유비쿼터스 컴퓨팅환경에서의 다양한 응용시스템 접속을 위한 메세징미들웨어에 관한 연구

A Study on the messaging middleware for various interfacing of application systems in the ubiquitous computing

김한욱, 조위덕, 강경란, 이정태, 최동순, 백경원

Department of Information and Communication Engineering ,
Ubiquitous System R&D Center Ajou University
San 5, Wonchun-Dong, Yeongtong-Gu, Suwon, 443-749, Rep. of Korea
Tel: +82-31-219-2441, E-mail: {hanoogi,chowd,korykang,jungtae,gooday,beeback}@ajou.ac.kr

아주대학교 정보통신전문대학원 정보통신공학과,
유비쿼터스시스템연구센터
경기도 수원시 영통구 원천동 산5 ,443-749

요약

유비쿼터스 컴퓨팅 환경은 다양한 장치 혹은 시스템들이 동적으로 결합하여 구성되는 환경이라고 할 수 있다. 이러한 장치 혹은 시스템들이 연동하여 사용자들을 위한 서비스를 제공할 수 있도록 하는 효율적인 기반을 구축하는 것이 중요하다. 본 논문에서는 특정한 운영 체제나 개발 환경을 가정하지 않고 다양한 환경에서 개발된 장치들이 쉽게 통합되어 연동될 수 있도록 하는 메시징 프로토콜을 제안한다.

1. 서론

유비쿼터스 컴퓨팅 환경은 "서로 상이하고 이동이 가능한 서브 시스템들이 동적으로 조화를 이루어 상호 작용하는 환경"이라고 정의할 수 있다. 즉, 컴퓨팅 환경을 구성하는 서브 시스템들이 동적으로 들어오고 나가는 것이 가능하며 그들의 협력 관계도 동적으로 변화할 수 있다. 이러한 환경에서의 상호 작용을 지원하기 위한 방법론으로 본 논문에서는 메시징 프로토콜을 제안한다. 메시징 프로토콜은 이미 기존의 미들웨어[1,2]에서 널리 사용되어 온 통신 기법이다. 본 논문에서 제안하는 메시징 프로토콜이 갖는 특징은, 유비쿼터스 환경에서 발생하는 상호 작용에 대한 모델을 기반으로 가능한 한 서로 상이한 환경에서 개발된 서브 시스템의 효율적인 상호 작용 모델을 지원하는 것을 주된 목적으로 한다는 것이다.

본 논문에서 제안하는 메시징 프로토콜은 유비쿼터스 컴퓨팅 환경을 추상화한 '메시지 버스' 구조를 가정한다. 메시지 버스에는 메시지 전달을 관리하는 '메시지 브로커'와, 서로 상이한 환경에서 개발된 개별 시스템의 특성을 완충하는 '어댑터'가 존재한다. 메시징 프로토콜에서는 메시지 브로커와 어댑터, 그리고 서브 시스템과 메시지 브로커, 서브 시스템 간에 교환하는 메시지들을 크게 응용 메시지 상황 정보 메시지, 미가공 데이터 메시지, 제어 메시지 등으로 분류하며, 이들의 기본 형식을 정의한다. 메시지의 종류로는 이 메시지들은 그 용도에 따라 필요한 정보 요소가 달라질 수 있다.

메시지 프로토콜은 다양한 플랫폼을 사용하여 구현할 수 있는데, 본 논문에서는 기존의 상용 미들웨어인 Tibco® RVD™ [3] 상에서 구현하는 방법론을 제시한다. 본 논문의 메시징 프로토콜의 주된 목표인 서브 시스템의 다양성을 지원하기 위해 서브 시스템의 컴퓨팅 능력에 따라 타당한 구현 방식을 선택하여 적용할 수 있도록 한다.

본 논문의 2장에서는 메시지 버스의 모델 및 메시징 프로토콜이 적용될 유비쿼터스 컴퓨팅 환경의 모델에 대해 기술하고, 3장에서는 메시징 프로토콜을 설계하는데 있어서의 고려 사항을 기술하고, 4장에서는 메시지 프로토콜의 기술 명세 사항을 자세하게 기술한다. 그리고, 5장에서는 구현 방안에 대해 기술하며, 6장에서 앞으로의 추가 연구 방향을

제시하는 것으로 전체 논문을 마무리한다.

2. 유비쿼터스 메시지 버스

본 논문에서 고려하는 유비쿼터스 컴퓨팅 환경의 구성 요소와 메시지 버스에 대한 개념적인 모델은 그림 1과 같다. 유비쿼터스 컴퓨팅 환경에서 다루어져야 할 논리적인 환경은 다양하지만, 본 논문에서의 유비쿼터스 컴퓨팅 환경은 집을 기본 환경으로 가정한다. 집 안의 장치들을 관리하는 **Home station**과 집 안에서 사용자를 대상으로 하는 응용 서비스인 **Home application**, 집 안 내 동적인 서비스 구성을 위한 **Agent group**, 집 안 내 환경 정보를 수집하기 위한 센서 네트워크, **RFiD** 네트워크 등이 있으며, 사용자가 갖고 이동하는 무선 단말기, 그리고, 마지막으로 집 안의 상황 정보를 판단하고 공지하는 역할을 수행하는 **Context manager**로 구성된다.

