

# URC 로봇 서버의 Load 테스트를 위한 시뮬레이션 시스템의 설계 및 구현

조성호<sup>†</sup>, 박병철<sup>\*\*</sup>, 최미정<sup>\*\*\*</sup>, 홍원기<sup>\*\*\*\*</sup>

## Network-based Load Testing Simulation System for URC Robot Servers

Seong-Ho Cho<sup>†</sup>, Byung-Chul Park<sup>\*\*</sup>, Mi-Jung Choi<sup>\*\*\*</sup>, James Won-Ki Hong<sup>\*\*\*\*</sup>

### 요 약

URC (Ubiquitous Robotic Companion) 로봇 서비스 시범 사업의 시작과 100만 로봇의 보급이 가시화 됨에 따라 URC 서비스를 사용자에게 보급하기 전에 실제 URC 로봇 서비스 시스템의 안정성 및 신뢰성의 검증이 필요하게 되었다. URC 로봇 서버의 안정성 및 신뢰성을 검증하는 방법으로는 서버의 성능을 테스트하는 툴을 이용하는 것이다. 그러나 현재까지 나온 서버 load test 프로그램들의 대상은 대부분이 웹 어플리케이션 서버들로서 URC 로봇 서버의 성능을 테스트하는 데는 적당치 않다. 본 논문에서는 URC로봇 서비스 시스템의 가용성 및 신뢰성 제고를 위한 통합서비스 플랫폼 기술 개발의 일환으로 OMA DM 서버의 가용성 및 신뢰성을 시험할 수 있는 OMA DM 서버 load 테스트 시뮬레이션 시스템의 요구사항을 분석하고, 그에 따라 시스템을 설계하고 개발한다. 개발 시스템은 서버-클라이언트 구조로서, 다양한 시뮬레이션 시나리오를 통해 가상의 URC 클라이언트를 생성하고 OMA DM 서버의 load 테스트를 수행한다. 개발한 시스템을 통하여 OMA DM 서버의 availability, scalability 그리고 response time을 측정하고 load 테스트 결과를 분석한다.

### Abstract.

A network-based intelligent service robot, called a ubiquitous robotic companion (URC), has been introduced recently. URC robots access services, content, and application software via the Internet and provide users with intelligent services in diverse fields such as education, entertainment, health care, etc. The quality of the URC service is dependent on the performance of the URC robot server. Thus, performance evaluation of the server needs to be conducted prior to the deployment of URC robot services. This study focuses on network-based load testing for URC robot servers. We have developed a load testing simulation system for URC robot management servers and have analyzed the performance evaluation results.

**Keywords:** URC, Server Load Testing System, OMA DM, Network-Based Intelligent Service Robot

본 연구는 두뇌한국 21과 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (ITA-2008-C1090-0801-0045)

<sup>†</sup> 포항공과대학교 컴퓨터공학과 분산처리 및 네트워크관리 연구실, [nology@postech.ac.kr](mailto:nology@postech.ac.kr)

<sup>\*\*</sup> 포항공과대학교 컴퓨터공학과 분산처리 및 네트워크관리 연구실, [fates@postech.ac.kr](mailto:fates@postech.ac.kr),

<sup>\*\*\*</sup> 포항공과대학교 컴퓨터공학과 분산처리 및 네트워크관리 연구실, [mjchoi@postech.ac.kr](mailto:mjchoi@postech.ac.kr),

<sup>\*\*\*\*</sup> 포항공과대학교 컴퓨터공학과 분산처리 및 네트워크관리 연구실, [jwkhong@postech.ac.kr](mailto:jwkhong@postech.ac.kr)

[논문접수일: 2008/5/30, 심사일: 2008/7/11, 게재확정일: 2008/7/17]

## 1. 서론

서버를 개발하여 서비스를 제공함에 있어서 실제로 서비스를 배포(deployment)하기 전에 서비스 수요자의 수를 고려한 서버의 성능 평가는 매우 중요한 단계이며 반드시 필요한 작업이다. 대중들의 많은 사용으로 인하여 인터넷은 IT 비즈니스들 중 가장 큰 비중을 차지하는 시장이 되었다. 인터넷을 이용한 많은 비즈니스들은 웹 어플리케이션을 통한 전형적인 서버-클라이언트 구조로 서비스를 제공하고 있다. 따라서 예상하지 못한 서버의 동작을 미연에 방지하고 원활한 서비스를 제공하기 위해서는 서버의 성능을 테스트하고 그 결과를 분석하는 것은 필수가 되었다. 낮은 수준의 QoE (Quality of Experience)를 경험한 고객들은 다시는 그 서비스를 이용하려 들지 않기 때문이다. 즉, 개발자들은 response time, throughput, availability 등의 성능 항목을 측정하고 이를 반영하여 서버의 성능을 향상 시켜야 한다.

URC (Ubiquitous Robotic Companion) 로봇 서비스[1, 2] 시범 사업의 시작과 100만 로봇의 보급이 가시화 됨에 따라 실제 URC 로봇 서비스 시스템의 안정성 및 신뢰성의 검증이 필요하게 되었다. URC 로봇 서비스 시스템은 URC 로봇 단말 인증, URC 로봇 단말 관리, URC 서비스 포탈, URC 콘텐츠 관리 등의 시스템들로 이루어져 있다. 이번 연구에서는 다양한 시스템 중에서 URC 로봇 단말 관리 시스템의 OMA DM (Open Mobile Alliance Device Management) 서버로 테스트 범위를 제한하였다. OMA DM 서버는 동시에 수많은 URC 로봇 단말을 지속적으로 관리해야 하는 핵심 시스템이기 때문이다.

본 연구의 목표는 URC 로봇 서비스 시스템의 가용성 및 신뢰성 제고를 위한 통합서비스 플랫폼 기술 개발의 일환으로 OMA DM 서버의 가용성 및 신뢰성을 시험할 수 있는 시뮬레이터 개발에 있다. 시뮬레이터를 활용하여 네트워크 기반의 가상의 로봇 클라이언트들을 생성하고 OMA DM 서버의 work load에 따른 성능을 테스트하고 분석할 수 있다. 분석 결과는 URC 로봇 서비스 시스템의 서비스 가용성을 판단하는 자료로 활용 가능하다. 실제 URC 로봇 서비스 시스템과 유사한 시뮬레이션 환경의 구축 및 테스트를 통하여 100만 로봇의 실제 보급에서 발생 가능한 문제점들을 사전에 인식할 수 있다. 따라서 로봇의 보급에 앞서 로봇 서버 시스템들의 성능 테스트를 사전에 수행하는 것은 보다 안정적인 시스템의 제공과 사후 관리 비용의 절감 효과도 기대할 수 있다.

OMA DM 서버의 성능을 측정하기 위해서는 많은 URC 로봇 클라이언트들이 필요하다. 하지만 수많은 실제 URC 로봇 클라이언트들로 서버의 성능을 측정하는 것은 로봇 클라이언트의 보급이 이루어지는 현 단계에서는 무리이다. URC 로봇 클라이언트와 같은 역할을 하는 가상의 URC 로봇 클라이언트를 만들어 서버의 성능 평가를 수행하는 것이 하나의 가능한 솔루션이다. 각종 시뮬레이션 파라미터들을 달리하면서 테스트를 수행하여 종단간 response time 및 성공률을 측정하여 OMA DM 서버를 평가한다. 네트워크 기반 환경에서 종단 사용자가 제공받는 서비스의 품질은 response time에 가장 큰 영향을 받는다.

본 논문은 서버의 work load에 대한 성능을 평가하기 위하여 가상의 URC 로봇 클라이언트들을 만들어 서버의 response time 및 성공률을 측정하는 OMA DM 서버 load 테스트 시뮬레이션 시스템을 제안한다. OMA DM 서버 load 테스트 시뮬레이션 시스템은 서버-클라이언트 구조로 이루어진 시스템으로 한 대의 Simulation-Server, 여러 대의 Simulation-Client들 그리고 결과 저장을 위한 데이터베이스로 구성된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 URC 로봇 서비스 시스템에 대해서 살펴보고, 로봇 서비스 제공을 위한 여러 서버 중 본 연구에서 테스트를 수행한 OMA DM 서버와 기존의 서버 load 테스트에 대해서 알아본다. 3장에서는 서버의 work load에 대한 성능을 테스트하기 위한 시뮬레이션 시스템의 요구사항에 대해서 정의한다. 4장, 5장에서는 시스템을 설계하고 구현한다. 6장에서는 실험 및 실험 결과를 설명하고, 마지막으로 7장에서는 결론을 맺는다.

## 2. 관련 연구

이 장에서는 URC 로봇 시스템에 대해서 알아보고, OMA DM의 표준, 프로토콜 및 구조에 대해서 살펴본다. 또한 기존의 웹 서버의 load 테스트와 관련된 연구에 대해서 살펴본다.

## 2.1 URC 로봇 서비스 시스템

국가 프로젝트의 일환으로 KT는 2005년부터 URC 로봇 서비스 시범 사업을 주도해왔다 [13]. URC 로봇 서비스 시스템은 다양한 URC 로봇 단말 (URC Robot Clients), 통합 서비스 플랫폼 (Unified Service Platform), 네트워크 환경 그리고 포탈 웹 서비스 (Web Applications)로 구성이 된다(그림 1).

