청 질의 예제

[3] http://www.json.org

3.2. 시맨틱 메시지 처리

HTTP 서버에는 2가지 구성물로 나뉘게 된다. 첫 번째는 HTTP Request Creator이며 두 번째는 Aggregate & Operation 이다. 이번 절에서는 2가지 구성물과 함께 각 입출력 값의 처리 방법에 대해 설명한다.

HTTP Request Creator는 탐색한 목적지 주소로 보낼 HTTP 요청 메시지를 생성한다. HTTP 요청 메시지의 생성방식은 클라이언트가 속한 호스트 주소와 목적지로 하고 있는 목적지(destination) 주소로 구성하여 생성하면 다음과 같은 형태로 생성된다.

http://{host-address}?coap_target_uri=coap://{target-address}/sensor

목적지 주소에 대한 질의를 하기 위하여 호스트 주소 뒷부분에 질의어를 붙이게 된다. 이때 질의 속성으로 'coap_target_uri' 속성을 사용하여 목적지 주소에 대해 질의를 한다. 위의 URI 형태를 이용한 예는 다음 예시 1과 같다.

예시 1)

http://www.server1.com?coap_target_uri=coap://www.coapserver2.com/temperature1

위의 예제는 server1에 속해있는 HTTP 클라이언트에서 coapserver2에 있는 온도센서1에 접속하는 예제로 이 주소를 전송하게 되면 단일 센서 목적지인 /temperature1로 요청을 보내게 되며 직접 전달하게 되는 것이다. 하지만 이는 사용자가 해당 센서에 대한 주소를 알고 있을 때의 경우가 되며, 본 논문에서는 각 센서의 명확한 URI를 알지 못하는 가정 하에 동작을 한다. 각 센서의 URI를 알기 위해서는 탐색을 수행해야 하는데, 탐색 과정에 대해서는 CoAP 프록시와 CoAP 서버 부분에서 설명할 것이다.

사물들은 하나의 기능만을 갖고 있는 사물이 아닌 다수의 기능을 갖고 있는 복합 사물들이 있다. 예를 들어, 온도와 관련하여 온도 감지 기능을 포함한 여러 기능을 갖고 있는 사물들은 온·습도계, 정수기, 냉장고, 에어컨 등이 있다. 이처럼 온도와 관련 있는 사물을 탐색하였을 때, 온도가 어느 사물의 온도인지 구분할 필요성이 있다. 사용자가 여러 사물이 있는 방 안의 온도를 요청하였을 때, 모든 사물의 온도 값을 주게 된다면 방 안에 있는 정수기 또는 냉장고의 온도 값을 전송하게 되는 문제점이 생긴다. 따라서 사물의 종류를 명확하게 구분하여 사용자가 원하는 요구 사항에 충족해야 한다. 기존의 CoAP 명세[1]에서는 리소스를 구분할 때, Resource type(rt)만을 이용하여 어떤 사물인지 구분하였다. 하지만 복합 사물일 경우에는 rt 만으로 구분하기가 어려우므로 속성을 나누거나 추가하여야 한다. 사물의 종류를 명확하게 구분하기 위하여 본 논문에서는 request에 속성을 추가하여 사용하며 속성들은 다음과 같다.

- rt (resource type): 기존에 있는 속성으로 사물이 측정하는 리소스를 의미한다. 각 사물들의 타입은 온톨로지를 이용하여 기술되어 있다.

- loc (location): 위치 속성으로 Physical entity에 사상되어 있다.

- op (operation): 사용자가 부가적으로 사물들이 측정한 값을 계산하거나 작동할 기능을 의미한다. (예, 평균값, 최대/최소값, on/off 등)

- cg (complex group): 새로운 속성으로 사용자가 생성할 그룹이 다수의 기능을 포함할 복합 그룹인지 여부를 의미한다.

위 속성들을 이용하여 특정 지역의 평균 온도에 대한 URI는 예시 2와 같다.

예시 2)

http://www.server1.com?coap_target_uri=coap://www.coapserver2.com?loc=engineeringbuilding&rt=temperature&op=average

온/습도를 감지하는 기능을 갖는 복합 그룹에 대한 예시는 다음과 같다.

예시 3)

http://www.server1.com?coap_target_uri=coap://www.coapserver2.com?loc=engineeringbuilding&rt=temperature&rt=humidity&cg=1

여러 서버로 요청을 보낼 경우에는 많은 숫자의 응답을 받게 된다. HTTP 클라이언트에서는 유니캐스트 방식으로 단일요청을 보내었기 때문에 단일 응답을 받을 것이라 생각할 것이다. 또한 한번에 여러 응답을 받게 될 경우에는 응답을 처리하는 메인 서버에서 네트워크 부하가 생기게 된다. 이런 문제들을 해결하기 위하여 본 논문에서는 여러 응답에 대한 Aggregation and Operation 모듈을 이용하여 통합 및 기능 작동을 하게 된다. HTTP 서버에서 받게 되는 다수의 응답 메시지들은 Aggregation and Operation 모듈에서 받게 된다. 받은 응답 메시지들은 사용자의 요구에 의해 통합을 하거나 특정 기능을 이용하여 응답 메시지들을 해당 기능을 수행하게 된다. 특정 기능에 대한 예시로는 평균, 최대, 최소 값 등을 구하는 기능이 있다.

3.3. CoAP 프록시와 CoAP 서버

CoAP 서버는 HTTP 서버와는 달리 한정적인 자원 환경에서 실질적으로 사물들이 연결되어 있는 서버이다. 각각의 사물들은 그림 5 하단의 네트워크 구조와 같이 IEEE 802.15.4 표준을 기반으로 통신하게 된다. 저전력, 저비용 기반인 IEEE 802.15.4 표준의 통신 반경은 10미터 내외가 되며, 10미터 이내의 공간을 Personal Operating Space(POS)라고 한다. POS에 무선 네트워크가 지원되는 사물이 들어오게 되면 사용자의 무선 개인 통신망(Wireless Personal Area Network)에 속하게 되어 데이터 통신을 하게 된다. 하지만 LoWPAN 신호는 송수신자간의 간섭이 없을 경우에 100미터까지 도달할 수 있다.