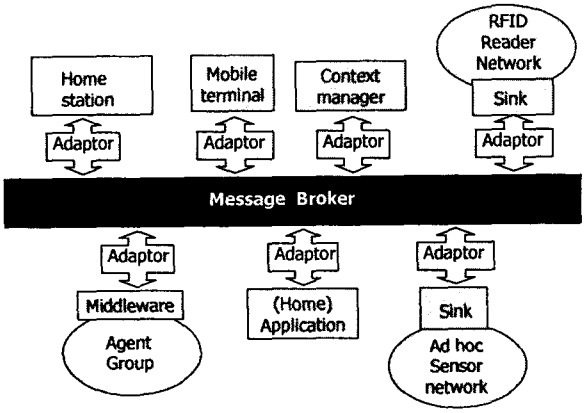

그림 1. 유비쿼터스 컴퓨팅 환경과 메시지 버스의 개념 모델

이들은 메시지 버스를 사용하여 통신하는데, 메시지 버스는 '메시지 브로커'와 '어댑터'로 이루어진다. 메시지 브로커는 기본적으로 메시지를 전달하는 역할을 담당하고, 어댑터는 개별 서브 시스템이 메시지 브로커와 연동할 수 있도록 서브 시스템의 특성을 보완하는 역할을 담당한다. 서브 시스템들은 유비쿼터스 환경 내에 들어오거나 다른 환경으로 이동하는 것이 가능하며, 상호작용 시에 서브 시스템들 관점에서는 상대 서브 시스템과 직접 통신하는 것으로 여길 수 있지만, 실제로는 어댑터를 거쳐 메시지 브로커를 거쳐, 즉, 메시지 버스를 통하여 이루어지게 되는 것이다. 이러한 메시지 브러커나 어댑터라는 개체를 채택하게 된 것은, 서로 다른 환경에서 서로 다른 개발자에 의해서 개발되는 서브 시스템들이 용이하게 상호 연동하도록 지원하기 위해서는, 중간에 서브 시스템들 간의 상이점을 보완해 주는 과정이 필요하다는 판단하였기 때문이다.

메시지 브로커와 어댑터는 기본적으로 메시지 브로커가 서비스하는 유비쿼터스 컴퓨팅 환경 내에서 어댑터가 서비스하는 서브 시스템이 동작하는 한 연결을 계속해서 유지한다고 가정한다. 유비쿼터스 컴퓨팅 환경의 물리적인 범위가 충분히 커서 유비쿼터스 컴퓨팅 환경 내에서의 이동이 네트웍 계층 혹은 데이터 링크 계층 등에서의 핸드오프를 유발한다고 하더라도, 해당 계층에서의 이동성 지원 방법론에 의해 연결이 계속해서 유지될 수 있다고 가정한다. 다시 정리하면, 본 논문에서 고려하는 메시지 버스는 프로토콜 스택의 관점에서 볼 때 전송 계층과 응용 계층 사이에 위치하므로, 논리적인 연결을 유지하고 관리하는데 있어서, 전송 계층 이하에서의 이동성 지원 기능[4,5]을 활용한다.

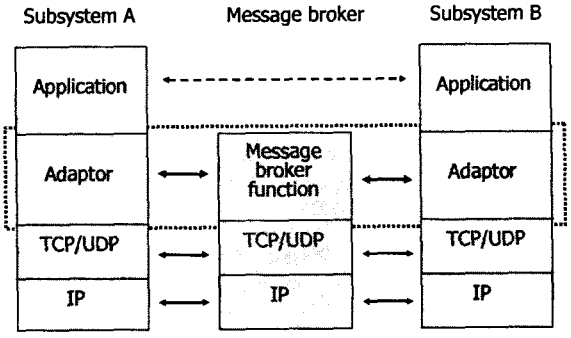

그림 2. 메시지 버스 프로토콜 스택 상에서의 위치

3. 메시지 프로토콜 설계 고려 사항

가장 기본적으로 요구되는 기능은, 크게 세 가지를 들 수 있다. 첫째, 유비쿼터스 컴퓨팅 환경의 서브 시스템이 들어 오고 나가는 등의 변화로 인하여 동적으로 재구성되는 상황을 지원할 수 있어야 한다 둘째, 재구성된 상태에서의 서브 시스템 간의 효율적인 상호 작용을 지원할 수 있도록 한다. 그리고, 마지막으로, 서브 시스템들의 자발적인 의사에 의해 그룹을 구성할 수 있도록 하며, 그룹 내에서의 효율적인 상호작용이 이루어질 수 있도록 지원해야 한다.