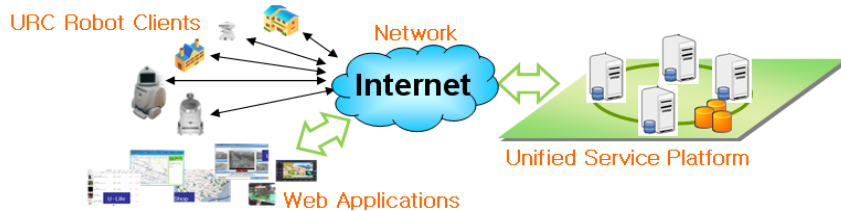


그림 1. URC 로봇 서비스 시스템의 구성

URC 로봇 단말은 통합 서비스 플랫폼의 URC 서버들과 통신을 할 수 있으며, URC 로봇 서버들은 인터넷을 통해 URC 로봇 단말들을 관리하며 다양한 정보들과 서비스들을 URC 로봇 단말들에게 제공한다. 포탈 웹 서비스는 URC 로봇 단말의 원격 제어와 다양한 관리 기능을 웹 어플리케이션을 통해서 제공한다.

그림 2 는 URC 로봇과 통합 서비스 플랫폼의 구조와 그들 사이의 표준 프로토콜을 보여준다. RUPI (Robot Unified Platform Initiative) [14] 는 다양한 로봇 단말이 서버와 연동하여 다양한 서비스를 수행 할 수 있도록 지원하는 URC 로봇의 표준 환경을 제공하는 제반 규격 및 플랫폼이다. OSGi (Open Service Gateway initiative) [15, 16] 프레임워크는 게이트웨이 상의 이기종 기술 또는 타 벤더의 서비스 간에도 통신이 가능하게 하는 표준 기술을 제공한다. OMA DM 프로토콜은 URC 로봇 단말들을 관리 하는데 사용이 되며, WSP (Wireless Session Protocol) [17] 는 데이터 교환 프로토콜로 사용이 된다.

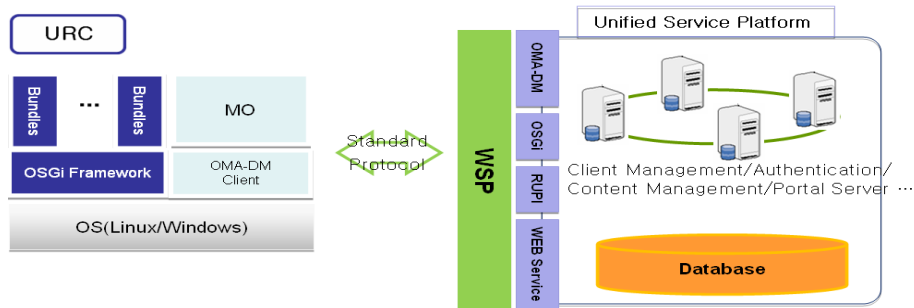


그림 2. URC 로봇과 통합 서비스 플랫폼의 아키텍처 및 표준 프로토콜

## 2.2 OMA DM

OMA DM 프레임워크[3]는 이동 단말관리를 위한 국제 표준이다. OMA DM 표준은 MO (Managed Object)를 정의하고 있으며 이러한 MO 정보를 통해 이동 단말을 관리한다[4]. OMA[5]가 제안한 DM 프로토콜은 SyncML (Synchronization Markup Language)[7] 기반의 프로토콜로서 무선 네트워크 환경에서의 대규모 정보를 수집하는데 적합하다.

OMA DM 프로토콜[6]은 그림 3과 같이 크게 두 부분으로 이루어져 있다. 그림 3의 (a) Setup phase에서는 인증과 기본 단말 정보가 교환되며, (b) management phase에서는 서버가 필요로 하는 정보들을 반복적으로 수집한다.

OMA DM의 관리 세션은 표 1과 같이 여러 개의 OMA DM 명령어들로 이루어진다. ‘Get’ 명령어를 통해 단말로부터 필요한 MO의 내용이나 MO 리스트를 수집한다. ‘Add’ 명령어를 통해 새로운 MO를 추가하고 ‘Replace’와 ‘Delete’ 명령어를 통해 MO의 내용을 변경하거나 삭제한다. 클라이언트는 ‘Alert’ 명령어를 통해 관리 세션의 시작을 알리며 서버는 ‘Exec’ 명령어를 통해 클라이언트에 새로운 프로세스를 실행시킨다.

표 1. OMA DM 명령어

Feature	Description	OMA DM Command
Reading a MO content or MO list	The server retrieves the content from the DM Client or the list of MOs residing in a management tree.	<i>Get</i>
Adding a MO or MO content	A new dynamic MO is inserted.	<i>Add</i>
Updating MO content	Existing content of an MO is replaced with new content.	<i>Replace</i>
Removing MO(s)	One or more MOs are removed from a management tree.	<i>Delete</i>
Management session start	Convey notification of device management session.	<i>Alert</i>
Executing a process	New process is invoked and returns a status code or result.	<i>Exec</i>

기본적으로 DM 프로토콜인 SyncML은 HTTP 프로토콜을 기반으로 XML 형태의 메시지를 전송하여 웹 서비스 형태로 오퍼레이션을 수행한다. 따라서 OMA DM 서버는 그림 4에서와 같이 TCP/IP 프로토콜 스택 위에 HTTP 서버 데몬이 존재하고, 그 상위에 실제 DM 어플리케이션이 있다. HTTP 서버 데몬은 TCP/IP 계층을 이용하여 실제 로봇 클라이언트의 접속을 책임지며, HTTP 프로토콜의 형태로 전달된 로봇 클라이언트의 request 메시지를 DM 어플리케이션으로 전달한다. DM 어플리케이션은 HTTP 서버 데몬 위에 존재하는 자바 어플리케이션의 한 형태로 로봇 클라이언트로부터 수신한 SyncML 메시지를 파싱하여 데이터베이스에 저장된 클라이언트의 MO 정보를 갱신하는 동작을 한다. 또한 새로운 MO 정보가 필요할 경우에는 클라이언트로부터 정보를 수집하는 역할도 수행한다. 마지막으로 backend 서버는 로봇 클라이언트의 진단이나 상태 모니터링에 있어서 추가적으로 수집이 필요한 MO 정보나 프로세스가 존재할 경우, 추가적인 오퍼레이션이 진행될 수 있도록 DM 서버에 명령을 내리는 역할을 한다.

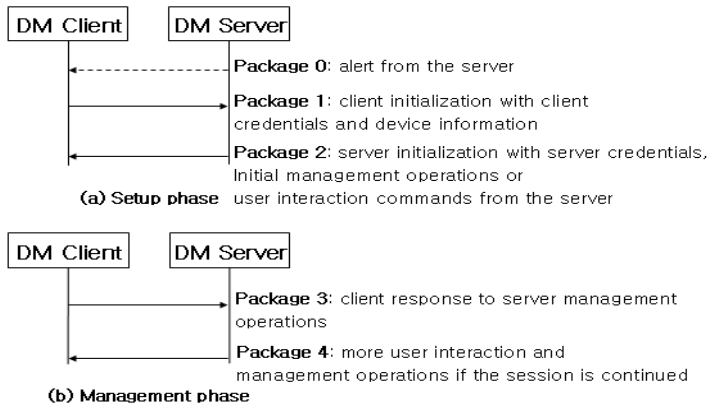


그림 3. OMA DM Protocol Package

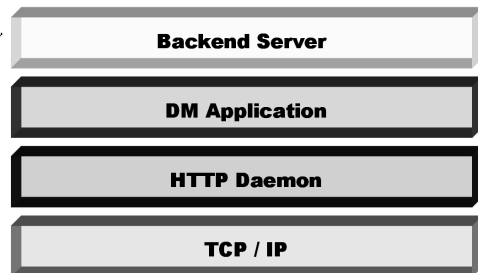


그림 4. OMA DM 서버 구조

### 2.3 웹 서버 load 테스트

기존 대부분의 서버 load 테스트는 웹 어플리케이션 서버를 대상으로 하고 있다[8, 9, 10, 11]. 사용자가 특정 웹사이트를 접속할 경우 script recorder가 사용자의 request들을 기록하고 이를 재연하기 위한 script를 생성한다. 그리고 load generator는 기록된 script를 바탕으로 파라미터를 조정하여 반복적으로 테스트를 수행한다. Load generator는 웹사이트에 반복적으로 request를 보내고 이에 대한 서버의 응답을 통해 QoS를 측정하게 된다. 이 때에 수많은 가상의 동시 사용자를 생성함으로써 웹사이트의 scalability와 availability를 측정한다[9]. 그리고 웹 기반 어플리케이션의 throughput, availability, response time 등을 분석하고[10], 이를 기반으로 보다 유효한 테스트 결과를 얻기 위해서 사용자 행동 패턴 등에 수학적 모델을 적용하여 웹 어플리케이션 서버의 성능을 측정하는 방법도 제시되고 있다[11].

## 3. 요구사항

이 장에서는 서버의 work load에 대한 성능을 평가하기 위한 시뮬레이션 시스템의 요구 사항에 대해서 살펴 본다.

### 3.1 가상 클라이언트 생성 요건

기존의 웹 어플리케이션을 테스트하기 위한 load 테스트 툴들에서의 각각의 가상 클라이언트는 2.3장에서 설명했듯이 개념적으로만 존재하는 클라이언트이다[8]. 이들은 단순히 work load에 대한 파라미터 값만을 갖는다. 서버는 클라이언트로부터 발생된 request에 대한 reply를 보내는 역할만을 담당하기 때문에 실제의 work load는 하나의 가상 클라이언트가 생성하는 단위시간 당 request의 수를 계산한다. 기존의 load 테스트 툴은 가상 클라이언트 수가 증가할 때에 load generator에서 발생시키는 request의 수를 증가시키는 방식으로 동작한다(그림 5).

