유비쿼터스 멀티미디어 응용을 위한 CMQ 미들웨어 프레임웍의 구현

최 태 욱[†] · 정 기 동[††]

요 약

종래의 응용들은 하나의 컴퓨터상에서 한정된 자원만을 사용하고 컨텍스트를 고려하지 않으므로 이동하는 사용자를 서비스하기 힘들다. 그러나 유비쿼터스 응용은 사용자 주변의 컨텍스트 정보와 여러 컴퓨터에 있는 다양한 자원을 이용하여 이동 사용자에게 최적의 서비스를 제공한다. 이를 위해 유비쿼터스 응용들은 컨텍스트 인지(context awareness), 사용자 이동성(user mobility), QoS 적응성(QoS adaptability) 등의 기능을 가져야 한다. 본 연구는 유비쿼터스 응용을 위한 CMQ(Context-aware, Mobility-aware, QoS-aware) 미들웨어 프레임웍을 설계하고 자바(Java) 와 지니(Jini) 기술을 이용하여 미들웨어 프로토타입을 구현한다. 구현된 미들웨어 시스템은 사용자 주변의 다양한 컨텍스트 정보를 효과적으로 표현 및 처리할 수 있고, 사용자의 이동에 따라 세션 핸드오프를 지원하며 적응적 QoS 조절 메카니즘을 응용에게 제공할 수 있다.

Implementation of the CMQ Middleware Framework for Ubiquitous Multimedia Applications

Tae Uk Choi[†] · Ki Dong Chung[††]

ABSTRACT

Traditional applications are executed using the restricted resources of a single computer, do not consider contextual information, and can not support mobile users. However, ubiquitous applications provide optimal services for mobile users using the various resources of computers and the contextual information around users and devices. Thus, ubiquitous applications need to have the functionality of context awareness, user mobility and QoS adaptability. This paper design the CMQ(Context-aware, Mobility-aware, QoS-aware) middleware framework for ubiquitous applications and implement the middleware framework using Jini and Java. The implemented middleware system can process various contexts, provide the session handoff for a mobile user, and allow applications to adjust its QoS dynamically.

키워드 : 유비쿼터스 응용(Ubiquitous Application), 미들웨어(Middleware), 컨텍스트 인지(Context Awareness), 사용자 이동성(User Mobility), QoS 프레임웍(QoS Framework)

1. 서 론

현재의 컴퓨팅 환경은 메인프레임 시대와 PC 시대에서 유비쿼터스 컴퓨팅 시대로 변화되고 있다. 메인프레임 시대에서는 여러 사용자가 하나의 컴퓨터를 공유하였고 PC 시대에서는 한 사용자가 한대의 컴퓨터를 사용하였다. 그러나 유비쿼터스 컴퓨팅 시대에서는 한 사용자가 여러 대의 컴퓨터를 사용할 수 있다. 응용의 관점에서 볼 때 종래의 응용들은 하나의 컴퓨터상에서 한정된 자원만을 이용하였다. 그러나 유비쿼터스 응용은 사용자 주변의 여러 컴퓨터에 있는 다양한 자원을 이용하여 사용자에게 맞춤식 서비스를 제공해 준다. 즉, 유비쿼터스 환경에서 유기적으로 연결된 많은 센서와 소형 컴퓨터들은 사용자 주변의 컨텍스트 정보를 감지하여 응용에게 전달하고, 유비쿼터스 응용들은 다양한 컨텍스트 정보를 토대로 이동 사용자에게 최적의 서비스를 제공한다. 이를 위해 유비쿼터스 응용들은 다음과 같은 기능들을 가져야 한다.

- 컨텍스트 인지(Context Awareness)[1-4] : 컨텍스트는 사용자나 사물 주변의 상황정보이다. 실제로 많은 센서나 소형 컴퓨터들은 컨텍스트 정보를 센싱하여 용응들에게 전달하며 응용들은 이러한 컨텍스트 정보를 효과적으로 표현, 저장, 통신할 수 있어야 한다.
- 사용자 이동성(User Mobility)[5-8] : 유비쿼터스 환경에서 사용자는 자유롭게 이동하면서 작업하길 원한다. 따라서 응용은 사용자의 이동에 따라 끊김없이(seamless) 서비스를 제공해야 하고 필요에 따라 응용들이 사용자를

※ 본 연구는 2004년 부산대학교 연구비 지원으로 수행되었음.
† 준 회 원 : 부산대학교 대학원 전자계산학과
†† 종신회원 : 부산대학교 전자계산학과 교수
논문접수 : 2004년 7월 16일, 심사완료 : 2004년 8월 17일

따라 이동할 수 있어야 한다.

- QoS 적응성(QoS Adaptability)[9-11] : 사용자가 이동할 때 주변의 컨텍스트와 자원양은 수시로 변한다. 이러한 변화에 맞추어 응용이 처리하는 데이터도 동적으로 변형되어야 한다. 특히, 멀티미디어 응용의 경우 변화되는 자원의 종류와 량에 따라 QoS 조절이 필수이다.