첫번째 요구 사항을 만족시키기 위해서는, 메시지 버스를 통해서 상호 작용하는 서브 시스템이 무엇이 있는지 관리가 이루어지되, 이에 대한 부담을 충분히 완화하여야 한다. 동적으로 들어 오고 나가는데 처리 부담이 크다면 이를 처리하는데 전체 컴퓨팅 능력을 소진하게 될 수 있기 때문이다.

두번째 요구 사항을 만족시키기 위해서는, 수신자 혹은 송신자 정보를 동적으로 처리할 수 있어야 한다. 즉, 메시지를 송신할 때 메시지 수신자를 구체적으로 명시하지 않아도 메시지 버스에서 이를 처리하여 적당한 수신자에게 전달해 줄 수 있어야 하며, 메시지를 수신하고자 할 때 송신 주체에 대해서도 구체적으로 명시하지 않아도 메시지 버스에서 이를 적당한 송신자로 대응시켜 줄 수 있어야 한다. 그래서, 본 논문에서 제안하는 메시지 프로토콜은 일대일 통신뿐만 아니라 publish/subscribe 기반의 상호 작용을 지원한다.

세번째 요구 사항을 만족시키기 위해, 그룹을 생성하고 가입하고 탈퇴하며 폐쇄하는 절차에 해당하는 메시지들을 정의한다. 또한, 그룹에 가입하고 탈퇴하는 절차를 제한할 수 있도록 하며, 메시지를 그룹 내에 속한 서브 시스템만 수신할 수 있도록 메시지 전달 범위를 제한할 수 있는 방법론을 지원한다. 그리고, 유비쿼터스 컴퓨팅 환경 내에 동적으로 들어오고 나가는 서브 시스템들이 그룹 정보를 검색하고 선택할 수 있도록 하는 방법론도 함께 지원한다.

4. 메시지의 유형

4.1 응용 메시지 (Application message)

4.1.1 용도

서브 시스템 혹은 서브 시스템에서 동작하는 응용 프로그램의 요구에 따라 만들어지는 메시지들을 가리킨다. 예를 들어, 응용 프로그램을 동작시키기 위해 혹은 응용 프로그램에서 필요한 서비스를 동작시키기 위해 보내지는 요구(request)와 이에 대한 응답(response) 메시지들이 있다.

4.1.2 정보 요소 (Information elements)

메시지 내의 데이터의 형식은 응용 프로그램에 따라 달라질 수 있을 것이고, 메시지의 내용을 기술하기 위한 정보 요소들도 응용 프로그램에 의해서 결정될 것이다. 하지만, 임의의 응용 프로그램에 대해서도 공통적으로 요구되는 정보 요소들로는 다음과 같은 것들이 있다.

- Source: 메시지 송신 서브 시스템
- Destination: 메시지 수신 서브 시스템
- Application class: 메시지 안에 포함된 데이터를 해석하기 위한 응용프로그램 분류에 대한 정보

4.2 상황 정보 메시지(Context message)

4.2.1 용도

Context manager와 같이 상황 정보 데이터 (context data)를 생성하는 서브 시스템이 상황 정보를 사용하여 인지된 상황 정보를 전송하는 메시지들을 지칭한다. 이러한 메시지에는 다음의 정보 요소들이 포함되어야 한다.

4.2.2 정보 요소

- Source: 누가 생성한 상황 정보인가를 표시한다. 상황 정보 데이터를 생성하는 역할은 Context manager 수행하는 것이 아니고, 다른 Home station 등과 같은 서브 시스템에서도 생성하는 것이 가능하므로, 소스를 명확하게 표시하는 것이 필요하다.
- Date: 상황 정보 메시지 안에 포함된 상황 정보 데이터가 생성된 시점을 표시한다
- Environment: 메시지 안에 포함되는 상황 정보 데이터가 어떤 환경에서 발생한 것인지를 표시한다. 소스와 중복되는 정보가 될 수도 있으나, 상황 정보의 종류에 따라서는 소스가 갖고 있는 이름에서 표현하지 못하는 정보들이 있을 수 있다.
- Expiry: 이 메시지 내에 포함된 상황 정보 데이터의 유효 기간을 표시한다. 이를 참조하여 메시지 브로커가 상황 정보 데이터 메시지를 종료시점까지 보존하다. 새로운 상황 정보 데이터 메시지가 publish되기 전에 해당 상황 정보를 subscribe하는 서브 시스템에게 전달한다.

기본적으로 상황 정보 메시지는 상황 정보 데이터를 원하는 모든 서브 시스템에 전송되는 것을 원칙으로 하므로, 대상을 명확하게 표시하지 않는다. 그리고, 메시지 브로커에 해당하는 위치의 상황 정보 데이터를 수신하겠다고 요청하는 서브 시스템들 모두에게 전달될 것이다.