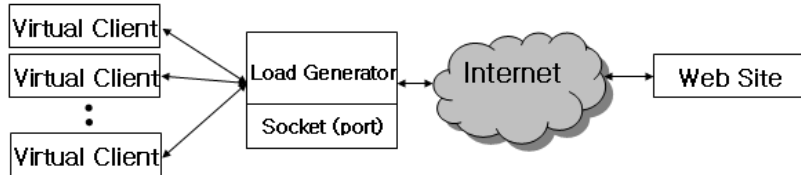


그림 5. 웹 어플리케이션의 가상 클라이언트 동작

웹 어플리케이션 load 테스트는 소켓 식별자를 필요로 하지 않기에, load generator는 request 수에 따라서 소켓을 생성만 하면 된다. 그러나 URC 로봇 서비스에 있어서 가상 클라이언트는 서버에서 각각의 클라이언트를 인식하는 Device ID와 같은 식별자를 필요로 한다. 더욱이 서버의 동시 접속 처리율을 측정하기 위해서 각각의 가상 클라이언트는 독립적으로 서버와 실제 커넥션을 맺을 수 있는 형태가 되어야 한다. 또한 각각의 가상 클라이언트는 다른 가상 클라이언트의 동작에 상관없이 독립적으로 DM 오퍼레이션을 수행할 수 있는 구조로 지원되어야 하며, DM 관리 세션을 생성하고 완료할 수 있어야 한다. 따라서 가상 클라이언트는 그림 6과 같이 별도의 load generator를 통하지 않고 서버와 통신을 할 수 있는 독립적인 소켓을 갖는 형태가 되어야 한다.

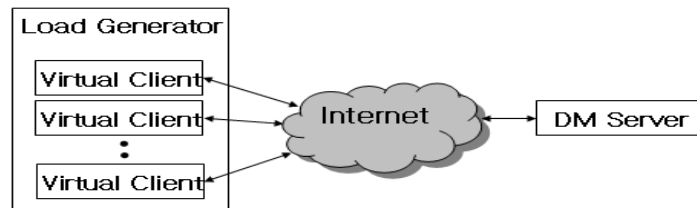


그림 6. OMA DM 서버의 가상 클라이언트 동작

### 3.2 동시 접속 요건

OMA DM 서버 load 테스트 시뮬레이션 시스템은 URC 로봇 서비스에 사용되는 OMA DM 서버의 가용성 및 안정성을 테스트 하는 것이 목적이다. 시뮬레이터에서 생성하는 가상 클라이언트는 기본적으로 서버에 동시에 접속하여 load를 생성할 수 있어야 한다. 최소의 조건으로 2,000대 이상의 가상 클라이언트가 생성되어 동작하는 것을 만족시켜야 하며, 서버의 최대 가용성을 측정하기 위하여 최대 어느 정도의 가상 클라이언트 수가 동시에 접속 가능한지 또한 테스트 되어야 한다.

DM 오퍼레이션은 HTTP 프로토콜을 사용하며 HTTP 버전 1.1의 경우 request와 reply의 한 세션이 하나의 TCP 커넥션 세션으로 이루어져 있다. 클라이언트는 TCP 커넥션으로 서버에 접속하고 접속이 성공하게 되면 HTTP 프로토콜을 사용하여 DM 오퍼레이션을 수행한다. 그리고 서버로부터 HTTP 200 OK 메시지를 받으면 TCP 세션을 종료한다. 추가적인 오퍼레이션이 존재할 때에도 동일한 TCP 세션 상에서 DM 오퍼레이션이 수행되는 것이 아니라 새로운 세션 동작이 반복된다. 위와 같이 DM 세션이 성공적으로 수행된다면 수 초 이내에 한 세션이 종료된다. 따라서 수천의 가상 클라이언트가 동시에 접속하는 상황을 재연하기 위해서는 하나의 DM 세션이 종료 되기 전에 시뮬레이터는 새로운 가상 클라이언트를 만들어 DM 세션을 생성하고 서버에 접속할 수 있어야 한다.

### 3.3 OMA DM 서버의 성능 분석 요건

시뮬레이터에서 생성되는 가상의 클라이언트들은 모두 독립적으로 서버와 통신을 하기 때문에 서버의 성능을 측정하기 위한 모듈이 각각의 가상 클라이언트에 포함되어야 한다. 시뮬레이터상에서 가상 클라이언트가 생성되면 클라이언트는 서버와 통신을 할 수 있는 소켓을 생성하고 서버에 접속을 시도한다. 이 때, 접속 성공여부를 기록하고 접속이 성공했을 경우에는 서버에 DM 명령을 보내고 서버가 reply를 보낼 때까지 대기 상태를 유지한다. Reply가 도착하면 request를 보낸 시점과 reply가 도착한 시점을 판단하여 response time을 기록한다. 따라서 서버의 성능을 분석하기 위해서는 모든 가상 클라이언트에서 기록한 수치를 현재 시뮬레이터상의 가상 클라이언트 수에 따라 보여 줄 수 있는 방법이 제공되어야 한다.

서버의 성능을 분석하기 위해서는 서버가 제공하는 네트워크 기반의 어플리케이션이 사용자의 요구를 availability, scalability와 response time의 측면에서 측정되어야 한다.

Availability는 서버-클라이언트 구조에서 서버가 클라이언트에게 정상적인 서비스를 제공할 수 있는 시간의 비율을 나타낸 것이다. 따라서 availability는 오랜 시간 동안의 측정을 필요로 한다. 하지만 OMA DM 서버의 availability 측정을 위한 시뮬레이션에서는 클라이언트와 서버의 세션 동작의 특수성 때문에 장시간의 측정이 불가능하다. 그러므로 availability 측정을 대신하기 위한 항목으로 'Connection success ratio (%)', 'Request send success ratio (%)', 'Response receive success ratio (%)'을 선정하였다. Connection success ratio는 클라이언트가 서버에 접속을 시도하였을 때 실제로 접속에 성공한 비율을 의미하며 request send / response receive success ratio는 각각 클라이언트가 서버로 보내는 request 메시지와 서버에서 클라이언트로 보내는 response 메시지가 정상적으로 전송된 비율을 의미한다.

Load 테스트 중 특정 scalability에 도달하게 될 경우 서버는 'flash crowd' 상태에서 빠진다. 이는 서버의 리소스 사용률이 한계에 이르게 되면 서버가 클라이언트의 접속이나 request 메시지의 요청을 받아들이지 못하게 되는 상태이다. 따라서 availability 측정을 위한 위의 세가지 항목에 대해 가상 클라이언트의 수를 늘려가며 서버의 scalability 테스트를 수행한다.

Response time(ms)은 클라이언트가 서버에게 request를 보내고 그에 해당하는 response를 받을 때까지 소요된 시간을 의미한다. 서버에 수많은 request가 동시에 전송될 경우에 서버의 성능이 저하될 수 있기 때문에 이 또한 가상 클라이언트의 수를 늘려가며 서버의 성능을 측정한다.

#### 4. 시스템 디자인

시스템 리소스의 한계로 인해 단일 PC상에서 생성할 수 있는 가상의 클라이언트 수는 제한될 수 밖에 없다. 가상의 클라이언트가 동시에 동작할 수 있도록 하기 위해서는 멀티 스레드를 이용해야 한다. 각각의 OS마다 지원되는 최대 멀티 스레드 수의 제한이 다르고 CPU의 종류와 물리 메모리에 따라 생성 가능한 스레드의 수는 차이가 있을 수 밖에 없다. 따라서 100만개의 가상 클라이언트를 생성하여 서버를 테스트하기 위해서는 다수의 시뮬레이터가 분산 구조로 이루어져야 하며 다수의 시뮬레이터를 관리하기 위한 시뮬레이터 서버도 필요하다. 그림 7은 분산 처리 형태의 시뮬레이터 구조를 나타낸 그림이며 점선의 사각형으로 표시된 부분이 실제 load 테스트 시뮬레이션 시스템에 해당한다.

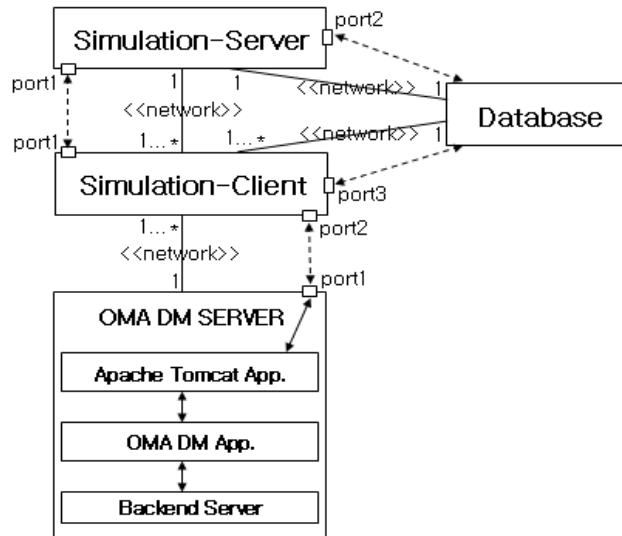


그림 7. 서버 Load 테스트 시뮬레이션 시스템 구조

#### 4.1 Simulation-Server

Simulation-Server는 Simulation-Client들을 관리하는 역할을 수행한다. 클라이언트가 설치되는 PC 성능에 따라 생성할 수 있는 Virtual-URC 클라이언트 수가 제한되므로 각 PC의 성능을 넘지 않는 범위에서 가상 클라이언트를 생성하여 OMA DM 서버의 성능을 측정하기 위해서 여러 대의 Simulation-Client가 필요하다. 또한 여러 대의 Simulation-Client를 동시에 관리하기 위한 Simulation-Server가 필요하다.