이러한 기능들은 응용 수준에서 구현되는 것보다 미들웨어에서 구현되는 것이 효율적이다. 미들웨어는 센서와 장치들의 연합체를 쉽게 구성할 수 있고 하드웨어와 운영체제에 독립적인 플랫폼을 응용에게 제공할 수 있다. 또한, 유비쿼터스 응용들의 실행과 관리를 용이하게 한다. 본 연구는 유비쿼터스 멀티미디어 응용을 위한 미들웨어 프레임웍을 설계하고 지니 기술을 이용하여 미들웨어 프로토타입을 구현한다. 지니[15]는 SunMicrosystems사에서 릴리즈된 분산환경 자원공유 플랫폼으로 코어(core)의 크기가 매우 작아 휴대용 단말에 탑재가 용이하고 소스코드가 공개되어 있어 수정 개발이 용이하다. 또한, 자바 기반으로 구현되어 있어 자바의 다양한 유틸리티는 물론 장치독립성과 보안모델등을 그대로 이용할 수 있다. 구현된 미들웨어 시스템은 사용자 주변의 컨텍스트 정보를 효과적으로 표현 및 처리할 수 있고, 사용자의 이동에 따라 세션 핸드오프를 지원하며, 적응적 QoS 조절 메카니즘을 응용에게 제공할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 관련연구를 살펴보고, 3장에서 미들웨어 프레임웍의 구조와 핵심 서비스들(컨텍스트 서비스, QoS 관리 서비스, 사용자 이동 서비스)을 기술한다. 그리고 4장에서 구현된 미들웨어 프로토타입 시스템의 클래스 구성과 실행 인터페이스들을 설명한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

지금까지 모바일 환경이나 유비쿼터스 환경에서 컨텍스트 인지, QoS 인지 그리고 이동성 인지 미들웨어에 대한 연구는 활발히 진행되어왔다. <표 1>은 기존에 연구된 미들웨어 시스템들의 특성 비교를 보여준다. RCSM[2]은 컨텍스트 인지와 자발적(spontaneous) 그룹 통신을 요구하는 응용들의 개발과 운영을 용이하게 하기 위해서 제안된 미들웨어로서 컨텍스트와 처리 방법을 CORBA 인터페이스로 정의하고 이를 컴파일하여 ADC(Adaptive Object Container)를 생성한다. ADC는 컨텍스트를 수집하여 응용에게 전달하며 실행중에 동적으로 생성과 삭제가 가능하다. Gaia[13]은 이동하는 사용자 중심의 active space 응용들의 개발을 위한 미들웨어 구조로서 위치, 컨텍스트, 이벤트, 저장(repository)과 관련된 서비스를 제공한다. Active space는 단순히 물리적인 공간이 아니라 내부의 컴퓨터들이 유기적으로 조직되어 지능적으로 사용자를 서비스하는 환경을 말한다. 그러나 RCSM과 Gaia는 CORBA기반의 분산 객체 응용들에게 다양한 컨텍스트를 제공하기 위한 통일된 인터페이스를 제공하지만 멀티미디어 데이터의 QoS 관리나 이동성 지원에 대한 부분은 고려하지 않았다.

〈표 1〉 미들웨어 시스템들의 특성 비교

미들웨어	기반기술	목표 응용	컨텍스트 처리	QoS관리	이동성 지원
RCSM[2]	CORBA	Ac-hoc 통신 응용	다양한 컨텍스트 처리	없 음	없 음
Gaia[13]	CORBA 기반 Component 기술	범용 응용 : 응용프레임웍 제공	다양한 컨텍스트 처리 : Context File System	없 음	없 음
Aura[8]	Task 이동 기술	비실시간 응용	시스템 자원	비실시간 QoS	사용자 이동성 : Task Manager (Prism) 이용
Klara[9]	Agent 기술	QoS-Sensitive 응용	사용자 위치 정보	2단계 프레임웍 (Development,RunTime)	사용자 이동성 : Agent이용
Mobiware[10]	CORBA	멀티미디어 응용	없 음	QoS Negotiation과 Adaptation	호스트 이동성 : QoS 제어 핸드 오프
Palol[12]	SOMA[14]	범 용	없 음	QoS Adaptation	호스트 이동성 : Agent 이용

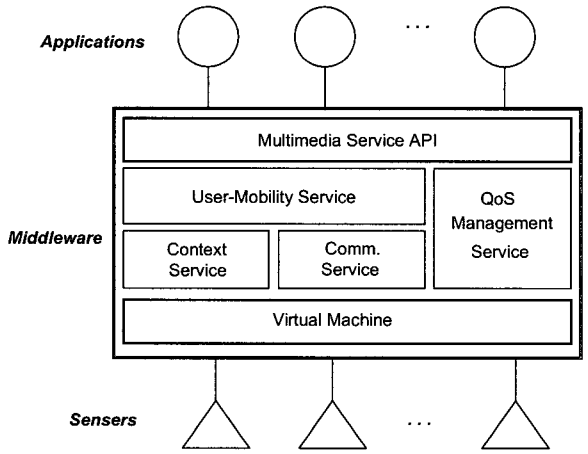
Aura[8]는 사용자의 작업공간을 이동할 때 수행중인 작업(task)을 이주시키기 위한 구조적 프레임웍을 제시하였으나 단순한 위치정보만을 이용하였고 비실시간적인 QoS만을 고려하고 있다. Klara[9]은 유비쿼터스 환경에서 QoS 인지 응용을 위한 미들웨어 프레임웍을 제시하고 에이전트 기술을 이용하여 사용자 이동성을 지원하지만 다양한 컨텍스트를 고려하지 않고 있다. 또한, Mobieware[10]와 Palol[12]은 사용자 이동성보다는 호스트 이동성 지원을 위한 프레임웍을 제시하였다. 전자는 핸드오프를 통해서 후자는 QoS 라우팅과 패스 재설정을 통해서 이동 호스트의 QoS를 조절하지만 컨텍스트 인식 기능은 없다.

그러나 본 연구는 다양한 컨텍스트 처리 기능과 멀티미디어 응용을 위한 QoS 관리 프레임웍 그리고 사용자 이동성 지원 기능을 모두 고려하는 미들웨어 프레임웍을 제안하고

자 한다.