4.3 미가공 데이터 메시지(Row data message)

4.3.1 용도

센스 네트워크의 싱크(sink) 나 RFiD 리더 네트워크의 싱크에 수집되는 데이터들은 특별한 처리과정이 적용되기 이전의 상태일 것이고, 이러한 데이터들을 수신하여 Context manager나 Home station 등에서 처리하고 상황 정보 데이터 등과 같은 가공된 데이터를 만들어 내게 될 것이다. 이렇게, 센스 네트워크의 싱크나 RFiD 리더 네트워크의 싱크가 Context manager나 Home station에게 수집된 미가공 데이터를

전달할 때 쓰이는 메시지가 미가공 데이터 메시지이다.

4.3.2 정보 요소

이 메시지에 포함되어야 할 정보 요소들에는 다음과 같은 것들이 있다.

- Source: 미가공 데이터를 전송하는 서브 시스템을 표시한다. 소스의 이름에 포함된 위치 정보만으로도 해당하는 미가공 데이터가 수집된 위치를 파악할 수 있다. 다만, 보다 상세한 위치 정보가 필요하다면, 추가적인 정보 요소를 정의할 수 있을 것이다.
- Data type: 미가공 데이터 내에 포함된 데이터의 내용에 대한 힌트를 제공한다.
- Data length: 미가공 데이터이고 이진 (binary) 데이터일 것이므로 데이터 길이 정보가 제공되어야 수신 서브 시스템에서 무리가 없이 메시지를 처리할 수 있다.
- Date: 메시지 안에 포함된 미가공 데이터가 수집된 시점을 표시한다.
- Expiry: 이 메시지 내에 포함된 상황 정보 데이터의 유효 기간을 표시한다. 이를 참조하여 메시지 브로커가 미가공 데이터 메시지를 종료시점까지 저장한다. 새로운 상황 정보 데이터 메시지가 publish되기 전에 해당 미가공 데이터를 등록하는 서브 시스템에게 전달한다.

4.4 제어 메시지(Control messages)

4.4.1 용도

서브 시스템이 메시지 버스에서 제공하는 서비스를 활용하기 위해 메시지 broker와 교환하는 메시지들을 지칭한다. 이러한 제어 메시지는 다음과 같은 경우에 사용되고, 각 용도 별로 정의된 메시지는 다음과 같다.

① 서브 시스템이 메시지 버스에 추가되거나 삭제될 때, (Register, Deregister 등)
② 유비쿼터스 컴퓨팅 환경 내에 동작하는 타 서브 시스템 정보를 요청할 때 (SystemListQuery, SystemListReseponse, PropertyQuery, PropertyResponse 등)
③ 일시적인 그룹을 생성하고, 가입하고, 탈퇴하고, 멤버쉽 목록 등을 요청할 때 (CreateGroup, JoinGroup, LeaveGroup, DeleteGroup, GroupQuery 등)
④ 자신 서브 시스템의 속성(property)를 갱신할 때 (PropertyUpdate 등)
⑤ 유비쿼터스 컴퓨팅 환경 내에서 발생하는 상황 정보 메시지와 미가공 메시지 수신을 요청할 때 (Subscribe, Unsubscribe 등)

4.4.2 정보 요소

제어 메시지의 헤더는 제어 메시지가 요청하는 처리의 종류와 처리 결과를 한 눈에 파악할 수 있도록 하는 정보를 포함하도록 하며, 처리의 보다 자세한 요청이나 처리 결과에 대한 보다 상세한 설명은 본문(body)에 포함시키는 것을 원칙으로 한다. 그래서, 모든 제어 메시지들은 다음과 같은 정보 요소를 메시지의 헤더 내에 포함한다.

- Source: 제어 메시지의 송신자를 표시한다.
- Destination: 제어 메시지의 수신자를 표시한다.
- Message id: 요청에 대한 응답을 구별하기 위한 용도이다. 응답 메시지는 요청의 메시지 id와 동일한 값을 사용해야 한다.
- Operation name: 서브 시스템이 메시지 브로커에게 요청하는 작동(operation)이나 서브 시스템이 메시지 브로커에 요청하는 작동이 표시된다. 이 작동 이름은 메시지 브로커에 의해서 결정되는 경우가 대부분일 것이며, 새로운 작동이 정의되면 메시지 브로커는 역시 제어 메시지를 사용하여 환경 내의 서브 시스템들에게 공지할 것이다.
- Operation parameter: 작동을 수행하기 위해 필요한 추가적인 정보를 표시한다. 여기서는, 작동에 따라 변수를 XML의 형태로 표현하는 것이 가능하다.

4.5 상호 작용 시나리오

4.5.1 Bootstrap 절차

앞서 정의된 메시지들을 사용하여 서브 시스템이 유비쿼터스 컴퓨팅 환경에 들어와서 메시지 브로커와 연결되는 과정은 그림 3과 같다.