Simulation-Server는 다양한 시뮬레이션 시나리오를 만들어 낼 수 있다. 그리고 Simulation-Client들에게 일괄적으로 시나리오를 전송하고 시뮬레이션 시작 명령을 내린다. Simulation-Client들로부터 시뮬레이션 종료 보고를 받으면, Simulation-Server는 데이터베이스에 저장되어 있는 각 Simulation-Client의 시뮬레이션 결과를 통합하여 보여준다.

그림 8의 Simulation-Server의 각 모듈의 기능은 다음과 같다.

- **Sim-Client Manager:** Simulation-Client와 통신을 수행하는 모듈이다. Simulation-Client들을 관리하며, Simulation-Client가 OMA DM 서버에 접속하여 원하는 시나리오대로 작동하게끔 각종 파라미터들을 넘겨준다.
- **User Interface:** OMA DM 서버의 IP 주소, PORT 번호 등을 직접 입력할 수 있으며 이 모든 파라미터들을 Sim-Client Manager를 통하여 Simulation-Client에 전송할 수 있다. 그리고 모든 Simulation-Client의 시뮬레이션이 끝나고 난 후, 시뮬레이션의 결과를 사용자에게 보여준다.
- **Graphic Module:** 각각의 Simulation-Client들이 시뮬레이션 결과를 데이터베이스에 저장한 후, Simulation-Server는 Sim-Client Manager를 통하여 시뮬레이션 완료를 보고 받는다. 이후, 각 Simulation-Client의 시뮬레이션 결과 및 통합 시뮬레이션 결과를 그래프로 그리기 위해서 원격 데이터베이스로부터 얻은 데이터를 바탕으로 그래프에 필요한 연산을 수행한다.

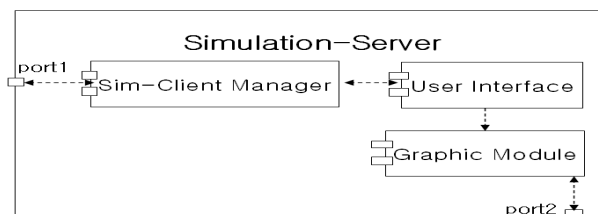


그림 8. Simulation-Server 프로그램

#### 4.2 Simulation-Client

Simulation-Client는 OMA DM 서버와 통신을 하며 실제 Simulation을 수행하는 Simulation-Client

프로그램이 설치된 PC이다. Simulation-Client의 성능을 넘지 않는 범위에서 실제 시뮬레이션이 실행되어야 한다. 그래야만 이후 발생하는 성능 저하의 원인을 OMA DM 서버에서 찾을 수 있기 때문이다. 본 논문에서는 최대 10,000개의 Virtual-URC를 40대의 Simulation-Client에서 생성하여 OMA DM 서버를 테스트한다. 따라서 Simulation-Client당 최대 250개의 Virtual-URC를 필요로 한다. 하나의 Simulation-Client가 실제 생성하는 Virtual-URC의 생성률과 전체 개수를 측정하였을 때, 시뮬레이션 시나리오에서 설정한 파라미터와 같았다. 따라서 하나의 Simulation-Client에서 250개의 Virtual-URC를 생성하는 것은 전혀 무리가 없었다.

Simulation-Client들은 Simulation-Server로부터 동시에 시뮬레이션 시나리오를 받는다. Simulation-Client들은 수신한 시나리오대로 OMA DM 서버에 접속을 하고 특정 메시지를 보내고 그 메시지에 상응하는 응답을 받고 그 결과를 데이터베이스에 저장을 한다(그림 9).

Simulation-Client의 각 모듈의 기능은 다음과 같다.

- **Sim-Server Conn:** Simulation-Server에 접속을 가능하게 한다. Simulation-Server가 송신한 데이터를 parsing하고 각 파라미터들의 값을 초기화 한다. 이후 OMA DM 서버와의 시뮬레이션을 끝내고 Simulation-Client는 시뮬레이션 완료를 Simulation-Server에 보고한다.
- **User Interface:** Simulation-Server와 통신하는 부분, 원하는 시나리오를 수행하기 위해 필요한 데이터들을 입력 받는 부분, 그리고 시뮬레이션 결과를 보여주는 부분으로 나눌 수 있다. 시뮬레이션에 필요한 데이터를 입력 받는 Interface는 OMA DM 서버의 정보들, 생성하려는 Virtual-URC 클라이언트의 정보들을 직접 입력 가능하게 한다. 또는 Sim-Server Conn을 통하여 Simulation-Server가 전송한 정보들을 보여준다.
- **Virtual-URC Manager:** User Interface를 통해 설정된 파라미터들을 바탕으로 Virtual-URC 클라이언트를 생성한다. 각 스테드마다 하나의 Virtual-URC 클라이언트가 생성된다. 생성된 Virtual-URC 클라이언트는 먼저 OMA DM 서버와 접속을 맺고 포맷에 맞는 데이터를 전송하고 나서, OMA DM 서버의 response 데이터를 수신한다. 이때 시간 및 성공 여부 정보들을 저장한다. 성공적으로 OMA DM 서버의 response 데이터를 수신한 경우, Virtual-URC 클라이언트는 종료가 된다. 일정 시간이 지나도 OMA DM 서버의 response 데이터를 수신하지 못한 경우 또는 0 byte를 수신한 경우는 실패로 처리하고 Virtual-URC 클라이언트를 종료 시킨다.
- **Result Analysis:** Virtual-URC Manager에서 저장된 데이터를 바탕으로 response time 등의 결과를 분석하는 모듈이다.
- **Result Report:** 원격 데이터베이스에 접속을 한 후, Simulation-Client의 시뮬레이션 결과를 저장한다. 그리고 나서 Simulation-Server에 Simulation-Client의 ID로 시뮬레이션 완료를 보고한다.
- **Graphic Module:** 시뮬레이션 이후, 결과를 그래프로 그리기 위해서 Result Analysis 모듈로부터 받은 데이터를 바탕으로 그래프를 그리기 위해 필요한 계산을 한다.

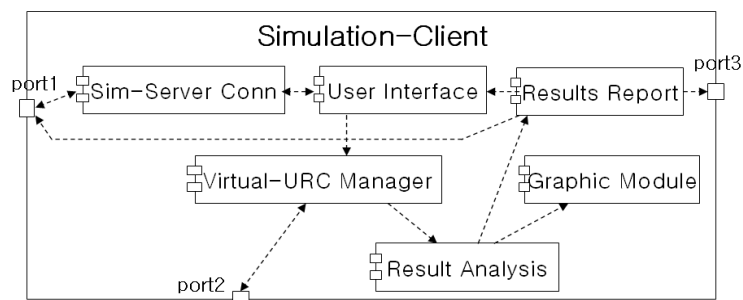


그림 9. Simulation-Client 프로그램

### 4.3 시뮬레이션 프로세스

그림 10은 한 번의 시뮬레이션 동안 Simulation-Server와 Simulator-Client, OMA DM 서버간의 프로세스 처리 구조를 나타낸 것이다.

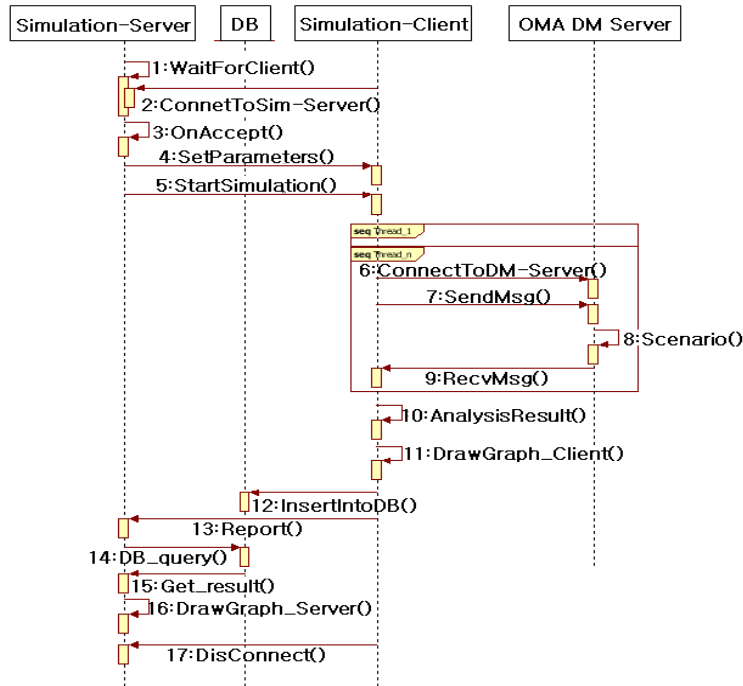


그림 10. 시뮬레이션 프로세스

각각의 프로세스는 다음과 같은 동작을 처리한다.