3. CMQ 미들웨어 프레임웍

(그림 1)은 본 논문에서 제안하고 있는 미들웨어의 구조를 보여준다. 이 미들웨어는 편재형 센서들과 응용 사이에 위치하여 실시간으로 감지된 컨텍스트 정보를 수집 및 가공하여 응용에게 끊김없는 멀티미디어 서비스를 제공한다. 이 미들웨어는 다음과 같은 여러 가지 서비스 모듈로 구성된다.

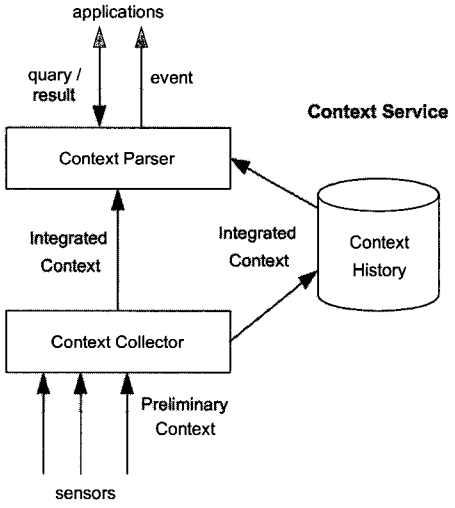


(그림 1) 제안된 미들웨어 구조

Context Service 모듈은 여러 형태의 센서로부터 컨텍스트를 감지하고 감지된 컨텍스트를 해석해서 응용이 요구한 타이밍에 이벤트를 발생시킴으로 컨텍스트의 변화를 응용에게 알린다. 또한 이 모듈을 통해서 응용은 특정 컨텍스트 정보를 질의하고 검색 결과를 제공 받을 수 있다. QoS Management Service 모듈은 멀티미디어 응용이 실행되기 전에 QoS 명세(specification)를 표현하고 QoS 협상(negotiation)을 하며, 미디어 전송 중에 QoS 모니터링(monitoring)과 QoS 적응(adaptation)을 수행한다. 또한 사용자가 한 컴퓨터에서 다른 컴퓨터로 이동할 때 QoS 재협상(renegotiation)을 수행한다. User-Mobility Service 모듈은 사용자 이동성을 지원하기 위한 핸드오프 프로토콜을 구현한다. 이 프로토콜은 사용자가 하나의 공간에서 다른 공간으로 이동할 때 서비스가 끊김이 없도록 지원한다. Communication Service 모듈은 편재형 컴퓨터간에 제어 데이터와 미디어 데이터의 통신을 위한 서비스이다. 제어 데이터의 통신을 위해 지니를 사용하고, 미디어통신과 재생을 위해서 JMF를 이용한다. 마지막으로 Multimedia Service API는 응용이 미들웨어가 제공하는 기능을 이용할 수 있도록 제공되는 프로그래밍 인터페이스이다. 이를 통해 유비쿼터스 응용 프로그램의 구현이 단순화될 수 있다.

3.1 컨텍스트 서비스

(그림 2)는 컨텍스트 서비스의 세부 구조를 나타낸다. 그림에서 볼 수 있듯이 이 구조는 Context Collector, Context Parser, 그리고 Context History로 이루어진다.



(그림 2) 컨텍스트 서비스의 구조

Context Collector는 다양한 센서로부터 컨텍스트를 수집하여 통합 컨텍스트(Integrated Context)를 생성한다. 센서로부터 인식된 초벌 컨텍스트(Preliminary Context)는 센서에 의존적인 형태이므로 바로 응용 수준에서 해석될 수 없다. 따라서 통합 컨텍스트로의 변환이 필요하다. 또한 통합 컨텍스트를 저장 하는 Context History와 컨텍스트의 변화를 이벤트 형식으로 응용에게 알리고 응용으로부터 질의에 대해 Context History에 저장된 정보를 결과값으로 반환하는 역할의 Context Parser 가 필요하다.

통합 컨텍스트 정보를 표현하고 처리하기 위해서는 컨텍스트 모델이 필요하다. Jang은 모든 응용에 적용 가능한 5W1H 컨텍스트 모델을 제안하였다[4]. 그러나 이 모델은 모든 컨텍스트를 통합적으로 표현할 수 있지만 특정 응용마다 요구하는 컨텍스트가 조금씩 다르기 때문에 응용수준에서 사용하기에는 조금 복잡하다. 본 연구는 (그림 3)에서 볼 수 있듯이 응용들이 쉽게 이용할 수 있도록 데이터베이스의 테이블 형태의 처리가 가능한 통합 컨텍스트 모델을 제안한다. 이 모델에서 하나의 컨텍스트 정보는 4개의 튜플로 구성된다.

Context (<Time>, <Object>, <Type>, <Value>)

(그림 3) 통합 컨텍스트 모델

<Time>은 컨텍스트가 발생한 시각을 의미하고 <Object>는 컨텍스트가 가르키는 대상을 의미한다. <Type> <Value>

는 컨텍스트의 종류와 값을 나타낸다. 예를 들어, "사용자 hspark가 현재 room417호에서 위치해 있다"라는 컨텍스트는 다음과 같이 표현된다.

```
Context ( <CURRENT>, <hspark>, <Location>, <room417> )
```

Context History는 통합 컨텍스트들을 저장 관리하는 데이터 베이스이다. Context Collector는 컨텍스트를 데이터베이스에 저장하고 Context Parser는 저장된 컨텍스트 정보에 대하여 두 가지 기능을 수행한다. 첫째 컨텍스트의 발생을 이벤트 형식으로 응용에게 알린다. 예를 들어서 다음의 컨텍스트 정보는 사용자 hspark가 room417호에서 room418로 이동했음을 나타낸다. 이 경우 Context Parser는 사용자의 이동이 발생했음을 이벤트를 기다리고 있는 응용에게 알린다.