위의 환경에서 CoAP의 Sub Gateway는 각 특정 공간에 배치된 CoAP 센서와 무선 통신을 한다. Sub Gateway는 각 CoAP 서버마다 구성되어 있는 것으로 각 CoAP 서버의 입구 역할을 한다. HTTP 서버에서 전송하는 메시지를 받아 처리하여 각각의 CoAP 센서로 전달하게 된다. 이번 절에서는 각 서버에서 프록시 처리를 하여 메시지를 멀티캐스팅 하는 방법에 대해 설명한다.

앞서 HTTP request Creator에서 생성한 메시지를 받아와 CoAP Proxy에서 처리하게 된다. 가장 먼저 처리하는 것은 프록시 처리이다. 생성한 요청 메시지는 coap_target_uri 질의어를 중간에 두어 호스트 주소와 목적지 주소로 구분되어 있다. 호스트 주소는 목적지 주소에서 전송하는 응답 메시지를 전송하기 위하여 프록시에서 임시로 저장을 하고 목적지 주소는 사용자가 기술한 질의 내용을 처리하게 된다. 다음 과정은 질의를 처리하는 과정으로 사물의 탐색, 멀티캐스트 그룹 생성, 멀티캐스팅, 응답 처리 순으로 이루어진다.

3.3.1 사물 탐색

우선 사물의 탐색 방법이다. 각각의 CoAP 서버는 무선 네트워크 환경에서 사물들과 통신을 하게 되며, 각 서버마다 리소스 디렉토리가 존재하게 된다. Resource Directory에는 사물들의 정보가 있는 목록으로 각 사물들을 리소스 디렉토리에서 탐색한다. 리소스 디렉토리의 경로는 /.well-known/core로 지정되어 있으며 질의를 통하여 원하는 리소스를 탐색할 수 있다.

리소스 디렉토리 중 사물 또는 그룹을 탐색하는 lookup 기능이 있다. lookup 기능은 REST 서비스의 GET 메소드를 이용하여 호출하며 원하는 타입의 리소스를 탐색할 수 있다. 타입의 종류에 따라 사물 또는 그룹을 지정하여 호출이 가능하다. 또한 소수의 사물이나 그룹만이 아닌 다수의 특정 사물이나 특정 그룹에 대해서도 호출이 가능하다. 다음 예시 4는 특정한 그룹에 속하는 사물을 lookup 하는 예제이다. lights1의 그룹에 속해 있는 사물들을 탐색하는 것으로 요청 질의 문에 ep(endpoint)는 사물을 의미하며 그룹 명이 lights1인 사물들을 요청하는 것이다. 해당 질의에 대한 응답으로 lights1에 속하는 사물들의 URI와 사물 명이 나오게 된다.

예시 4) 특정 그룹의 사물을 호출하는 예제

Request: GET /rd-lookup/ep?gp=lights1

Response: 2.05 Content

<coap://host-address:port>;ep="node1",

<coap://host-address:port>;ep="node2"

다음 예시 5는 클라이언트가 존재하는 모든 그룹을 lookup 하는 예제로, 요청 질의 문의 /rd-lookup/은 리소스 디렉토리의 lookup 기능이며 gp는 그룹을

의미한다. 요청 메시지를 보내게 되면 응답 메시지로 상태 코드와 함께 탐색한 결과가 나오게 된다.

예시 5) 모든 그룹을 호출하는 예제

Request: GET /rd-lookup/gp

Response: 2.05 Content

</rd-group/12>; gp="lights1"; gp="lights2"

위의 예제인 그룹 타입에 대해 호출하는 것과 달리 속성 종류 (resource type(rt))를 온도와 같은 특정한 속성으로 지정하여 질의(rt=temperature)를 하게 되면 리소스 타입이 온도 센서인 사물만을 탐색한다. /.well-known/core{?rt} URI 형식에 질의를 넣게 되면 속성 종류에 해당하는 사물(endpoint)을 리소스 디렉토리에서 탐색하게 된다. 하지만 단순한 질의를 하게 될 경우에는 탐색 범위에 포함되는 사물의 수가 많아지게 되어 마찬가지로 response도 많아지게 될 것이다. 또한 복합 사물이 포함되어 있을 경우에는 사용자가 원하는 요구사항에 맞지 않는 상황이 생길 수도 있다. 이를 방지하기 위해 질의 내용을 확장 및 추가하여 탐색 범위를 줄이고 정교하게 사용자가 원하는 사물을 찾아야 한다. 탐색 방법으로는 시맨틱 매치메이킹 알고리즘[13]을 적용하여 요청 메시지 질의의 속성과 온톨로지에서 정의한 개념을 매칭하여 탐색하게 된다. 시맨틱 매치메이킹 알고리즘은 다음 그림 7과 같다.

Algorithm 1. 시맨틱 매치메이킹

```
1:  Input: Query Resource type (Query.rt),
2:         Resource Directory Resource type (RD.rt)
3:  Output: EXACT/SUBSUMEDBY/SUBSUMES/FAIL
4:  Procedure match(Query.rt, RD.rt)
5:
6:  Begin
7:  queryClass = getDataTypeClass(Query.rt)
8:  rdClass = getDataTypeClass(RD.rt)
9:
10: If queryClass ≡ rdClass then
11:  return EXACT
12: Else if queryClass ⊆ rdClass then
13:  return SUBSUMEDBY
14: Else if queryClass ⊇ rdClass then
15:  return SUBSUMES
16: else
17:  return FAIL
18: End if
19: End
```

그림 7. 탐색 과정을 위한 시맨틱 매치메이킹 알고리즘

위의 알고리즘을 이용하여 탐색하는 과정은 다음과 같다.

첫 번째는 우선 해당 질의를 만족하는 그룹을 찾는다. CoAP 서버의 리소스 디렉토리 중 그룹 디렉토리에서 질의에 명시되어 있는 속성(location, resource type)과 부합하는 그룹을 찾게 된다. 해당하는 그룹을 찾지 못하였을 경우 다음 과정으로 넘어간다.

두 번째 과정부터는 그룹을 조합할 수 있는 사물들이나 그룹들을 찾게 된다. 도메인 온톨로지에서 사용자가 질의한 위치를 구성할 수 있는 요소(건물, 층, 방)들을 찾아 그에 해당하는 그룹을 찾게 된다. 예를 들어, 어느 한 건물을 구성할 수 있는 층 정보들을 찾거나 층을 구성할 수 있는 방들을 찾는 것이다.