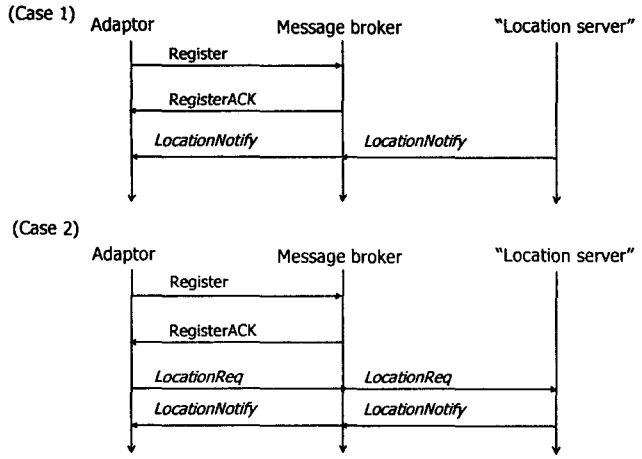

그림 3. Bootstrap 절차 예

두 경우의 차이는 새로 들어온 서브 시스템의 위치 정보를 메시지 브로커가 필요에 의해 위치 정보 서비스하는 서브 시스템에게 요청해서 획득하게 되는 것인지 아니면 위치 정보를 서비스 서브 시스템이 메시지 브로커에게 일방적으로 통지하는가 하는 점에서 차이가 있다. 다만, 각 서브 시스템이 직접 메시지 브로커와 상호 작용하는 것이 아니고, 서브 시스템이 갖고 있는 어댑터가 메시지 브로커와 상호작용한다는 점에서 동일하다.

4.5.2 Context manager과 sensor sink와의 연동

Context manager는 집 안에서 발생하는 환경 정보를 수신하여 상황 정보를 파악하는 기능을 수행한다. 그림 4는 context manager가 메시지 브로커를 통하여 센서 네트워의 싱크에서 publish하는 미가공 메시지를 수신하는 과정을 보이고 있다. 그림에 있는 Register, RegisterACK, SystemListQuery, SystemListResponse, Subscribe, SubscribeACK 등은 메시징 프로토콜 내에서 정의된 제어 메시지이며, Raw Data는 미가공 메시지를 나타낸다.

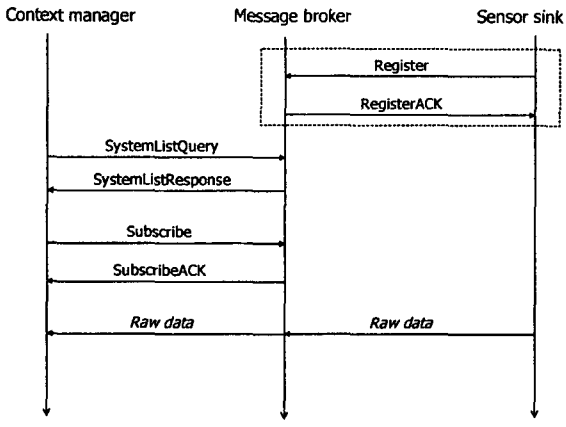

그림 4. 미가공 데이터를 사용한 상호 작용 예

4.5.3 Home application과 Home station, context manager와의 연동

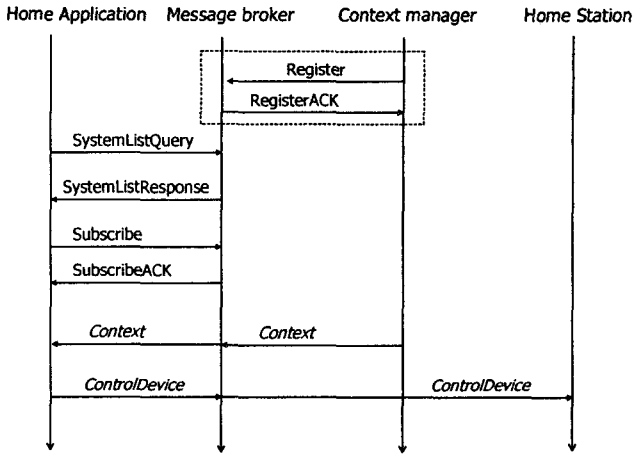

그림 5. 응용 메시지를 사용한 상호 작용 예

그림 5는 Context manager가 publish한 상황 정보 메시지를 수신하여 Home station이 집 안의 특정 서비스를 운용하기 위해 Home application과 연동하는 과정을 보이고 있다. Home station이 Home application에게 전송하는 'ControlDevice'가 앞서 정의한 응용 메시지에 해당한다.

4.5.4 그룹 관리 절차

그림 6은 그룹 관리를 위한 상호 작용의 예를 보이고 있다. 서브 시스템 간에 합의에 의해 그룹을 만들기로 결정을 하면, 그룹 창시자가 메시지 브로커에게 CreateGroup이라는 제어 메시지을 전송하여 그룹 정보를 생성하고, 나머지 구성원들은 JoinGroup이라는 제어 메시지를 사용하여 가입한다. 이 때, 그룹 가입 정책에 따라 JoinGroup 메시지 내에 가입 허락을 위한 요소들이 포함될 수 있다. 그리고, 그룹 구성원에 대한 정보를 전체 구성원이 공유할 것인지 아닌지에 따라 신규 가입 때마다 기존의 구성원에게 공지할 수도 있고 아닐 수도 있다. 그룹에서 탈퇴하고자 하면 LeaveGroup이라는 제어 메시지를 사용하며, 그룹 자체를 삭제하고자 할 때 DeleteGroup이라는 제어 메시지를 사용한다. 그리고, 그림 6에는 표시되지 않았지만, 기존의 그룹이 무엇이 있는지 확인하기 위한 제어 메시지로 GroupQuery라는 제어 메시지가 정의된다.