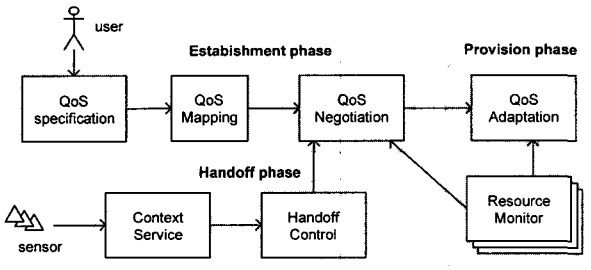
```
Context1 ( <03 : 45 : 05>, <hspark>, <Location>, <room417> )
Context2 ( <03 : 47 : 12>, <hspark>, <Location>, <room418> )
```

둘째, Context Parser는 사용자가 특정 컨텍스트 정보를 질의할 때 Context History의 내용을 검색하여 결과를 사용자에게 반환한다. 사용자는 <Time> 또는 <Object>, <Type> 등을 기준으로 특정 컨텍스트 정보를 질의할 수 있다. 다음의 질의는 12시 이후의 사용자 hspark의 모든 컨텍스트 정보를 검색하는 질의 예이다.

```
Context ( <After 12 : 00 : 00>, <hspark>, ?, ? )
```

3.2 QoS 관리 서비스

(그림 4)는 제안된 미들웨어에서 QoS 관리 서비스 프레임웍을 보여준다. 이 프레임웍은 Establishment, Provision 그리고 Handoff 단계로 구성된다.



(그림 4) QoS 관리 서비스 프레임웍

3.2.1 Establishment 단계

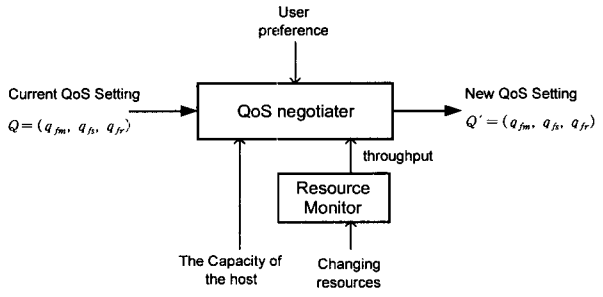
QoS 명시(Qos Specification)는 사용자 수준(user level), 응용 수준(application level) 그리고 자원 수준(resource level)에서 가능하다. 편재형 환경에서는 사용자 중심의 서비스가 제공되어야 하기 때문에 사용자 수준의 QoS 명시가 반드시 필요하다. 이 사용자 수준의 QoS 명시는 응용 수준의 QoS 파라메터와 나아가 자원 수준의 QoS 파라메터로 매핑될 것이다. Establishment 단계의 핵심은 QoS 협상(QoS Negotiation)이다. QoS 협상은 앞에서 예를 들고 있는 시나리오와 같이 유비쿼터스 환경에서 사용자의 이동에 의해서 일어날 수 있는 환경과 장치의 변화에 의해서 시스템 자원과 네트웍 자원을 고려해서 실제 실행 가능한 응용 수준의 QoS 파라메터를 설정하는 작업이다. 유비쿼터스 비디오 응용의 경우는 3가지의 협상(Intra-Client 협상, Client-Server 협상, Inter-Client 협상)이 있을 수 있다. Intra-Client 협상은 현재 클라이언트 시스템의 자원으로 사용자가 제시한 QoS를 최대한 만족시킬 수 있도록 QoS 파라메터를 조절한다. Client-Server 협상은 서버와 클라이언트 호스트간에 네트웍 자원 상태를 고려해서 QoS 파라메터를 조절하는 것이다. Inter-Client 협상은 사용자의 이동이 발생하였을 때 이전 클라이언트에서 제공되는 QoS를 새로운 클라이언트에서 최대한 만족시켜주기 위한 QoS 협상이다.

3.2.2 provision 단계

이 단계는 초기의 설립 단계에서 협상된 QoS를 기준으로 QoS를 집행하는 단계이다. Resource Monitor는 시스템 자원과 네트웍 자원의 변화 상황을 주시하고 있다가 허용하는 QoS 임계치를 벗어날 때 이를 QoS Adaptor에게 알린다. 또한, QoS Adaptor는 자원 모니터에 피드백되는 정보를 이용하여 서비스중인 세션의 QoS 파라메터를 적응적으로 조절한다. 예를 들어 네트웍 패킷 손실률이 급격히 증가할 경우 프레임율을 낮추고 네트웍 상태가 양호할 경우 프레임율을 높이는 등의 조절이 필요하다.

3.2.3 Handoff 단계

유비쿼터스 환경에서 사용자 이동은 센서를 통해 감지되며 Context 서비스에 의해 해석되고 이벤트 형식으로 사용자 이동정보가 응용에게 전달된다. Handoff Controller는 사용자가 이동할 때 사용자 주변의 컴퓨터간에 작업이동을 지원한다. 즉, 이동중인 사용자에게 서비스가 끊김 없이 제공될 수 있도록 QoS 제어 핸드오프를 수행하며 이 과정에서

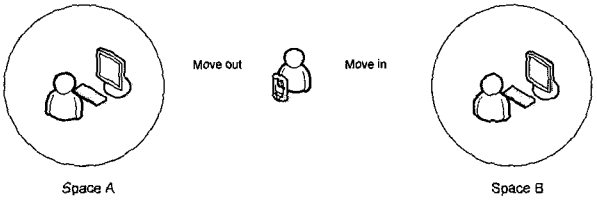


(그림 5) QoS 핸드오프 과정

QoS 재협상이 수반된다. QoS 재협상을 통해 새롭게 이동한 클라이언트 호스트를 기준으로 QoS 파라메터를 다시 설정한다. (그림 5)는 QoS를 핸드오프하는 과정을 보인다. 현재의 QoS 설정 정보는 사용자가 이동하는 새로운 호스트의 사양과 자원량, 그리고 사용자 선호도를 고려해서 새로운 QoS 설정 정보로 바뀐다. 이 때 QoS 설정 정보는 응용 수준의 QoS 파라메터로서 비디오 포맷(q_{fm}), 프레임 크기(q_{fs}), 프레임률(q_{fr}) 등이 될 수 있다.