세 번째 과정은 두 번째 과정에서 찾은 구성 요소들 중에 찾을 수 없는 그룹이 있다면 그 그룹을 구성할 수 있는 사물들을 탐색하는 과정이다. Resource type을 이용하여 탐색을 수행하여 해당 위치의 그룹을 구성할 수 있는 모든 사물들을 찾는다.

위의 과정들은 다음 3.3.2절에서 사물들을 조합하는 알고리즘과 함께 자세히 보일 것이다.

3.3.2 사물 조합 및 Multicast 그룹 생성

다음은 멀티캐스트 그룹을 생성하는 단계로 앞서 탐색한 사물들은 각 CoAP 서버 단위로 조합되어 그룹을 구성할 수 있게 된다. 그룹을 구성하는 것은 멀티캐스트 주소와 그룹을 구성하는 그룹원 주소이다. 기본적인 그룹 구조는 다음 그림 8과 같이 구성되어있다.

Multicast Address	
Group type	Number of Sources(N)
Source Address[1]	
Source Address[2]	
Source Address[3]	
Source Address[4]	
...	
Source Address[N]	

그림 8. 멀티캐스트 그룹 구조

그룹을 생성하면 그룹은 그룹 특성에 기반한 주소를 생성하게 된다. 주소의 기본적인 구조는 coap://host-address:Port/Uri-path 형태로 구성이 되며, host-address은 해당 서버의 주소가 되며 포트 번호는 기본 번호인 5683이 된다. Uri-path는 그룹의 경로로 사용자가 질의한 내용에 기반하여 경로를 생성하게 된다.

생성된 그룹은 추후 같은 요청을 받았을 때 재사용하기 위하여 리소스 디렉토리에 추가된다. 기본적으로 그룹의 경로는 사물과 마찬가지로 리소스 디렉토리에 저장이 된다. 리소스 디렉토리의 그룹 디렉토리(/.well-known/core.gp)에서 해당 경로를 관리하게 된다. 리소스 디렉토리에 그룹을 등록, 수정, 삭제 및 조회 방법은 REST 메소드인 GET, PUT, POST, DELETE 메소드를 이용한다. 리소스 디렉토리에 그룹을 등록할 때, 위의 예시를 이용하여 설명하면 메소드

와 URI 형태는 POST /rd-group?gp=temperature1가 되며 리소스 디렉토리에 생성된 그룹 위치는 URI로 /rd-group/temperature1가 된다.

다음 그림 9는 위의 과정을 통해 그룹을 생성하는 알고리즘이다. 알고리즘에 기술된 개념들의 설명은 다음과 같다.

- Query: 알고리즘의 입력 값으로 사용자가 질의하여 생성한 질의
- rt (resource type): 질의 중 사물이 측정하는 리소스를 의미하는 변수
- loc (location): 탐색하려고 하는 사물 또는 그룹의 위치를 의미
- Group: 리소스 디렉토리에 존재하는 그룹
- Sensor: 리소스 디렉토리에 존재하는 센서
- match: 의미적으로 같은 정보를 갖는 변수인지 구별하는 함수
- GM: 멀티캐스트 그룹 리스트
- MA: 생성된 그룹의 멀티캐스트 주소

Algorithm 2. 사용자 질의에 대한 사물들의 시맨틱 조합

```
1:  Input: user request query(Query)
2:  Output: Multicast Group Address(MA)
3:  Procedure CreateMulticastList(Query)
4:
5:  Begin
6:  MA = checkURI(Query.loc, Query.rt);
7:  If MA is not null then
8:    sendCoAPMulticastRequest(MA);
9:  Else
10:   serverInfo = retrieveCoAPServers(loc, rt);
11:   If serverInfo is not null then
12:     locResult = match(Query.loc, RD.loc);
13:     If locResult is true then
14:        locList = addList(loc);
15:     While(locList is not null) do
16:       group = retrieveGroup(locList);
17:         If group is not null then
18:            rtResult=match(group.rt, Query.rt);
19:            If reResult is true then
20:              GM = addGroup(group);
21:            End If
22:         Else
23:            sensorList = retrieveSensors(locList, Query.rt);
24:            newGroup = addGroup(sensorList);
25:            GM = addGroup(newGroup);
26:         End If
27:     End While
28:     End if
29:     MA = CreateMulticastAddress(GM);
30:     sendCoAPMulticastRequest(MA);
31: End If
32: End
```

그림 9. 멀티캐스트 그룹 생성 알고리즘

그림 9는 질의를 통하여 그룹을 탐색하거나 조합하는 알고리즘이다. 위치 속성(loc)과, 리소스 속성(rt)을 질의하여 그룹 또는 사물이 매치되는지 여부

를 판단하여 탐색하게 되는 것이다. 6~8번째 줄은 질의에 해당하는 그룹 주소가 있는지 없는지를 확인하고 있으면 즉시 해당 URI에 질의를 보낸다. 그룹 주소가 없다면 9번째 줄부터 과정을 진행하게 된다. 10번째 줄에서 위치에 CoAP 서버를 탐색한 후, 11~12번째 줄에서 위치를 구성할 수 있는 하위 위치들을 탐색하여 리스트에 저장한다. 15~26번째 줄은 리스트에 저장한 각 위치마다 그룹(group)이 있는지 탐색을 한 후 있으면 그룹(GM)에 추가를 한다. 그룹이 없으면 사물들을 탐색하여 새로운 그룹(newGroup)을 생성한 다음 그룹(GM)에 추가를 하게 된다. 이 과정으로 생성한 그룹은 29번째 줄에서 URI 주소(MA)를 생성하고 30번째 줄에서 질의를 보낸다.

그룹의 생성은 사물(endpoint)들만의 조합이 아닌 그룹들의 조합이 가능하여 복합 그룹 생성을 할 수 있다. 복합 형태의 그룹은 같은 서로 다른 기능의 그룹을 조합하여 생성할 수 있다. 예를 들어, 서로 다른 기능의 그룹으로 온도를 감지하는 센서들의 그룹과 습도를 감지하는 센서들의 그룹을 조합하여 온·습도 그룹으로 통합할 수 있게 된다. 따라서 두 그룹을 통합하게 되면, 특정 지역의 온도 및 습도 정보를 한 번에 알 수가 있게 되는 것이다. 복합 그룹의 구조는 다음 그림 10과 같다.