- **WaitForClients():** Simulation-Server에서 Simulation-Client들이 접속하기를 기다린다.
- **ConnectToSim-Server():** Simulation-Client들이 Simulation-Server에 접속을 시도한다.
- **OnAccept():** Simulation-Server는 Simulation-Client의 접속을 허용하고, Simulation-Client의 정보들을 구조체에 저장한다.
- **SetParameters():** Simulation-Server의 User Interface를 통하여 미리 저장된 파라미터들을 접속된 각 Simulation-Client에 맞게 바꾸어 Simulation-Client에 전송한다.
- **StartSimulation():** Simulation-Server에 접속된 모든 Simulation-Client에게 시나리오에 맞는 시뮬레이션을 수행하는 명령어를 보낸다.
- **Thread\_1 ~ Thread\_n**
  - ✓ **ConnectToDM-Server():** User Interface 또는 Simulation-Server를 통하여 설정된 정보들을 바탕으로 시뮬레이션 된다. 이때의 성공 여부 및 시간을 메모리에 저장한다.
  - ✓ **SendMsg():** 이미 만들어 놓은 SyncML 메시지에 Virtual-URC 클라이언트 ID 부분을 수정하여 OMA DM 서버에 보낸다. 이때의 성공 여부 및 시간을 메모리에 저장한다.
  - ✓ **Scenario():** OMA DM 서버는 Virtual-URC 클라이언트의 메시지를 처리한다.
  - ✓ **RecvMsg():** Simulation-Client에서 OMA DM 서버로 보낸 메시지에 대한 response 메시지를 수신한다. 이때의 성공 여부 및 시간을 메모리에 저장한다
- **AnalysisResult():** Thread\_1 ~ Thread\_n이 저장한 데이터들을 바탕으로 response time 등을 구한다. 그리고 시간 흐름에 따라 증가하는 Virtual-URC 클라이언트의 수에 따른 각종 시간 정보들도 구한다.
- **DrawGraph\_Client():** 각 Simulation-Client의 결과를 바탕으로 response time 및 각종 시간 정보들을 그래프로 보여준다.
- **InserIntoDB():** 메모리에 저장된 response time 및 각종 시간 정보들을 원격 데이터베이스에 저장한다.
- **Report():** 원격 데이터베이스 insert 작업까지 완료되고 나면 Simulation-Client는 Simulation-Server에 시뮬레이션 완료라는 메시지를 보낸다.
- **DB\_query():** Simulation-Client에서 데이터베이스 작업이 성공적으로 마치면, Simulation-Server는 데이터베이스에 접속하여 Simulation-Client들이 저장한 데이터들을 요구한다.
- **Get\_results:** 원격 데이터베이스는 Simulation-Server가 요구한 정보들을 전송한다
- **DrawGraph\_Server():** 원격 데이터베이스로부터 받은 각 Simulation-Client의 결과들을 통합한

정보를 바탕으로 response time 및 각종 시간 정보들을 그래프로 보여준다.

- **DisConnect():** Simulation-Client는 report 이후 언제든지 Simulation-Server에 disconnect 메시지를 보낼 수 있다.

## 5. 시스템 구현

이 장에서는 4장에서 설계한 시물레이션 시스템의 구현에 대해서 살펴본다. Windows 기반으로 개발된 시물레이션 시스템의 개발 환경은 다음과 같다.

- 운영체제: Windows XP Service Pack 2
- 프로그래밍 언어: C++ (Visual Studio .Net 2005)
- 라이브러리: libmysql.lib, ws2\_32.lib
- 데이터베이스: MySQL

### 5.1 Simulation-Server 프로그램

Simulation-Server는 Simulation-Client 프로그램을 동시에 관리하며, 각 Simulation-Client 프로그램이 데이터베이스에 저장한 데이터들을 통합하여 전체 결과를 그래프로 보여준다. 그림 11은 Simulation-Server 프로그램의 실행 결과로 Simulation-Client들의 접속 정보, 시나리오의 파라미터 그리고 원격 데이터베이스 상태를 보여준다. 처음 3개의 그래프는 전체 Virtual-URC 클라이언트의 접속 및 송·수신 성공 여부들을 보여준다. 오른쪽 하단의 나머지 그래프는 전체 Virtual-URC 클라이언트의 생성 시간 순서에 따른 Virtual-URC 클라이언트의 response time을 보여준다.

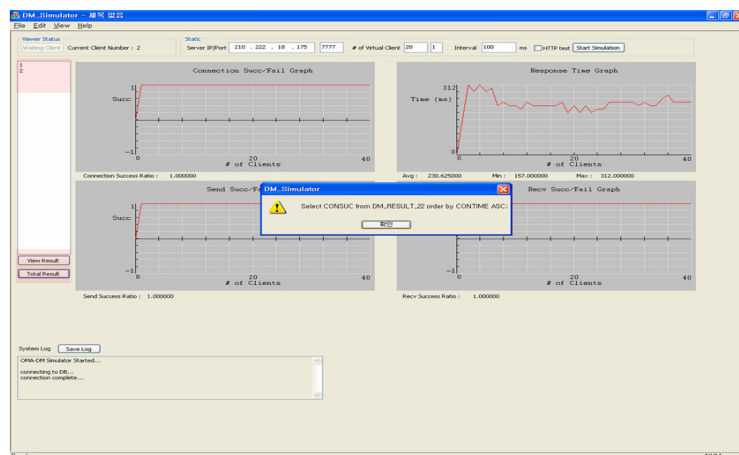


그림 11. Simulation-Server 프로그램

### 5.2 Simulation-Client 프로그램

실제 OMA DM 서버와 통신하는 프로그램으로 Simulation-Server 프로그램이 정한 시나리오대로 동작하면서 수집한 정보들을 원격 데이터베이스에 저장한다. 그림 12는 각 Simulation-Client 프로그램의 실행 결과로 OMA DM 서버 정보, 시나리오 파라미터, 그리고 시물레이션 결과 정보들을 Virtual-URC 클라이언트 별로 보여준다.

3개의 그래프는 각 Virtual-URC 클라이언트의 접속 및 송·수신 성공 여부들을 보여준다. 나머지 하나의 그래프는 해당 Simulation-Client의 Virtual-URC 클라이언트의 생성 순서에 따른 Virtual-URC 클라이언트의 response time을 보여준다.

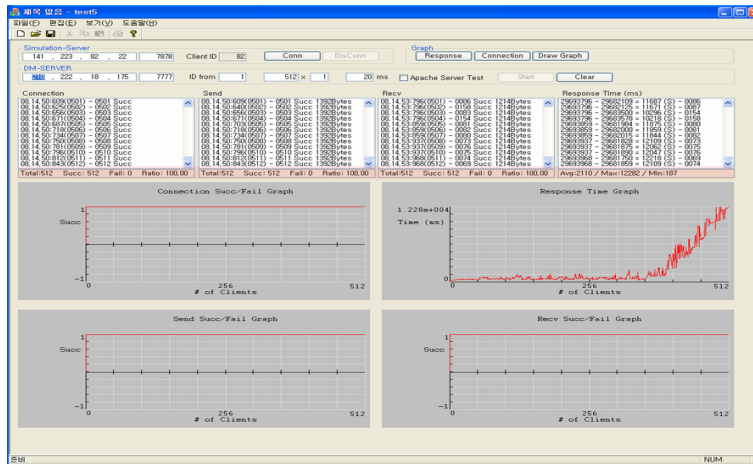


그림 12. Simulation-Client 프로그램

## 6. 실험 및 실험결과

이 장에서는 시뮬레이션 테스트를 수행한 환경과 테스트 방법 및 테스트 결과를 분석한다.

### 6.1 실험 환경

Simulation-Client 프로그램이 수행되는 PC에서 스레드 개수를 제한하는 가장 큰 요인은 메모리이다. 각 스레드는 기본으로 1MB의 스택 메모리를 예약한다. 32비트 Windows에서, 하나의 프로세스는 기본적으로 2GB의 가상 메모리 주소를 가지므로 이론적으로 최대 2,000개까지 스레드를 만들 수 있다[12]. 본 논문에서는 최대 10,000 개의 Virtual-URC 클라이언트를 필요로 하며, Simulation-Client에 걸리는 부하를 최소로 하기 위해서 40대의 PC에 Windows 기반의 Simulation-Client 프로그램을 각각 설치하였다. 한 대의 PC는 최대 250 개의 스레드를 생성하고 250MB의 가상 메모리를 필요로 한다. 이는 실제로 물리 메모리의 1/4 정도만 차지하며, Simulation-Client 프로그램이 Virtual-URC 클라이언트를 생성하는데 전혀 무리가 없었다.

각각의 Simulation-Client는 Simulation-Server에 접속한다. 그림 13은 시뮬레이션 실험망으로 사용된 POSTECH의 WAN 네트워크 구성을 나타낸 것이다. POSTECH의 캠퍼스 네트워크 내에 Simulation 시스템이 있으며 1Gbps Ethernet이 2개의 서로 다른 ISP의 네트워크에 연결된 형태이다.

Simulation-Server 프로그램과 Simulation-Client 프로그램이 설치된 PC의 사양은 다음과 같다.

- CPU: Pentium D 3.2GHz
- 메모리: 1GB
- 운영체제: Windows XP Service Pack 2
- .NET Framework 2.0

OMA DM 서버는 URC 로봇 서비스 시범 사업자인 KT에서 제공하였다.