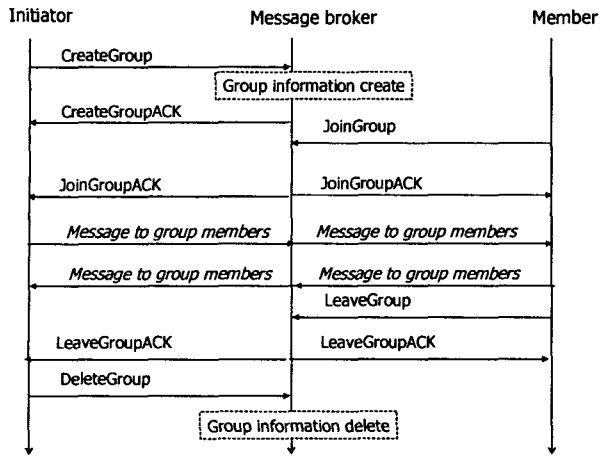

그림 6. 그룹 관리 예

4.6 XML 기반의 메시지 형식

메시지 형식은 기본적으로 SOAP (Simple Object Access Protocol)[6]에서 정의하고 있는 형식에 따른다. 즉, 헤더(header)와 본문(body)으로 나뉘어지는 것으로 하며, 헤더와 본문은 XML[7]태그로 구분된다. 그리고, 정보 요소들도 마찬가지로 XML 태그로 구분된다. 메시지 형식을 단순하게 표현하면, 그림 7과 같다.

```
<header>
<attr1>value1</attr1>
<attr2>value2</attr2>
</header>
<body>data<body>
```

그림 7. 단순화된 메시지 형식

<header>와 </header>사이에는 메시지 전달에 필요한 정보 요소들을 갖고 <body>와 </body> 사이에는 송신 서브 시스템에서 수신 서브 시스템에게 전송하고자 한 데이터가 포함되며, 그 형식은 메시지의 종류에 따라 달라질 수 있다. 응용 메시지의 경우에는 메시지의 송수신 서브 시스템의 채널로 그 형식에 대한 합의가 이루어져 있어야 할 것이다.

4.7 헤더의 정보 요소

메시지의 종류에 무관하게 모든 메시지에 기본적으로 포함되어야 할 정보 요소들은 다음과 같다.

① Source: 송신 서브 시스템의 이름이 올 수 있다.

② Destination: 수신 서브 시스템을 명시하는 경우에 포함되며, 수신 서브 시스템의 이름 혹은 id가 올 수 있다.

③ Message scope: 해당 메시지가 전송되어야 할 범위를 지시한다. 'Destination'이 명시된 경우, 특히 destination이 전체 이름(full name)으로 명시된 경우에는 메시지 전송 범위를 제한할 필요가 없다. 하지만, destination의 이름이 상대 이름(relative name)이거나, destination이 명시되지 않은 경우에는 이 message scope이 명시되어야 한다.

④ Message id : 송신 서브 시스템에서 부가하는 일종의 일련번호 혹은 문자들이다. 추후 요청에 대한 응답이거나 특정 메시지에 대한 오류 등이 발생했을 때, 해당 메시지를 구분하기 위한 용도이다.

⑤ Date: 메시지를 전송하는 시점이 될 수도 있고, 메시지 내에 포함된 데이터가 생성된 시점을 표시할 수 있다.

⑥ Message type: 메시지 브로커에서 해당 메시지를 처리하는 과정을 돕기 위한 것으로 어떤 분류에 속하는지를 표시한다.

⑦ Content description: 메시지 내의 내용에 대한 힌트를 메시지 브로커에게 제공하기 위한 것이다. 물론 수신 서브 시스템에서도 이를 참고할 수 있다. Content description에 필요한 속성을 명시하는 태그와 값의 리스트들이 포함될 것이다. 메시지의 종류에 따라 속성들이 달라질 수 있으며 동일한 종류의 메시지라 하더라도 속성들이 달라질 수 있다. 전자의 경우는, 제어 메시지는 작동 이름과 작동 파라메타들이 포함되지만, 다른 종류의 메시지에는 이러한 속성들이 필요가 없고, 또한 상황 정보 메시지에는 환경 정보가 포함되지만, 다른 종류의 메시지에는 필요하지 않은 경우를 가리킨다. 그리고, 후자의 경우는, 응용 메시지에서 응용프로그램의 종류에 따라 content description에 포함되어야 할 속성이 달라지는 경우를 지칭한다.