Group of Group Address	
Group of Group type	Number of Sources(N)
Source Group Address[1]	
Source Group Address[2]	
Source Group Address[3]	
Source Group Address[4]	
…	
Source Group Address[N]	

그림 10. 복합 그룹 구조

3.4. 응답 메시지 처리

위에서 처리한 멀티캐스트 요청을 통해 다수의 센서에서 받아오는 응답 메시지들을 받아오게 되며 이를 처리하여야 한다. 이때 처리하는 방법이 두 가지 존재한다. 두 가지 경우는 받아온 응답 메시지의 사물 타입이 같은 경우와 다른 경우이다. 두 경우에 대한 설명은 다음과 같다.

첫 번째로 받아온 응답 메시지의 타입이 같은 경우는 같은 종류의 센서 값들만을 가져온 경우를 말한다. 예를 들어, 온도 센서들의 값들만 받아오는 경우가 있다. 값을 받아올 때 단일 센서에서 제공하는 단위가 각각 센서마다 다를 경우가 있다. 온도 센서 같은 경우에는 화씨(°F) 또는 섭씨(℃)를 제공하고 있으며 각 온도 센서들이 제공하는 값들의 온도 단위가 다를 경우에는 이를 같은 단위로 통합하여야 한다. 사용자의 단위 요구가 없을 경우에는 기본 단위를 사용한다. 본 논문에서는 섭씨 온도를 기본으로 사용하므로 화씨 온도를 전달하는 온도 값들은 섭씨 온도로 변환하여야 한다. 단위를 맞춘 온도 값들은 사용자의 요구에 따라 평균, 최대, 최소 값 등을 계산하여 보내게 되거나 단순 수치가 아닌 의미 있는 정보로 가공하여 사용자에게 보내게 된다.

두 번째는 받아온 응답 메시지의 타입이 다른 경우는 위의 경우와는 다르게 다른 종류의 센서 값들을 가져온 경우를 말한다. 복합 그룹의 온도, 습도, 가스 센서 등의 값들을 받아왔을 경우, 각각의 데이터들을 처리하여 새로운 정보를 만들어 사용자에게 제공할 수 있다.

위에서 처리된 응답 메시지는 최종적으로 클라이언트에게 제공되어 사물들의 데이터를 확인할 수 있게 된다. 다수의 사물들의 정보를 하나의 정보로 가공하여 보여주기 때문에, 클라이언트는 다수의 응답 메시지를 받지 않게 된다.

제 4 장 실험 결과

4.1 실험 환경

본 논문이 제안하는 CoAP 프록시 게이트웨이 및 사물들의 시맨틱 조합에 대한 실험 및 평가를 하기 위하여 시스템의 프로토타입을 설계하였다. HTTP 서버 환경에는 JSON-LD 기반의 사용자 질의를 생성하는 클라이언트와 생성한 질의를 HTTP 요청 메시지에 변환하는 모듈 그리고 CoAP Proxy에서 받아오는 응답메시지를 처리하는 모듈을 설계한다. HTTP 서버와 CoAP 서버를 연결하는 CoAP Proxy에는 각각의 메시지들을 변환시켜 전달하는 모듈을 설계하고 CoAP 서버 환경에는 여러 종류의 사물들을 시맨틱 조합할 수 있는 모듈을 설계하였다. 각각의 모듈에 전달되는 메시지의 설계 내용은 다음과 같다. 실험의 입력인 시맨틱 표기법이 적용된 질의 메시지와 더불어 도메인, 사물 인터넷 온톨로지를 각각 설계한다. 그리고 설계된 온톨로지들을 이용하여 실험에 사용될 사물들과 CoAP 서버를 설계한다. 사물들의 구성 및 설계 후, CoAP 프록시 게이트웨이를 이용하여 멀티캐스트 통신을 통한 각 사물들과의 통신과 응답 메시지를 처리하는 시스템을 설계한다.

도메인, 사물 인터넷 온톨로지들의 설계를 위하여 Protégé 4.3[4]을 이용하였다. 설계된 온톨리지들은 Apache Jena API 2.13.0[5]을 이용하여 리소스 디렉토리에 등록되는 또는 등록되어있는 사물들의 속성을 시맨틱 데이터들로 구성하게 된다. CoAP 서버와 리소스 디렉토리는 오픈 소스인 CoAP Framework Californium(Cf)[20]을 이용하여 본 논문에서 제안하는 방법론을 적용시켰다.

[4] http://protege.stanford.edu/

[5] http://jena.apache.org/

JAVA 기반의 프레임워크인 Californium을 Java SE 1.7.0_51과 Eclipse Kepler Service Release 2를 사용하여 수정하였다. 사물 인터넷 환경을 위해서 CoAP 서버와 통신하는 사물들은 아두이노 우노 키트 (Arduino UNO kit)와 센서들을 사용하고 각각의 사물들을 동작하기 위하여 Arduino IDE 1.0.5를 사용하여 센서들을 설계하였다. 아두이노 키트는 온도, 습도, 조도 센서 등 다양한 센서를 포함하는 감지 센서 장비와 LED 조명, 경보기 등 다양한 작동기 장비로써 실세계 데이터 및 동작하기에 적합한 장비이다. 아두이노 키트를 통해 수집한 데이터와 동작할 작동기들은 다수의 사물들이 연결된 환경을 만들기 위하여 각 사물 별로 복제를 만들어 가상화하였다.