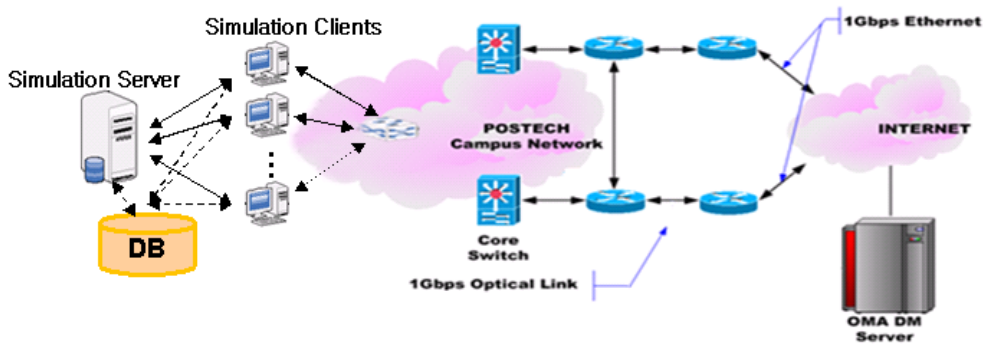


그림 13. OMA DM 서버 Load Test 를 위한 실험망

## 6.2 실험 방법

OMA DM 서버의 로봇 관리 기능 및 성능을 측정하기 위하여 표 2와 같은 다양한 시뮬레이션 시나리오를 설정하여 각각의 시나리오에 대한 시뮬레이션을 수행한다. Virtual 클라이언트는 전체 Simulation-Client에서 실험 동안 생성하는 전체 Virtual-URC 클라이언트의 수이다. Virtual-URC 클라이언트 생성물은 전체 Simulation-Client에서 초당 Virtual-URC 클라이언트를 생성하는 수이다. 시나리오 별로 각각의 파라미터를 달리하여 시뮬레이션을 실행하여 로봇의 connection 성공률, message send 성공률, message receive 성공률, response time을 측정하였다. OMA DM 서버는 Virtual-URC 클라이언트 별로 식별자를 필요로 한다. 따라서 시뮬레이션을 하기 전에 OMA DM 서버 내의 데이터베이스에 Device ID를 미리 등록을 하였다.

2.2장의 그림 4에서 OMA DM 서버의 구조에 대해서 설명을 했듯이, OMA DM 서버의 성능 장애 요소가 되는 모듈을 찾기 위해서는 계층별 성능 측정이 필요하다. 계층별 성능을 비교하기 위해서는 HTTP 서버 데몬의 성능을 측정하고 다시 DM 어플리케이션의 성능을 측정한다. 이와 같은 과정을 통해, 어느 계층에서 성능 저하가 일어나는지를 판단할 수 있다.

HTTP 서버 데몬의 성능 측정을 위한 request는 DM 오퍼레이션을 포함하지 않는 단순한 HTTP GET 명령어로 이루어진다. 이러한 request는 OMA DM 서버로 메시지가 전달되지 않으며, HTTP 서버 데몬 자체에서 reply를 보낸다. 이를 통해 HTTP 서버 데몬의 response time과 다양한 성공률을 측정할 수 있다.

DM 어플리케이션의 성능 측정은 다음과 같은 방식으로 이루어진다. Virtual-URC 클라이언트가 DM 오퍼레이션을 포함한 request를 서버로 전달하면 HTTP 서버 데몬은 메시지를 DM 어플리케이션으로 전달하고, SyncML 메시지를 파싱한 DM 어플리케이션은 자체 데이터베이스에 포함되어 있는 '/DevInfo' 항목을 수정하고, 이에 대한 응답을 클라이언트로 전송한다. 이때 DM 어플리케이션의 reply에 대한 response time을 측정한다. 추가적인 'Get', 'Replace' 명령들에 대한 동작을 수행할 경우에는 backend 서버와 OMA DM 서버 간의 추가적인 메시지 전달이 이루어지며 이 둘간의 통신은 synchronous한 방식으로 통해 이루어지므로 response time이 급격하게 증가하게 된다. 따라서 순수한 OMA DM 서버의 클라이언트 인증 및 클라이언트 정보 교환 성능 측정을 위해서는 그림 3의 (a) setup phase에 해당되는 DM 오퍼레이션만을 수행하면 된다.

표 2. 시뮬레이션 시나리오

Scenario	Virtual 클라이언트	Virtual-URC 클라이언트 생성률 (#/s)	App.
1	1,000	20, 40, 60, 80, 100	HTTP
2	2,000	20, 40, 60, 80, 100	HTTP
3	3,000	20, 40, 60, 80, 100	HTTP
4	4,000	20, 40, 60, 80, 100	HTTP
5	5,000	20, 40, 60, 80, 100	HTTP
6	6,000	20, 40, 60, 80, 100	HTTP
7	7,000	20, 40, 60, 80, 100	HTTP
8	8,000	20, 40, 60, 80, 100	HTTP
9	9,000	20, 40, 60, 80, 100	HTTP
10	10,000	20, 40, 60, 80, 100	HTTP
11	1,000	20, 40, 60, 80, 100	DM
12	2,000	20, 40, 60, 80, 100	DM
13	3,000	20, 40, 60, 80, 100	DM
14	4,000	20, 40, 60, 80, 100	DM
15	5,000	20, 40, 60, 80, 100	DM
16	6,000	20, 40, 60, 80, 100	DM
17	7,000	20, 40, 60, 80, 100	DM
18	8,000	20, 40, 60, 80, 100	DM
19	9,000	20, 40, 60, 80, 100	DM
20	10,000	20, 40, 60, 80, 100	DM

## 6.2 실험 결과

표 3은 표 2의 시나리오 1번부터 10번까지의 HTTP 서버 데몬의 시뮬레이션 결과의 평균과 표준편차를 보여준다. Virtual-URC Client 생성률에 상관없이 평균 response time이 24ms를 넘지 않을 정도로 빠른 반응을 보였으며, 성공률 역시 99% 이상으로 높음을 알 수 있다.

표 3. HTTP 서버 데몬의 시뮬레이션 결과

Virtual-URC 클라이언트 생성률	성공률 (%)			응답시간 (ms)
	접속	송신	수신	
20	99.8988 (0.0983)	99.8988 (0.0983)	99.8988 (0.0983)	23.3836 (11.5920)
40	99.9478 (0.0704)	99.9478 (0.0704)	99.9478 (0.0704)	20.6908 (11.7056)
60	99.9577 (0.0645)	99.9577 (0.0645)	99.9577 (0.0645)	21.0276 (13.4861)
80	99.9489 (0.0762)	99.9489 (0.0762)	99.9489 (0.0762)	20.4977 (12.1382)
100	99.9089 (0.1148)	99.9089 (0.1148)	99.9089 (0.1148)	21.3293 (11.2068)

HTTP 서버 데몬의 시뮬레이션 결과(표 3)와 비교하였을 때, DM 어플리케이션의 시뮬레이션은 Virtual-URC 클라이언트 생성률과 전체 Virtual-URC 클라이언트의 수에 따라서 성능의 차이가 확실히 드러남을 알 수 있다(그림 14, 그림 15).

그림 14와 그림 15는 초당 Virtual-URC 클라이언트 생성 수에 따라서 각 시나리오 별로 response time 과 접속 또는 수신 성공률을 나타내는 그래프이다. 수신 성공률은 항상 접속 성공률보다 낮거나 같을 수 밖에 없어서, 그림 15에서 수신 성공률은 접속 성공률의 그래프에 오차 형식으로 표현을 하였다.

40대의 Simulation-Client에서 Virtual-URC 클라이언트의 수를 초당 20개씩 생성하였을 경우, 전체 Virtual-URC 클라이언트 수에 관계없이 평균 response time은 약 254ms으로 빠른 결과를 보였다. 초당 Virtual-URC 클라이언트의 수를 40으로 테스트하였을 때, 시나리오 11, 12, 13의 평균 반응속도는 약 649ms를 보였지만, 시나리오 14이후의 평균은 3867ms로 4초에 가까운 response time을 보였다. 초당 Virtual-URC 클라이언트의 수를 60이상으로 테스트 한 경우, 시나리오에 관계없이 DM 어플리케이션의 느린 response time을 볼 수 있다. 그리고 시뮬레이션 동안의 전체 Virtual-URC 클라이언트 수가 1,000개씩 늘어날수록 DM 어플리케이션의 response time도 같이 비례해서 늘어남을 볼 수 있다(그림 14).

접속 성공률과 메시지 전송 성공률은 모든 시나리오에 있어서 같았다. 하지만 메시지 수신 성공률은 앞의 두 성공률보다 낮음을 시나리오 11에서 20까지의 시뮬레이션을 통해서 확인할 수 있었다. 그림 15에서 보면 40대의 Simulation-Client에서 Virtual-URC 클라이언트의 수를 초당 40개까지 생성하였을 경우, 시나리오 20을 제외하고 100%의 접속 및 수신 성공률을 보였다. 이후 시뮬레이션에서는 전체 Virtual-URC 클라이언트 수와 초당 Virtual-URC 클라이언트 생성 수가

많아짐에 따라서 접속 및 수신 성공률이 더 낮아짐을 알 수 있다.