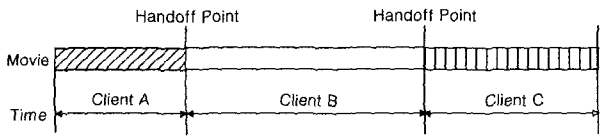
3.3 사용자 이동 서비스

이 서비스의 목적은 사용자가 컴퓨터간에 이동할 때 끊김 없는 멀티미디어 서비스를 제공하기 위한 것이다. 먼저 유비쿼터스 환경에서 사용자 이동성을 지원하는 VOD 서비스의 예를 살펴보자. (그림 6)에서 볼 수 있듯이, 사용자는 유비쿼터스 공간 A의 데스크탑 컴퓨터에서 VOD 서비스를 시작한다. 이 후 사용자는 공간 A에서 공간 B로 이동하며, 이동 중에 사용자는 PDA를 통해서 동일한 VOD서비스를 연속해서 제공 받는다. 공간 B로 이동한 사용자는 공간 B에 있는 컴퓨터를 통해서 나머지 부분을 계속해서 서비스 받는다.



(그림 6) 유비쿼터스 공간에서 사용자 이동 시나리오

사용자가 이동함에 따라 VOD 세션은 사용자를 따라 컴퓨터간을 이동한다. 핸드오프는 공간 A의 데스크탑에서 PDA로 그리고 PDA에서 공간 B의 데스크탑으로 일어난다. 이러한 핸드오프 과정은 (그림 7)과 같이 서브 세션(sub-session)이 새로 생성되고 소멸되는 연속적인 과정으로 볼 수 있다. 핸드오프가 일어나는 시점을 기준으로 하나의 서브 세션이 소멸되고 새로운 서브 세션이 생성됨을 알 수 있다. 즉, 하나의 세션은 여러 개의 서브세션으로 나누어진다.



(그림 7) VOD 응용에서 sub-session 개념

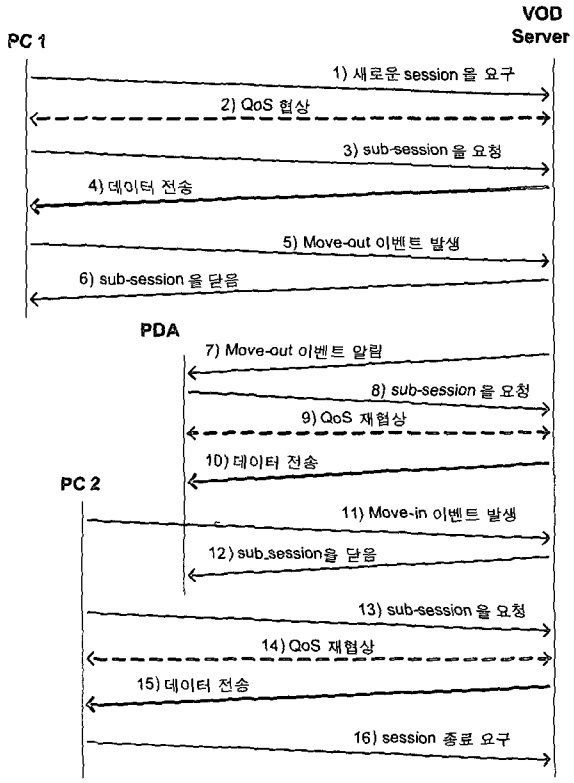
서브 세션의 정보는 송신자와 수신자의 IP 주소 그리고 비디오 파일 내의 정지 지점을 포함한다. 따라서 서브세션은 다음과 같이 표현할 수 있다.

```
SubSession( Sender, Receiver, Movie, Position )
```

예를 들어 (그림 7)은 다음과 같은 서브세션 정보로 표현될 수 있다.

```
SubSession( Sender, Client A, "aaa.mpg", 0 )
SubSession( Sender, Client B, "aaa.mpg", LAST_STOPPED_POINT )
SubSession( Sender, Client C, "aaa.mpg", LAST_STOPPED_POINT )
```

실제로 이전 서브 세션이 소멸하고 새로운 서브 세션이 생성될 때 앞에서 제시한 QoS 재협상이 수행된다. 이전 호스트에서 설정된 QoS 셋팅은 새로운 호스트의 사양과 허용된 자원의 양, 그리고 사용자 선호에 따라 새로운 QoS 셋팅으로 재협상된다. (그림 8)은 사용자 이동을 지원하는 유비쿼터스 VOD 응용을 위한 핸드오프 과정을 보여준다. 공간 A의 PC1과 공간 B의 PC2에 사용자의 위치를 인식할 수 있는 시스템이 탑재되어 있음을 가정한다. 공간 A에서 사용자는 PC1 컴퓨터를 이용하여 VOD 세션을 요청하고 VOD 서버는 세션을 생성한다. PC1과 VOD 서버 사이에 QoS 협상이 수행된 후 PC1은 서브세션을 요청하고 서버는 데이터를 전송한다. 이동하는 사용자가 공간 A에서 벗어날 때 "move out" 이벤트가 발생한다. 서버는 PC1과의 서브세션을 종료하고 PDA에게 이벤트를 발생을 알린다. PDA는 새로운 서브세션을 서버에게 요청하고 서버와 QoS 재협상을 수행한