그림 8는 헤더 정보 요소를 모두 포함한 응용 메시지의 헤더의 예를 보이고 있다.

```
<header>
    <source>
        <where>
            <city>suwon</city>
            <street>uman 2</street>
            <bldg>202</bldg>
            <room>806</room>
        </where>
        <who>1548518</who>
    </source>
    <destination>
        <where>
            <city>suwon</city>
            <org>ajou univ.</org>
            <bldg>dasan</bldg>
            <room>505 </room>
        </where>
        <who>2162623</who>
    </destination>
    <message-type>application
    </message-type>
    <date>2004.7.4.08.35.00</date>
    <message-id></message-id>
    <content-description>
        <type>image</type>
        <subtype>jpg</subtype>
    </content-description>
</header>
<body><data>[binary data]</data>
</body>
```

그림 8. 메시지 헤더 예

5. 구현

본 논문에서 제안하는 메시징 프로토콜을 Tibco® Rendezvous™을 사용해서 구현한다. 상용이나 공개된 메시지 버스 개발 환경들이 많이 있지만 구현의 용이성이나 보급 현황 등을 고려하여 Tibco® Rendezvous™을 선택하였다. 그러나, 본 논문에서 제안하는 메시징 프로토콜은 다른 개발 환경에서도 구현이 가능하다.

본 논문에서 제안하는 메시지 버스가 Tibco® RVD™ 상에서 동작하므로, 메시지들이 그림 9에서 보이는 바와 같이 다시 Tibco® 메시지 안에 인캡슐레이션된다. 어댑터와 메시지 브로커 사이에서 교환되는 메시지는 TIBCO® Rendezvous™에서 정의하는 메시지 헤더를 헤더로 갖고 어댑터에서 생성한 메시지가 본문(body)에 위치할 것이다. 그리고, 어댑터가 생성한 메시지는 서브 시스템이 전송하고자 한 데이터가 본문에 위치하고 앞서 4장에서 정의한 기본 정보 요소들이 헤더에 위치할 것이다. 그래서, 궁극적으로 서브 시스템에게 전달되는 데이터는 메시지의 가장 안쪽에 위치할 것이다.

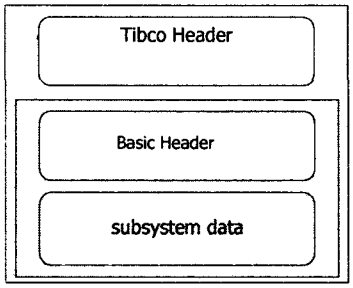

그림 9. 메시지 인캡슐레이션

메시지 브로커에서 유지하고 관리해야 하는 정보에는 다음과 같은 것들이 있다.

- 서브 시스템 목록: <이름, 속성 정보 URI, 논리 채널 번호, publish하는 데이터 제목>
- 그룹 목록: <그룹 이름, 그룹 창시자, 참가 서브 시스템, 범위, 신규 가입 멤버 인증 정책 구성원 변화 공지 정책>
- 가입 목록: <데이터 제목, subsystem lists>

  여기서 '데이터 제목'은 단순한 문자열이 아니라 <attribute><value>의 짝으로 표현될 수 있으며, 'attribute'에는 송신 서브 시스템 이름, 데이터 형식, 메시지 종류, 환경 이름 등이 올 수 있는데, 이는 context manager가 생성하는 상황 정보 데이터나 센서 네트워크로부터의 미가공 데이터를 표현하는 기법에 따라 달라질 수 있다.
- Publish된 메시지 보관소: <송신 서브 시스템 이름, 시각, 데이터 제목, 유효 기간, 저장 위치>

  Context manager가 생성하는 상황 정보 데이터나 센서 네트워크의 미가공 데이터는 데이터의 생성 주기 동안에는 유효하므로, 필요에 따라 이러한 데이터를 유효 기간까지 일시적으로 저장한다.

그림 10은 메시지 버스를 서브 시스템에 적용하는 모델의 예를 보이고 있다. 기본적으로 메시지 브로커의 역할은 TIBCO® Rendezvous™의 RVD가 담당하고, 각 서브 시스템은 어댑터를 구현하여 갖고 있다.

그리고, 각 서브 시스템의 컴퓨팅 능력에 따라 예1, 예2, 예3 중에 선택하여 구현할 수 있다.

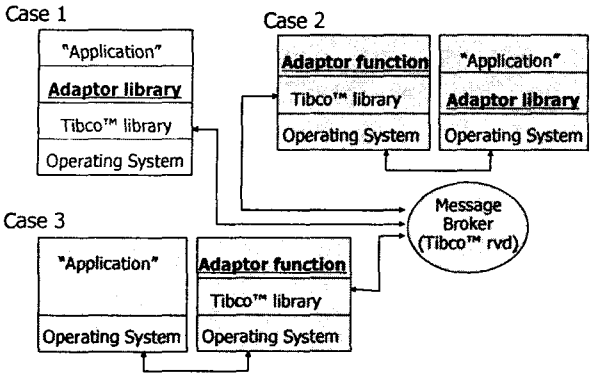

그림 10. 메시지 버스 적용 모델

5.1 구현 예 1

서브 시스템이 Home station, Context manager, 센서 네트워크의 싱크, Home application 등과 같이 고성능 CPU, 대용량메모리 등을 갖는 환경에서 구현되는 경우를 고려한 것이다. 어댑터가 프로그래밍 라이브러리의 형태로 주어지고, 서브 시스템 개발자는 이 API를 사용하여 메시지를 송신하고 수신하는 기능을 구현한다. 이 때, 메시지 브로커가 TIBCO® Rendezvous™을 사용하므로 메시지 브로커와 어댑터는 TIBCO® Rendezvous™의 라이브러리를 사용하여 메시지를 교환한다.