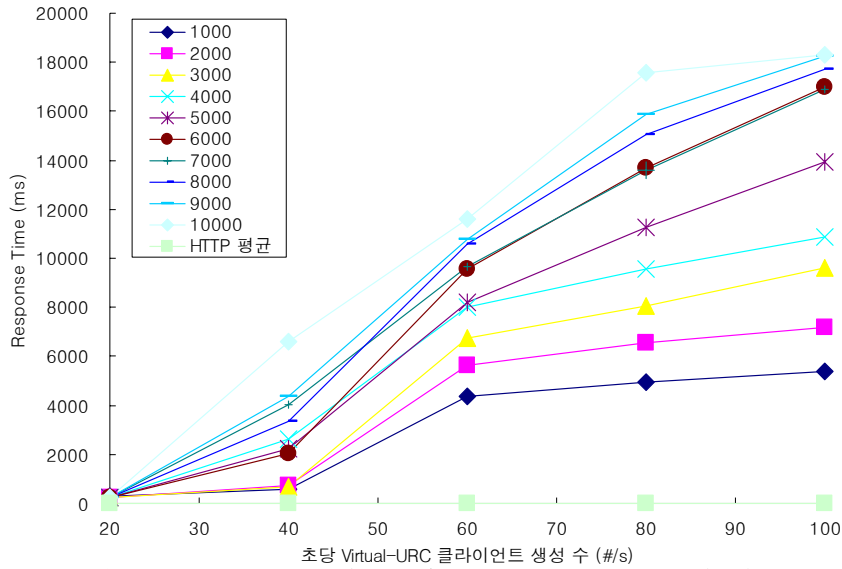


그림 14. Virtual-URC 생성률에 따른 Response time 의 비교

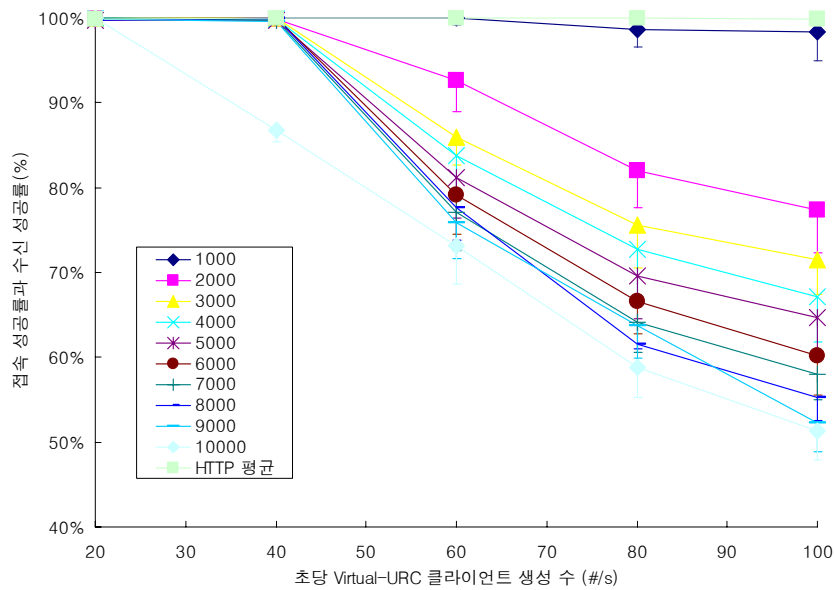


그림 15. Virtual-URC 생성률에 따른 접속 성공률 및 수신 성공률 비교

그림 16과 17은 Virtual-URC 클라이언트 생성률이 일정할 때 한 번의 시뮬레이션 동안 생성되는 전체 Virtual-URC 클라이언트 수에 따른 response time과 접속 및 수신 성공률의 결과를 보여준다.

그림 16에서 보면, Virtual-URC 클라이언트 생성률이 초당 20일 때, 전체 Virtual-URC 클라이언트 수에 관계없이 URC 서버는 Virtual-URC 클라이언트 요청의 99% 이상을 242ms 안에 처리하였다. 생성률이 초당 40일 때, URC 서버는 3,000개까지 Virtual-URC 클라이언트 요청의 99% 이상을 732ms 내에 처리했다. 하지만 그림 17에서 보면 전체 Virtual-URC 클라이언트 수가 9,000에 이르기까지는 URC 서버는 Virtual-URC 클라이언트 요구의 99% 이상을 처리했지만, response time은 전체 Virtual-URC 클라이언트 수가 증가함에 따라서 비례하며 증가 했다.

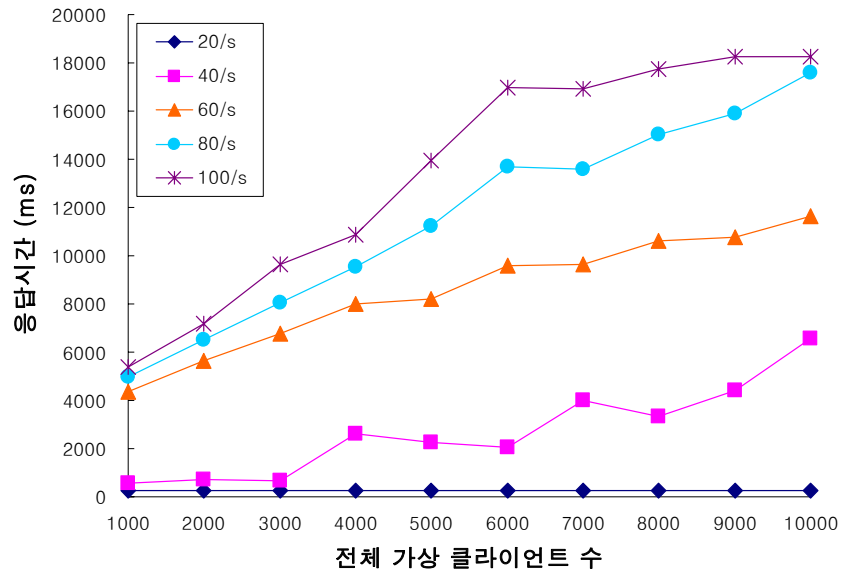


그림 16. 전체 Virtual-URC 수에 따른 Response time 비교

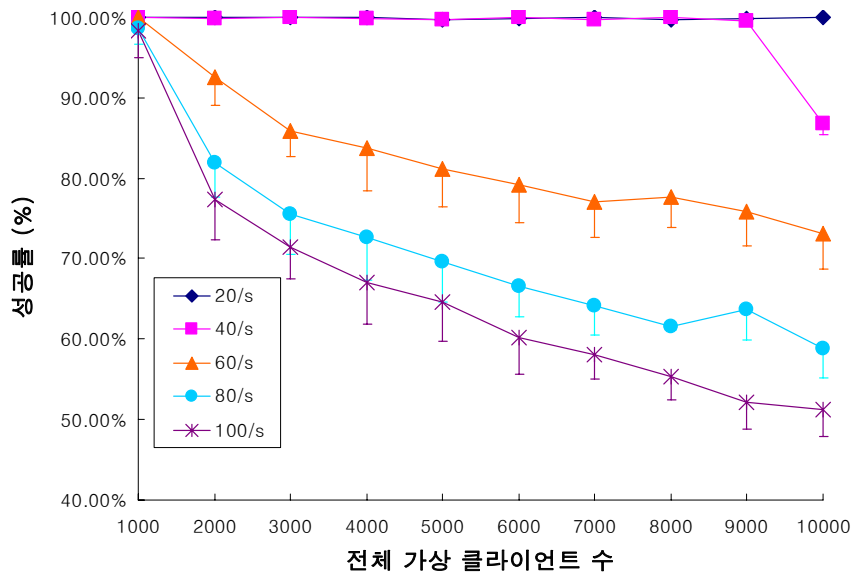


그림 17. 전체 Virtual-URC 수에 따른 접속 성공률 및 수신 성공률 비교

### 6.3 고찰

Virtual-URC 클라이언트의 수와 초당 생성률을 서로 달리하여 실험한 테스트에서 HTTP 데몬 서버는 거의 영향을 받지 않았지만(표 3), DM 어플리케이션은 파라미터에 따라서 그 차이가 확연히 드러난다(그림 14, 그림 15, 그림 16, 그림 17). OMA DM 서버의 구조(그림 4) 때문에 DM 어플리케이션 테스트 결과의 response time이 HTTP 서버보다 더 길어지는 것은 사실이다.

DM 어플리케이션 테스트는 총 네 단계를 거친다. HTTP 서버 데몬이 HTTP 메시지를 파싱하여 SyncML 메시지를 DM 어플리케이션으로 보낸다. DM 어플리케이션은 SyncML 메시지를 다시 파싱하고 클라이언트의 정보를 얻는다. 이 정보를 바탕으로 내부의 데이터베이스를 업데이트 한다. 그리고 그림 3의 (a) setup phase를 따라서 그 결과를 다시 클라이언트로 전송한다.

시뮬레이션 결과에서 HTTP 서버까지의 처리 시간은 짧은 것으로 보아 OMA DM 서버의 작업에 많은 시간이 소요되었을 것이라고 예측할 수 있다. 즉, SyncML의 파싱이나 데이터베이스의 업데이트 과정에서 병목 현상이 발생하여 response time이 증가한 것으로 예측할 수 있다.

그림 18은 시나리오 11을 통해서 한 번의 시뮬레이션 시간 동안에 Virtual-URC 클라이언트의

생성 시기를 평균적으로 보여주는 그림이다. 즉, Virtual-URC 클라이언트의 생성률이 초당 20일 경우 1,000개의 Virtual-URC 클라이언트 생성하는데 50초가 걸린다. 50초 동안에 평균 271ms를 수행하는 Virtual-URC 클라이언트 999개가 존재하려면, Virtual-URC 클라이언트 생성에 있어서 생성 시간이 약 50ms의 차이가 나와 한다. 따라서 평균 response time 동안에는 평균 약 5.4개의 Virtual-URC 클라이언트가 동시에 OMA DM 서버에 존재한다고 예상할 수 있다.

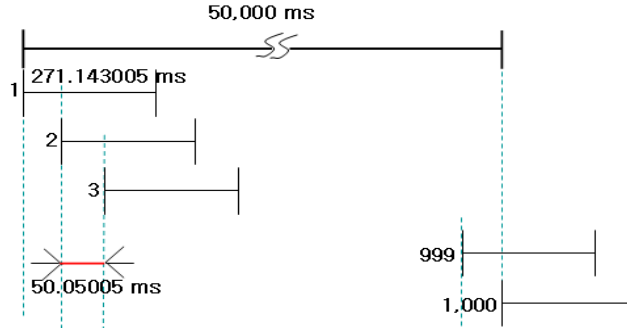


그림 18. Virtual-URC 클라이언트 생성 순서

이 계산 과정을 시나리오 20까지 확장하면, 시뮬레이션 시간 동안에 OMA DM 서버가 동시에 처리한 Virtual-URC 클라이언트의 수(scalability)를 Virtual-URC 클라이언트의 생성률에 따라서 구할 수 있다(표 4, 그림 19).