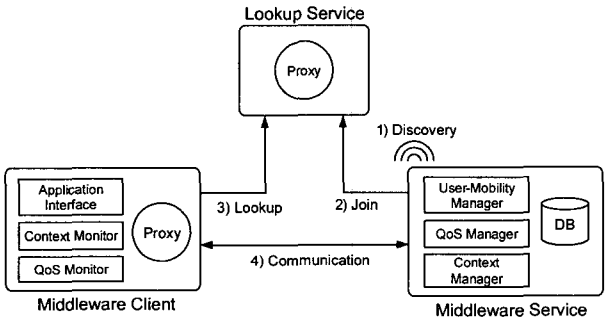


(그림 8) VOD 세션의 핸드 오프 과정

다. QoS 재협상 후 서버는 데이터를 전송한다. 사용자가 다시 공간 B로 들어갈 때 PC2는 "move in" 이벤트를 발생시키고 서버는 PDA와의 서브세션을 닫으며 PC2는 새로운 서브세션을 요청한다. QoS 재협상을 수행한 후 서버는 데이터를 전송한다. 비디오 재생이 끝나면 PC2는 세션의 종료를 요청하고 서버는 세션을 종료시킨다.

4. 미들웨어 구현

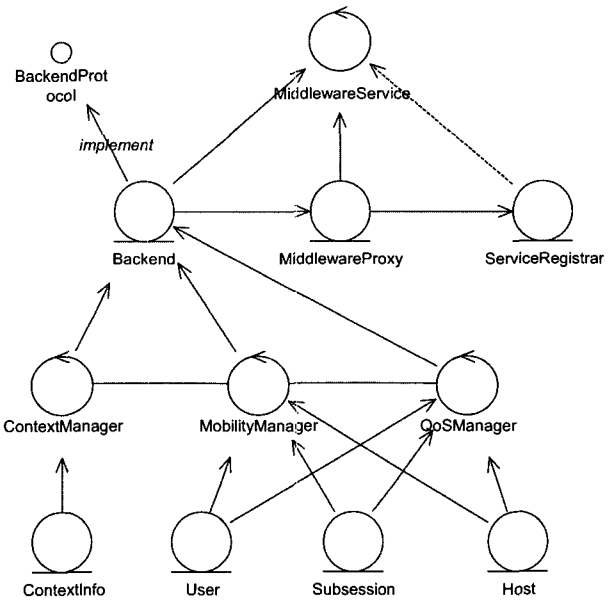
우리는 자바 환경에서 지니 기술을 이용하여 제안된 미들웨어의 프로토타입을 구현하였다. 지니 응용은 크게 클라이언트, 서비스, 조회서비스(Lookup Service)로 구성된다. 서비스는 특정한 기능을 수행하는 개체이며 사람, 프로그램, 또는 다른 서비스에 의하여 접근된다. 지니 서비스들은 Discovery/Join 과정을 통해 조회서비스를 찾아 자신의 프락시(proxy) 객체를 등록한다. 지니 클라이언트는 조회서비스로부터 프락시 객체를 다운로드하여 해당 서비스와 통신한다. 프락시는 직렬화된 객체로서 지니 서비스 객체에 대한 인터페이스를 제공한다. (그림 9)는 구현된 미들웨어 시스템의 구성도를 보여준다. 미들웨어 서비스는 컨텍스트 서비스, QoS 관리 서비스, 사용자 이동 서비스를 구현하는 객체들을 포함하고 있으며, 미들웨어 클라이언트는 QoS 모니터, 컨텍스트 모니터, 응용 인터페이스 객체들을 포함한다. 미들웨어 서비스는 조회서비스에 자신의 프락시를 등록하고 미들웨어 클라이언트는 그 프락시를 다운로드 받는다. 사용자가 어디로 이동하더라도 미들웨어 클라이언트는 프락시를 통하여 컨텍스트 정보, QoS 정보 및 세션 정보 등을 얻어 올 수 있다.



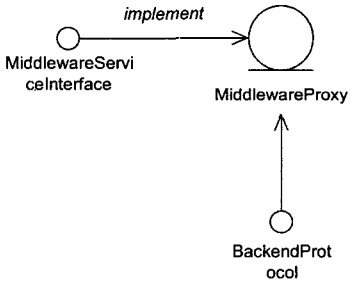
(그림 9) 지니 기반 미들웨어 시스템의 구조

4.1 클래스 구성도

(그림 10)은 실제로 구현된 미들웨어 서비스의 클래스 구성도를 보여준다. MiddlewareService 객체는 MiddlewareProxy 객체를 생성하고 ServiceRegistra 객체를 통해 지니의 조회서비스(Lookup Service)에 등록한다. 또한, Backend를 객체는 MiddlewareProxy객체가 ContextManager, MobilityManager, QoSManager와 정보를 주고 받을 수 있도록 통신 인터페이스를 제공한다. ContextManager는 Context 정보를 체계적으로 저장 관리하며, MobilityManager는 사용자의 이동에 따라 이전 서브세션을 종료하고 새로운 서브세션을 생성함으로써 세션 핸드오프를 제어한다. QoSManager는 사용자의 선호도, 이동 호스트의 사양과 자원량 등을 고려하여 서브세션의 QoS를 조절한다.