4.1.1 실험 데이터

제안된 시스템이 얼마나 효과적인 멀티캐스트 통신을 하는지 평가하는 실험을 하기 위하여 다음과 같은 실험 데이터를 사용한다. 본 실험의 시나리오 환경은 캠퍼스 내의 건물이며 건물에 존재하는 다양한 사물들과 통신하는 환경이다. 건물은 기본적으로 상하의 구조를 갖고 있어서 건물, 층, 방 또는 복도라는 계층 구조를 갖게 된다. 방 또는 복도를 건물 위치의 가장 최하위 층으로 지정하여 각 사물들은 시맨틱 데이터인 위치 속성(physical entity의 location 속성)이 방(room) 또는 복도(hallway)를 갖게 된다. 그리고 사물들은 동작하는 기능에 따라 각기 다른 속성을 가지게 된다. 예를 들어 온도 센서는 temperature 속성을 갖게 되고 습도 센서는 humidity 속성을 각각 갖게 된다. 이 외에도 사물에 맞게 온톨로지의 개념과 속성을 갖게 된다.

실험 평가에 사용한 질의 내용은 다음과 표 2와 같다. 각 질의들은 특정 장소에 배치되어 있는 여러 사물들에게 질의를 보내는 것이다. 특정 장소는 앞서 설명한 것과 같이 건물내의 위치가 되며, 질의의 위치가 상위 계층으로 올라갈수록 더욱더 많은 사물들을 대상으로 메시지를 전송하게 된다.

표 2. 실험 질의 내용

질의 내용	
질의 1	공학관 5층 529호 온도
질의 2	공학관 5층 온도
질의 3	공학관 온도
질의 4	공학관 5층 온·습도
질의 5	공학관 온·습도
질의 6	공학관 5층 전등 켜기
질의 7	공학관 전등 켜기

4.2 시나리오 분석

첫 번째 시나리오는 질의 1부터 3까지의 내용은 단일 그룹에 대한 질의 내용으로 온도 값에 대해 요청 메시지를 보내는 것이다. 이 시나리오의 결과는 각 질의마다 기술되어 있는 위치에 해당하는 온도 값들을 받아오는 것이다. 질의 예제는 다음 그림 11과 같으며 앞서서 설명했듯이 질의에 들어가는 내용은 위치, 사물 종류 그리고 기능이다.

```
1:  {
2:   "@context" :{
3:    physical entity: "http://icl.yonsei.ac.kr/uCampus#physicalEntity" ,
4:    resource type: "http://icl.yonsei.ac.kr/uCampus#resourceType" ,
5:    operation: "http://icl.yonsei.ac.kr/operation#" ,
6:   },
7:     "location" : "engineering building" ,
8:     "resource type" : "temperature" ,
9:     "operation" : "average"
10: }
```

그림 11. 공학관 평균 온도 구하는 질의

두 번째 시나리오는 단일 그룹과는 달리 복합 그룹 시나리오로 사물의 종류가 두 개 이상일 경우의 시나리오이다. 질의 4, 5의 내용으로 해당하는 위치의 온·습도 정보들에 대해서 요청 메시지를 보내는 것이다. 질의 예제는 그림 12와 같으며 위의 질의 예제인 그림 11과의 차이는 resource type에 두 종류의 사물이 기입된 것이다. 시스템에서 이를 구별하여 각 위치에 해당하는 각 사물 그룹을 생성 또는 검색하여 온도 값과 습도 값들을 요청하게 된다. 따라서 이 복합 질의의 결과는 두 값들을 통합한 하나의 응답 메시지가 된다.

```
1:  {
2:      “@context” :{
3:      physical entity: “http://icl.yonsei.ac.kr/uCampus#physicalEntity”,
4:      resource type: “http://icl.yonsei.ac.kr/uCampus#resourceType”,
5:      operation: “http://icl.yonsei.ac.kr/operation#”,
6:      },
7:      “location” : “engineering building”,
8:      “resource type” :{
9:          “temperature”, “humidity”
10:     },
11:     “operation” : “average”
12: }
```

그림 12. 공학관 온, 습도 평균 구하는 질의

세 번째 시나리오는 앞의 두 시나리오와는 달리 정보를 얻어오는 것이 아닌 사물을 작동시키는 시나리오이다. 질의 6, 7의 내용으로 해당하는 위치의 사물들을 작동시키는 메시지를 보내는 것이다. 그림 13의 예제는 앞선 예제 그림 11과 내용은 비슷하지만 operation type이 on으로 되어 resource type에 명시되어 있는 illumination 사물을 on으로 동작시키는 것이다. 이에 대한 결과는 조명을 켜고 사용자에게 모든 조명들이 제대로 동작을 했는지 확인 시켜주는 응답 메시지를 보내게 된다.

```
1:  {
2:      “@context” :{
3:      physical entity: “http://icl.yonsei.ac.kr/uCampus#physicalEntity”,
4:      resource type: “http://icl.yonsei.ac.kr/uCampus#resourceType”,
5:      operation: “http://icl.yonsei.ac.kr/operation#”,
6:      },
7:        “location” : “engineering building 5th floor”,
8:        “resource type” : “illumination”
9:        “operation” : “on”
10: }
```

그림 13. 공학관 5층 조명 켜는 질의

4.3 실험 평가

4.3.1 자동 사물 조합의 유효성 검사 평가

우선 첫 번째 실험으로 질의를 통해 생성된 사물들의 그룹을 유효성(Validation) 검사를 통하여 질의를 정확하게 반영하여 그룹이 생성되는지 검사한다. 각각의 질의마다 생성되는 그룹들은 그룹간의 조합이 가능하면 그룹간의 조합을 하게 되고, 그룹 간의 조합이 되지 않는 경우에는 단순 조합이 된다. 이에 따라 시스템이 질의의 내용에 따라 정확하게 조합을 하였는지 검사를 하여 본 시스템의 정확성을 확인하고자 하였다. [4]에서도 그룹을 생성한 뒤 유효성 검사를 하였다. 하지만 본 논문의 그룹 생성 방식은 [4]와는 달리 온톨로지를 토대로 생성하였기 때문에 유효성 검사에 3.3.1절의 시맨틱 매치메이킹을 이용한 데이터 검사하는 과정으로 수행이 된다.

실험 결과인 표 3과 같이 각 질의 당 대한 유효성 검사를 100번씩 하여 그 평균을 구한 것으로 대부분 90% 이상의 정확성을 보였다. 검사가 모두 100%가 되지 않았던 이 UDP 메시지들을 전달하는 과정에서 패킷을 잃는 과정에서 오류가 생겼기 때문에 완벽한 정확도를 보이지 못하였다. 또한 각각의 결과를 살펴보면 조합되는 사물의 양 또는 그룹의 양이 많아질수록 수치가 감소하는 것을 확인할 수 있다. 이는 검사 중 더 많은 양의 사물 또는 그룹을 확인하는 메시지 전달 과정에서 패킷을 잃었기 때문에 정확도가 낮아졌다.