표 4. OMA DM 서버의 scalability

Scenario	Virtual-URC 클라이언트 생성률 (#/s)				
	20	40	60	80	100
11	5.417	22.333	261.506	396.671	536.179
12	5.160	29.270	337.552	522.635	716.543
13	5.253	26.216	406.078	644.079	962.414
14	5.262	105.152	479.588	763.984	1086.774
15	5.240	89.787	492.691	899.648	1394.916
16	5.326	81.246	573.859	1095.473	1699.565
17	4.974	160.334	579.456	1086.763	1690.276
18	5.022	133.939	635.403	1201.920	1771.973
19	4.781	175.627	645.819	1270.772	1824.757
20	4.843	263.192	696.862	1405.855	1827.589

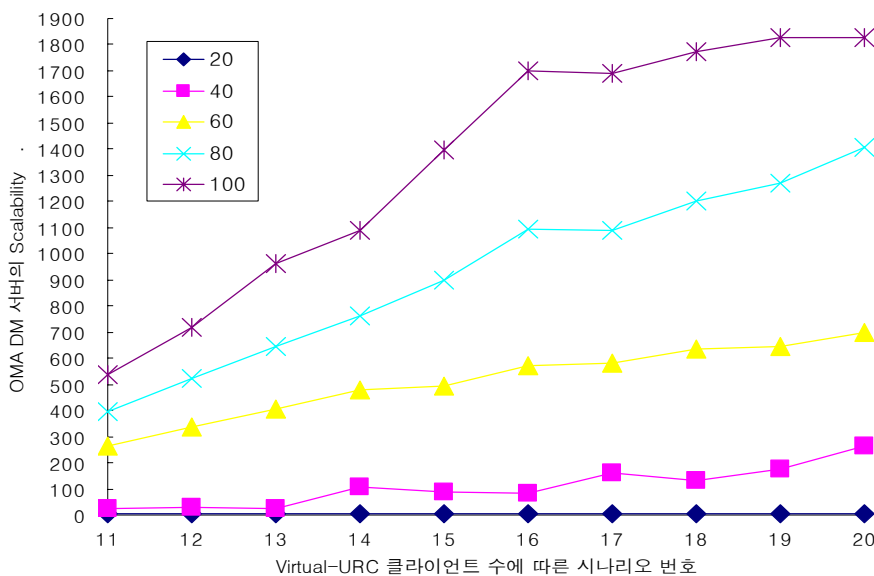


그림 19. OMA DM 서버의 scalability 비교

## 7. 결론 및 향후 연구과제

본 논문에서는 URC 로봇 서비스 시범 사업의 시작과 100만 로봇의 보급이 가시화 됨에 따라 URC 로봇 서버 시스템의 가용성 및 신뢰성 제고를 위한 통합서비스 플랫폼 기술 개발의 일환으로 OMA DM 서버의 가용성 및 신뢰성을 시험할 수 있는 시뮬레이터를 개발하였다. 시뮬레이터는 클라이언트 쪽의 실험 결과만을 가지고 OMA DM 서버의 성능을 측정하였다. 즉, 서버 내에서 성능 측정 프로그램을 따로 실행시키지 않고 실제 클라이언트와 같은 역할을 수행하는 가상의 클라이언트를 만들어 load 테스트를 하였기에 서버의 순수한 성능을 측정할 수 있다.

OMA DM 서버에서 병목 현상이 발생하고 있다는 사실을 이번에 개발한 시스템을 통한 시뮬레이션으로 알 수가 있다. 6.3장에서 SyncML 파싱 문제와 데이터베이스 업데이트 문제를 OMA DM 서버의 병목 현상의 원인으로 예측하였다. 하지만 OMA DM 서버의 각 모듈별 수행시간을 측정한 것이 아니기 때문에 어떤 부분이 전체 서버의 병목 현상을 일으켰는지 정확히 알 수는 없었다. 따라서 서버 내에서 병목 현상을 일으키는 부분을 찾는 방법이 필요하다. 그리고 그 문제를 해결하고 본 논문과 같은 시뮬레이션을 다시 수행한 후, 그 결과를 이번 논문의 결과와 비교하는 작업 역시 필요하다.

이번에 개발한 시뮬레이터는 URC 로봇 단말 관리 시스템의 OMA DM 서버만을 대상으로 load 테스트를 수행하였다. 하지만 URC 로봇 서비스 시스템은 URC 로봇 단말 인증, URC 로봇 단말 관리, URC 서비스 포탈, URC 콘텐츠 관리 등의 시스템들로 이루어져 있다. 따라서 본 시뮬레이터를 이용하여 URC 로봇 서비스를 위한 전체 서버에 대한 성능 평가 역시 이루어져야 한다.

추가적으로, 본 논문의 시뮬레이션 결과를 OMA DM 서버의 물리적 리소스와 연결 지어 생각할 수 있다. 따라서 물리적 리소스 확장을 한 서버를 대상으로 시뮬레이션을 하고 마찬가지로 이번 논문의 결과와 비교하는 작업이 필요하다.

## 참고 문헌

- [1] URC (Ubiquitous Robotic Companion), <http://www.urckorea.com>.
- [2] URC 기술협력 포럼, <http://www.urcforum.or.kr>.
- [3] OMA DM (Device Management) Working Group, [http://www.openmobilealliance.org/tech/wg\\_committees/dm.html](http://www.openmobilealliance.org/tech/wg_committees/dm.html).
- [4] OMA: OMA Device Management Standardized Objects (2007).
- [5] OMA (Open Mobile Alliance), <http://www.openmobilealliance.org/>.
- [6] OMA: OMA Device Management Protocol (2007).
- [7] SyncML Forum: SyncML Device Management Protocol, <http://www.syncml.org/>.
- [8] 천기요, 박상근, 최덕재, “사용자 관점의 웹 서비스 성능저하 원인분석 시스템 구조 설계 및 구현,” *KNOM Review*, Vol. 4, No. 1, June 2001, pp.66-75.
- [9] Daniel A. Menascé, “Load Testing of Web Sites,” *Internet Computing, IEEE*, Volume 6, Issue 4, July-Aug. 2002.
- [10] Y. Wu and J. Offutt, “Modeling and Testing Web-based Applications,” *Internet Computing, IEEE* Volume 6, Issue 4, July-Aug. 2002.
- [11] Dirk Draheim, John Grundy and John Hosking, “Realistic Load Testing of Web Applications,” Proceedings of 10th European Conference on Software, Volume 00, 22-24 March 2006 Page(s):11 pp.
- [12] Microsoft, <http://support.microsoft.com/default.aspx?scid=889654>.
- [13] Yun Koo Chung and Sun-Myung Hwang, “Software Testing for Intelligent Robots,” International Conference on Control, Automation and Systems, Seoul, Korea, October 2007.
- [14] RUPI, <http://www.rupi.or.kr>.
- [15] OSGi (Open Service Gateway initiative) Alliance, <http://www.osgi.org/Main/HomePage>.
- [16] OSGi Service Platform, Core Specification, Release 4, Version 4.1, <http://www.osgi.org/Download/Release4V41>.
- [17] WAP: (Wireless Application Protocol) WSP (Wireless Session Protocol), WAP Forum, <http://www.wapforum.org/>.



**조 성 호**

2007 부산대학교, 컴퓨터공학과 학사  
2007 ~ 현재 포항공과대학교, 컴퓨터공학과 석박사통합과정  
<관심분야> 인터넷 트래픽 모니터링 및 분석, 네트워크 관리 및 관리 시스템



**박 병 철**

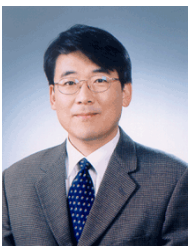
2006 포항공과대학교, 컴퓨터공학과 학사  
2006 ~ 현재 포항공과대학교, 컴퓨터공학과 석박사통합과정  
<관심분야> 인터넷 트래픽 모니터링 및 분석, 응용 트래픽 분류, 네트워크 관리 및 관리 시스템



**최 미 정**

1998 이화여자대학교, 컴퓨터공학과 학사  
1998 ~ 2000 포항공과대학교, 컴퓨터공학과 석사  
2000 ~ 2004 포항공과대학교, 컴퓨터공학과 박사  
2004 Post-Doc., Dept. of computer Science and Engineering, POSTECH, Korea  
2004 ~ 2005 Post-Doc., MADYNES Team, LORIA-INRIA Lorraine, Nancy, France  
2005 ~ 2006 Post-Doc., School of Computer Science, Univ. of Waterloo, Canada  
2006 ~ 현재 포항공과대학교, 컴퓨터공학과 연구교수.

<관심분야> XML 기반의 네트워크 관리, 에이전트 기술, 정책 기반의 네트워크 관리.



**홍 원 기**

1983 Univ. of Western Ontario, BSc in Computer Science  
1985 Univ. of Western Ontario, MS in Computer Science  
1985 ~ 1986 Univ. of Western Ontario, Lecturer  
1986 ~ 1991 Univ. of Waterloo, PhD in Computer Science  
1991 ~ 1992 Univ. of Waterloo, Post-Doc Fellow  
1992 ~ 1995 Univ. of Western Ontario, 연구교수  
1995 ~ 현재 포항공과대학교 컴퓨터공학과 교수

2005 ~ 현재 IEEE ComSoc CNOM Chair

<관심분야> 네트워크 트래픽 모니터링, 네트워크 및 시스템 관리, Network Security