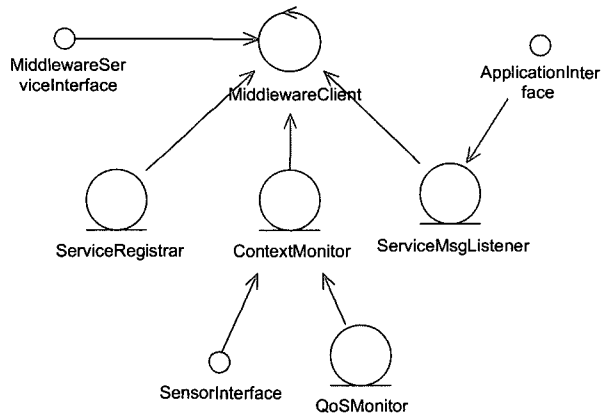


(그림 10) 미들웨어서비스의 클래스 구성도



(그림 11) 미들웨어프락시 클래스

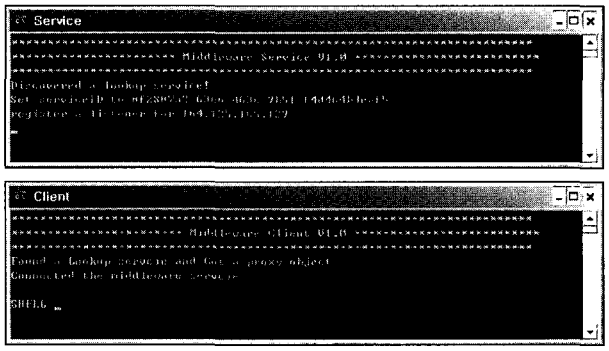
(그림 11)은 미들웨어프락시의 클래스 구조를 보인다. 이 프락시는 MiddlewareServiceInterface를 구현하고 BackendProtocol 객체를 포함한다. (그림 12)는 미들웨어 클라이언트의 클래스 구성도를 보여준다. MiddlewareClient 객체는 ServiceRegistra 객체를 통해 지니의 조회서비스로부터 프락시를 다운로드 받는다. 다운로드받은 프락시는 MiddlewareServiceInterface 객체를 통해서 접근된다. 실제로 다양한 센서로부터 감지되는 컨텍스트 정보와 호스트 자원의 QoS 정보들은 프락시 객체를 통해 미들웨어서비스로 전달된다. 미들웨어서비스에서 피드백되는 이벤트들은 ServiceMsgListener를 통해 해석되고 이 객체는 ApplicationInterface를 통해 응용에게 이벤트를 전달한다.



(그림 12) 미들웨어 클라이언트의 클래스 구성도

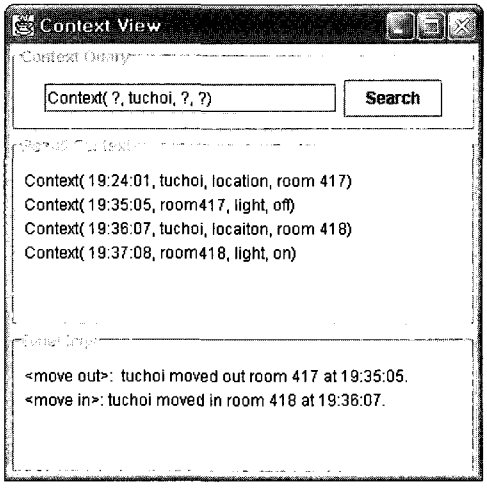
4.2 실행 결과

지니 서비스를 실행하기 위해서는 웹 서버, RMI 활성화 데몬, 조회 서비스 등이 필요하다. 지니 서비스들이 실제로 필요한 코드를 내려 받기 위해서는 HTTP 프로토콜을 이용하기 때문에 웹 서버가 필요하며, RMI 활성화 데몬은 객체에 대한 활성화와 비활성화 상태간의 전이를 관리하며 핵심적인 다른 지니 런타임 서비스에 의해 광범위하게 사용된다. 조회서비스는 프락시를 등록하고 관리하기 위해 반드시 필요하다. (그림 13)은 구현된 미들웨어 서비스와 클라이언트의 실행 모습을 보여준다. 미들웨어 서비스는 조회서비스를 찾아 프락시를 등록하고 자신의 서비스 ID를 설정한다. 미들웨어 클라이언트는 조회서비스를 찾아 프락시를 다운로드한 후 이를 통해 미들웨어 서비스에 등록한다. 미들웨어 클라이언트는 쉘 기반 인터페이스를 가지며 텍스트 명령을 입력함으로써 미들웨어 서비스가 제공하는 모든 기능을 이용할 수 있다.



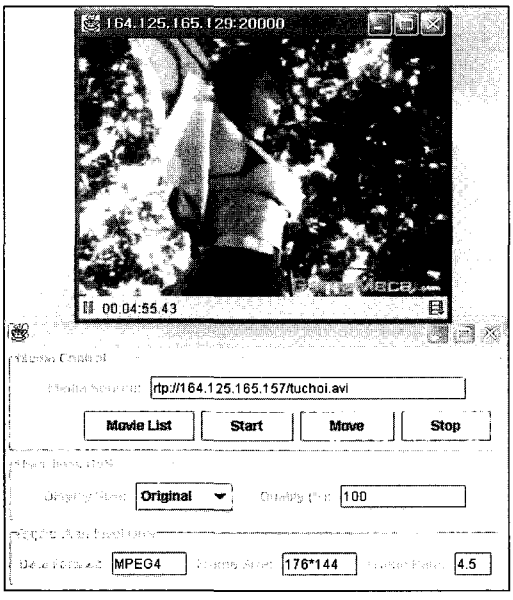
(그림 13) 미들웨어 서비스와 클라이언트의 실행화면

구현된 미들웨어 시스템은 아직 센서들이 탑재되지 않아 초벌 컨텍스트의 입력은 수작업을 통해 이루어진다. 그러나 미들웨어 클라이언트에서 통합 컨텍스트 정보의 검색과 질의는 언제 어디서나 가능하다. (그림 14)는 사용자 "tuchoi"에 대한 컨텍스트 정보와 이벤트 정보를 검색하는 예를 보여준다.