표 3. 유효성 검사 정확도

	질의 1	질의 2	질의 3	질의 4	질의 5	질의 6	질의 7
정확도(%)	99.8%	98.8%	93.9%	98.5%	90.2%	99.0%	94.0%

4.3.2 그룹에 따른 트래픽 양 평가

기존의 방법론인 CoAP [6]의 기본적인 멀티캐스트 방식보다 효과적인 멀티캐스트 통신을 보이기 위하여 두 번째 실험은 본 논문의 사물 그룹 방법론과 비교하였다. 단순하게 사물들만 조합하여 그룹을 만든 것과 그룹 간의 그룹을 이용하여 조합한 그룹들과 비교를 한 결과는 다음 그림 14와 같이 모두 기존의 방법보다 우월한 성능을 가짐을 확인하였다.

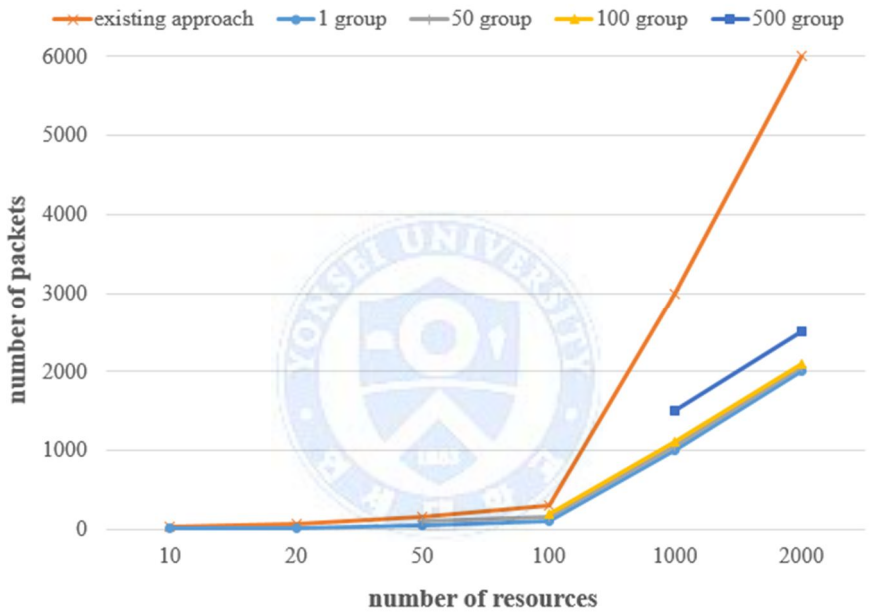

그림 14. 그룹의 정도에 따른 패킷 수

그룹들의 조합 정도에 따라 다른 패킷 수의 결과를 확인하였지만 전부 기존의 방법보다 낮은 양의 패킷 수를 보냈기 때문에 트래픽 오버헤드 문제의 해결이 가능함을 보였다. 하지만 사물의 수에 따라 생성할 수 있는 그룹에 제한이 있기 때문에 다음 실험 4.3.3절에서 그룹의 수를 줄여 그룹 정도에 따른 결과를 평가하였다.

4.3.3 그룹의 정도에 따른 응답 시간 평가

생성된 하나의 그룹 안에 그룹 정도에 따라 전달되는 메시지 양이 많아지거나 줄어들게 된다. 그룹 안에 그룹이 많아질수록 보내지는 요청 메시지들은 줄어들게 된다. 그에 따라 각 그룹에서 전달받는 요청 메시지들의 양이 현저히 줄어들기 때문에 처리하는 메시지 양이 줄어든다. 이를 확인하기 위하여 세 번째 실험은 각 질의에 대한 사물들을 조합한 후, 그룹의 정도에 따라 메시지를 처리하는 시간을 확인하였다. 질의 내용은 그룹이 가장 많이 생성되는 질의 5로 실험을 하였다. 질의 5로 그룹을 생성할 때 각각 그룹 안의 그룹 양을 다르게 하여 응답 시간을 측정하였다. 그 결과 다음 그림 15와 같은 결과를 확인할 수 있었다.

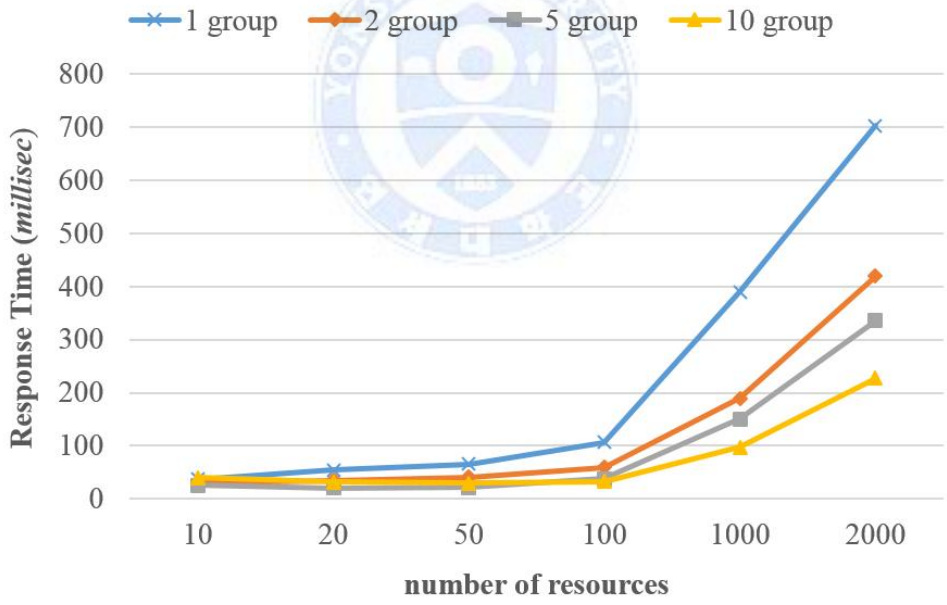

그림 15. 그룹 정도에 따른 응답 시간

사물의 양이 100개 이하일 때는 그룹의 정도에 따른 응답 시간이 100ms 안으로 많은 차이를 보이진 않았다. 그러나 사물의 양이 그 이상으로 늘어났을 경우에는 각 응답 시간의 차이가 크게 나는 것을 확인할 수 있었다. 이 결과를 통하여 많은 양의 사물들과 멀티캐스트 통신을 할 때 그룹 안에 그룹이 많을수록 처리하는 시간이 빠른 것을 확인하였다.