5.2 구현 예 2

이동 단말기 등과 같이 낮은 컴퓨팅 능력을 갖는 서브 시스템을 고려한 경우이다. 메시지 변환이나 메시지 브로커와의 상호 작용을 담당하는 어댑터가 독립된 프로세스로 존재하고, 서브 시스템에 제공된 어댑터 라이브러리는 이 어댑터와 연결을 설정하고 해지하며 메시지를 받고 보내는 정도의 단순한 기능을 수행한다.

5.3 구현 예 3

기본적으로 다른 서브 시스템과의 상호 작용을 단순한 TCP/IP를 기반으로 한 소켓 프로그램이 아닌 HTTP[8]와 SOAP의 결합 등과 같이 TCP/IP의 상위 응용 계층 프로토콜을 사용하는 서브 시스템을 지원하기 위한 경우이다. 어댑터가 HTTP 서버와 같은 형식으로 구현되어, 서브 시스템이 HTTP를 사용해서 자신이 전달하고자 하는 메시지가 있으면 어댑터에게 HTTP로 접속하여 메시지를 POST한다. 어댑터는 이 메시지를 적절하게 변형하여 메시지 브로커에게 전달하고, 최종적으로 수신 서브 시스템에게 전달될 것이다.

6. 결론

유비쿼터스 컴퓨팅 환경은 기본적으로 환경의 구성 요소의 다양성과 환경의 동적인 변화를 가정한다. 본 논문에서는 이러한 환경에서의 효율적인 상호 작용을 방법론을 제시하는 것이 본 연구의 목적이다 본 연구는 지속적으로 진행 과정에 있으며, 중간 연구 결과로서 본 논문에서는 메시지 버스의 모델, 메시징 프로토콜, 그리고 구현 방안에 대해서 기술하였다. 메시징 프로토콜에서는 상호 작용의 기본적인 유형을 모델링하여 제어 메시지, 응용 메시지, 상황 정보 메시지, 미가공 메시지로 나누어 이들을 활용한 상호 작용의 예도 함께 제시하였다. 그리고, 구현 방안으로는 상용 메시징 시스템 개발 환경인 Tibco® Rendezvous™을 사용하여 구현하는 방안과 서브 시스템의 컴퓨팅 환경에 따라 적용 가능한 구현 모델을 제안하였다.

본 연구는 본 논문에서 기술하고 있는 메시징 프로토콜과 구현 모델에 근거하여 구현 중이며, 조만간 구현을 완료하여 그 성능을 평가할 수 있을 것이며, 이를 기반으로 보다 최적화된 메시징 프로토콜 및 메시지 버스를 개발하게 될 것이다. 그리고, 본 논문에서 제시하고 있는 메시징 프로토콜이나 구현 모델은 본 연구에만 국한되는 것이 아니고 동적인 환경 변화나 상이한 개발 환경을 기반으로 하는 유비쿼터스 컴퓨팅 환경에 쉽게 적용하여 활용이 가능할 것으로 판단된다.

감사의 글

본 논문은 정통부의 21세기 프론티어 사업을 수행하는 (재)유비쿼터스컴퓨팅사업단(사업단장; 조위덕)의 유비쿼터스컴퓨팅 및 네트워크 원천기술개발사업의 지원으로 연구되었음

참고문헌

[1] Jörg Ott, Colin Perkins and Dirk Kutscherm, "The Message Bus: A Platform for Component-based Conferencing Applications," Proc. of CSCW2000 workshop on Component-Based Groupware; December 2000.

[2] Sun Microsystems, Java Message Service, http://java.sun.com/developer/technicalArticles/Networking/messaging/

[3] TIBCO, *TIBCO Rendezvous (TM)*, http://www.tibco.com/software/enterprise_backbone/rendezvous.jsp

[4] C. Perkins, et al., IP Mobility Support for IPv4 , IETF RFC 3344, August 2002.

[5] R. Droms, *Dynamic Host Configuration Protocol(DHCP)*, IETF RFC, March 1997

[6] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S. and D. Winer, *Simple Object Access Protocol (SOAP) 1.1*, May 2000.

[7] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen and Eve Maler, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation. eds., October 2000.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *Hypertext Transfer Protocol -- HTTP/1.1*, IETF RFC 2616, June 1999.