(그림 14) 컨텍스트 검색 화면

구현된 미들웨어 시스템은 유비쿼터스 응용의 개발을 용이하게 할 수 있다. (그림 15)는 구현된 미들웨어 상에서 JMF를 이용하여 구현한 간단한 VOD 응용을 보여준다. 사용자는 "Movie List" 버튼을 눌러 비디오를 선택하고 "Start" 버튼과 "Stop"을 눌러 비디오 시청을 시작하고 종료한다. 사용자가 이동할 때는 "Move" 버튼을 누른 후 이동하고 새로운 호스트에서 다시 "Move" 버튼을 누르면 이전 호스트에서 시청한 비디오를 연속해서 볼 수 있다. 사용자는 비디오 "Start" 버튼을 누르기 전에 사용자 수준 QoS 지정할 수 있으며 지정된 사용자 수준 QoS는 실행 중에 응용 수준 QoS로 매핑된다.



(그림 15) 유비쿼터스 VOD 응용의 실행 화면

5. 결 론

본 연구는 컨텍스트 인지, 사용자 이동성, 그리고 QoS 관

리를 지원하는 CMQ 미들웨어 프레임웍을 제안하였고 실제 미들웨어 시스템 원형의 구현을 통해 컨텍스트 인지, QoS 관리, 사용자 이동성 서비스들이 통합적으로 구현될 수 있음을 보였다. 제안된 CMQ 미들웨어는 자바와 지니 기술을 기반으로 설계되었으며, QoS 인지 멀티미디어 응용의 개발을 지원한다. 또한, 다양한 컨텍스트를 해석 처리할 수 있고 3단계 QoS 프레임웍을 통해 멀티미디어 응용의 QoS를 최적으로 관리한다. 그리고 사용자 이동성을 지원하기 위한 핸드오프 프토토콜을 지원한다. 그러나 본 연구는 명시적인 시험망 환경, 센서와 컨텍스트, QoS 파라메터 등을 구체적으로 가정하고 있지 않다. 따라서 향후 다양한 센서들을 장착한 시험망 환경을 구축한 후, 컨텍스트 처리, QoS 관리, 사용자 이동서비스 기능들에 대한 구체적인 구현과 성능 평가가 논의되어야 한다. 또한, 이 미들웨어 구조 하에서 멀티미디어 응용 모델에 대한 연구가 필요하다.

참 고 문 헌

[1] A. K. Dey and G. Abowd, "The Context Toolkit : Aiding the Development of Context-aware Applications," Proc. Conf. Human Factors in Computing Systems (CHI), New York, 1999.

[2] S. S. Yau, et al., "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," IEEE Pervasive Computing, 2002.

[3] A. Ranganathan, et al., "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," In ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June, 2003.

[4] S. Jang and W. Woo, "Ubi-UCAM : A Unified Context-Aware Application Model," in Proc. of Fourth International and Interdisciplinary Conference on Modeling and Using Context, Stanford, California, June, 2003.

[5] A. D. Stefano, C. Santoro, "NetChaser : Agent Support for Personal Mobility," IEEE Internet Computing, Vol.4, No.2, pp.74-79, 2000.

[6] G. Appenzeller et al., "The Mobile People Architecture," ACM Mobile Computing and Communication Review, Vol.1, No.2, 1999.

[7] Yi Cui, K. Nahrstedt, D. Xu, "Seamless User-level Handoff in Ubiquitous Multimedia Service Delivery," Multimedia Tools and Applications Journal, 2003.

[8] J. P. Sousa and D. Garlan, "Aura : an Architectural Framework for User Mobility in Ubiquitous Computing Environments," IEEE/IFIP Conference on Software Architecture, Montreal, 2002.

[9] K. Nahrstedt, et al., "QoS-aware Middleware for Ubiquitous and Heterogeneous Environments," IEEE Communications Magazine, 2001.

[10] A. T. Campbell, "QoS-aware Middleware for Mobile Multimedia Communications," Multimeida Tools and Applications, Vol.7, No.1/2, pp.67-82, July, 1998.

[11] T. Yamazaki, M. Kosuga, N. OgiNo and J. Matsuda, "MARM : An Agent-based Adaptive QoS Management Framework," IEICE trans. Comm., Vol.E84-B, No.1, Jan., 2001.

[12] P. Bellavista, Antonio Corradi, "Active Middleware for Internet Video on Demand : the QoS-aware routing solution in ubiQoS," Microprocessors and Microsystems, 2003.

[13] M. Roman et al, "Gaia : A Middleware Infrastructure to Enable Active Space," IEEE Pervasive Computing, Dec., 2002.

[14] P. Bellavista, A. Corradi and C. Stefanelli, "Mobile Agent Middleware for Mobile Computing," IEEE Computer, Vol.34, No.3, March, 2001.

[15] Sun Microsystems, http://developer.java.sun.com/developer/products/jini.

최 태 욱

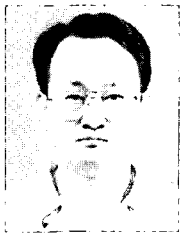
e-mail : tuchoi@pusan.ac.kr

1997년 동의대학교 전산통계학과(학사)

1999년 부산대학교 대학원 전자계산학과 (이학석사)

2000년~현재 부산대학교 대학원 전자계산학과 박사과정

관심분야 : 유비쿼터스 컴퓨팅, 멀티미디어 QoS



정 기 동

e-mail : kdchung@pusan.ac.kr

1973년 서울대학교(학사)

1975년 서울대학교 대학원(이학석사)

1986년 서울대학교 대학원(이학박사)

1990년~1991년 MIT대학 교환 교수

1995년~1997년 부산대학교 전자계산소 소장

1999년~2001년 부산대학교 BK21 사업 단장

1978년~현재 부산대학교 전자계산학과 교수

관심분야 : 병렬처리, 멀티미디어