4.3.4 유니캐스트와 멀티캐스트의 비교 평가

네 번째 실험으로는 같은 환경에서 본 논문에서 제안한 멀티캐스트 방법과 유니캐스트 기반의 방법 차이를 확인하는 것으로 멀티캐스트 통신의 성능을 파악하고자 하였다. 요청 메시지의 수는 1부터 100까지 두 가지 통신 방법 전부 같은 수로 지정하여 실험을 진행하였다. 그 결과는 다음 그림 16과 같이 멀티캐스트 통신 방법이 유니캐스트 통신 방법보다 걸리는 시간이 더 적은 것을 확인하였다. 요청 메시지가 적은 경우에는 차이가 적었지만, 메시지의 수 50개 이상으로 커졌을 때 유니캐스트 방법의 시간이 급격하게 상승하여 두 가지 통신 방법의 큰 차이를 보인다.

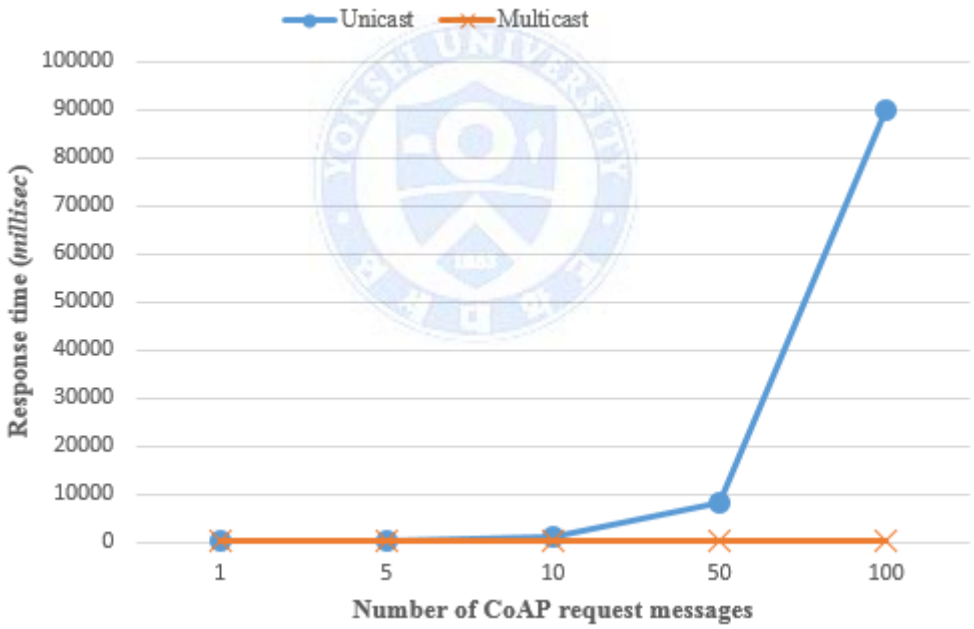

그림 16. 유니캐스트 방법과 멀티캐스트 방법의 경과 시간

제 5 장 결론 및 향후 연구

다양한 사물들이 연결되어 있는 사물 인터넷환경에서 각각의 사물들과 통신을 하면서 에너지 소모율을 줄이고자 하는 연구가 제안되었다. 그 중에서도 자원이 한정적인 환경에서 통신하기 위한 프로토콜로 CoAP이 HTTP 프로토콜 대체로 표준화되었다. CoAP의 특징으로는 한정적인 자원 환경에서 작은 크기의 메시지로 통신을 하여 에너지 소모를 낮춘다. 또한 멀티캐스트 통신을 제공함으로써 다수의 사물들과 동시에 통신할 수 있는 특징이 있다. 하지만 멀티캐스트 통신을 할 때, 서로 다른 네트워크 환경간 통신시 다수의 응답 메시지에 대한 병목 현상이 생기기 마련이다. CoAP [6]에서는 이 문제에 대해 다루지 않고 있으며 트래픽 오버헤드가 발생하게 된다. 또한 멀티캐스트 통신을 하기에 앞서 그룹을 미리 생성해야 하는데 그룹의 생성은 항상 사용자가 직접 그룹에 등록해야 한다.

이러한 문제점들을 해결하기 위하여 CoAP 프록시 게이트웨이와 시맨틱 사물 조합 시스템을 제안하였다. 서로 다른 네트워크 환경인 HTTP와 CoAP 환경 간에 메시지를 주고 받으며 멀티캐스트 통신의 병목 현상을 처리하기 위한 프록시 게이트웨이를 제시하였다. 또한 시맨틱 질의를 기반으로 사용자의 질의를 분석하고 사용자의 요구를 정확히 반영하는 그룹을 자동 조합하는 방법을 제시하였다. 그 결과 높은 정확도를 유지하며 사용자의 요구를 정확히 반영하는 그룹을 자동으로 조합하게 되었으며 기존에 수동으로 그룹을 만든 방법보다 편리성과 시간을 적게 들일 수가 있었다. 자동으로 조합한 그룹을 이용하여 멀티캐스트 통신을 하게 되는데, CoAP 프록시 게이트웨이에서 다수의 응답 메시지들을 한 개의 메시지로 처리하게 된다. 그리하여 실험 결과에서는 트래픽 데이터의 감소와 응답 시간을 감소하게 되어 병목 현상을 피할 수가 있었다.

그러나 제안하는 CoAP 프록시 게이트웨이와 전체적인 시스템에서 다양한 종류의 사물들을 처리하기 위하여 더 많은 요소들을 고려해야 한다. 예를 들어 조명을 작동할 때 있어서 어떤 조명은 on/off로 작동을 하지만 어떤 조명은 수치에 의하여 작동을 할 수가 있다. 이와 같이 두 가지의 작동 값을 갖는 사물이 한 그룹에 있을 경우를 처리하기 위하여 동시 처리 방안이 필요하다. 또한 다양한 operation을 할 수 있는 온톨로지 설계를 통해 사용자에게 다양한 기능을 제공할 필요가 있다. 마지막으로, 실험 환경의 가상화가 아닌 더 큰 실제 환경을 구성하여 다양한 응용의 실험 결과를 보여줄 필요가 있다.

참고문헌

[1] D. Miorandi, S. Sicari, F. De Pellegrini and I. Chlamtac, “Internet of Things: Vision, Applications and Research Challenges,” Ad Hoc Networks, Vol. 10, No. 7, pp. 1497-1516, 2008.

[2] BACnet, http://www.bacnet.org/.

[3] Knx association, http://www.knx.org/.

[4] Enocean, http://www.enocean.com/en/home/.

[5] L. Pérez-Lombard, J. Ortiz and C. Pout, “A review on buildings energy consumption information,” Energy and buildings, Vol. 40, Issue. 3, pp. 394-398, 2008.

[6] Z. Shelby, K. Hartke, C. Bormann and B. Frank, “The Constrained Application Protocol (CoAP),” Internet Engineering Task Force, https://datatracker.ietf.org/doc/rfc7252/.

[7] A. Rahman and E. Dijk, “Group Communication for the Constrained Application Protocol (CoAP),” Internet Engineering Task Force, https://datatracker.ietf.org/doc/rfc7390/.

[8] Z. Shelby, S. Krco and C. Bormann, “CoRE Resource Directory,” http://tools.ietf.org/html/draft-shelby-core-resource-directory-04/.

[9] I. Ishaq, J. Hoebeke, F. Van den Abeele, I. Moerman and P. Demeester, “Group communication in constrained environments using CoAP-based entities,” In Proc. the 2013 IEEE Int’l Conf. Distributed Computing in Sensor Systems, pp. 345-350, 2013.

[10] I. Ishaq, J. Hoebeke, F. Van den Abeele, J. Rossey, I. Moerman and P. Demeester, “Flexible Unicast-Based Group Communication for CoAP-

Enabled Devices," Sensors, Vol. 6, pp. 9833-9877, 2014.

[11] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox and K. Taylor, "The SSN ontology of the W3C semantic sensor network incubator group," Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 17, pp. 25-32, 2012.

[12] H. Stephan, S. Alexandru, B. Marin and C. Francois, "A Domain Model for the Internet of Things," In Proc. the 2013 IEEE Int'l Conf. on Internet of Things (iThings), pp. 411-417, 2013.

[13] M. Paolucci, T. Kawamura, T. R. Payne and K. Sycara, "Semantic matching of web services capabilities," The Semantic Web-ISWC 2002. Springer Berlin Heidelberg, pp. 333-347, 2002.

[14] M. Ruta, F. Scioscia, G. Loseto, F. Gramegna, A. Pinto, S. Ieva and D. Sciascio, "A Logic-based CoAP Extension for Resource Discovery in Semantic Sensor Networks," In the 11th Int'l Semantic Web Conference, pp. 17-32, 2012.

[15] J. Mäenpää, J. J. Bolonio and S. Loreto, "Using RELOAD and CoAP for wide area sensor and actuator networking," EURASIP Journal on Wireless Communications and Networking, Vol. 1, pp. 1-22, 2012.

[16] E. Dijk, A. Rahman, T. Fossati, S. Loreto and A. Castellani, "Guidelines for HTTP-CoAP Mapping Implementations," https://datatracker.ietf.org/doc/draft-ietf-core-http-mapping/.

[17] "Californium (Cf) CoAP framework in Java," ETH Zurich, http://people.inf.ethz.ch/mkovatsc/californium.php/.

[18] jCoAP, http://code.google.com/p/jcoap/.

[19] WebThings, https://github.com/koanlogic/webthings/.

[20] Nodes, http://code.google.com/p/nodes/.

[21] “Squid 3.1.9 with transparent HTTP-CoAP mapping module,” http://telecom.dei.unipd.it/iot/.

[22] G. Bovet and J. Hennebert, “A Distributed Web-based Naming System for Smart Buildings,” In Proc. of IEEE Int’l Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1-6, 2014.

[23] G. Bovet, A. Ridi and J. Hennebert, “Toward Web Enhanced Building Automation Systems,” In Big Data and Internet of Things: A Roadmap for Smart Environments, Springer Int’l Publishing, pp. 259-283, 2014.

[24] M. Kovatsch, “CoAP for the web of things: from tiny resource-constrained devices to the web browser,” In Proc. of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication. ACM Press, pp. 1495-1504, 2013.

[25] S. Chun, J. Jung, X. Jin, G. Cho, J. Shin and K.-H. Lee, “Short paper: Semantic URI-based event-driven physical mashup,” In Proc. of the IEEE World Forum on Internet of Things (WF-IoT), pp. 195-196, 2014.

ABSTRACT

Enhancing CoAP Proxy for Semantic Composition and Multicast Communication

Cho, Gunhee
Dept. of Computer Science
The Graduate School
Yonsei University

With the development of the Internet of Things (IoT), the number of smart things is increasing rapidly. Due to the resource constrained nature of things, an efficient mechanism for managing things and processing their data is required. The Constrained Application Protocol (CoAP) is proposed as a substitution for HTTP since the compressed size of CoAP messages is more suitable for IoT. CoAP provides multicast communication with various things concurrently. However, multicast communication has a disadvantage for handling multiple responses when they travel through a CoAP proxy. Multiple responses might cause a traffic overload and even a client might expect a single response from a single multicast request.

In this paper, we propose a novel approach to constructing a CoAP proxy, which supports multicast communication efficiently and groups things based on the semantic descriptions of requests and things. After sending a multicast request to a group of resources, the proposed proxy merges multiple responses to an aggregated one and sends a single response to a client. In order to demonstrate the feasibility of the approach we present an implementation and evaluate the performance of the proposed approach.

Key words : Internet of Things, Constrained Application Protocol(CoAP), proxy, Multicast communication, Semantic composition