디지털산업정보학회 논문지 제5권 제1호 - 2009년 3월

사람-사람, 사람-사물의 대화를 지원하는 인스턴트 메신저 구현

최 종 명*

Implementation of an Instant Messenger Supporting Human-Human and Human-Thing Communication

Choi, Jong Myung

〈Abstract〉

This paper is about the implementation of MyTalk, an instant messaging system, which supports human-human and human-thing communication. It consists of agents which are representatives of communication entities, and agent windows for user interfaces. Users can communicate with their buddies, and monitor and control remote "things" and devices with MyTalk. We also introduce the concept of agent window, which provides the most suitable GUIs (text, form, graph etc) for communication. Currently, MyTalk supports ordering goods, controlling devices like printers, monitoring sensors, managing applications, and real-time talk. Furthermore, we can extend it to application platform for remote monitoring, remote controlling, and real time cooperation.

Key Words : Human-Thing Communication, Remote Monitoring, Instant Messenger

Ⅰ. 서론

유비쿼터스 컴퓨팅 환경으로 발전할수록 장소와 시간에 관계없이 사람과 사람, 사람과 사물의 실시간 협력의 중요성이 커지고 있다. 이에 따라 사람과 사물을 인터넷을 통해서 연결하기 위한 연구에 대한 관심이 높아지고 있고, 이와 관련된 연구 및 학술대회들이 점차 많아지고 있다[1, 2]. 컴퓨팅 환경에서 사람간의 실시간 대화와 협력은 인스턴트 메신저를 통해 이루어지고 있으며, 이는 일상생활과 업무에서 상당히 큰 비중을 차지하고 있다[3-6]. 사람과 사물의 대화와 협력의 예로는 원격 모니터링, 원격 제어 등이 있으며, 이러한 유형의 대화는 사용자의 편의성 때문에 점차 중요해지고 있다.

사람-사람, 사람-사물의 대화와 협력의 중요성은 부각되고 있지만, 현재까지 이러한 문제를 해결하기 위한 호환성 있는 방법에 대한 연구가 진행되지 않았다. 원격 모니터링[7], 핸드폰 혹은 인터넷을 이용한 원격 제어[8], 사람과 컴퓨터의 상호작용[9] 등은 컴퓨팅 능력을 갖춘 장비와 사람간의 실시간 대화를 지원하기 위한 연구들이지만, 지금까지 연구들은 서로 독립적으로 진행되어 왔고, 호환되지 않는 문제점을 갖고 있다. 또한 대화의 대상이 컴퓨팅 능력을 갖춘 장비에 국한되는 단점이 있다.

본 논문에서는 선행연구[10]에서 소개한 MyTalk이라

* 국립목포대학교 정보공학부 컴퓨터공학 교수

는 인스턴트 메신저의 구현에 관해서 소개한다. MyTalk는 기존의 인스턴트 메신저를 확장해서 사람-사람과 사람-사물의 대화와 협력을 지원할 수 있다. 이를 위해서 MyTalk는 사람과 사물을 대표할 수 있는 에이전트를 두고, 에이전트를 통해서 사물에 대한 정보 검색, 모니터링, 제어, 협력 등의 작업을 수행할 수 있다. MyTalk의 에이전트는 컴퓨팅 파워가 없는 사물은 물론 논리적인 개념의 컴퓨팅 서비스 혹은 응용프로그램에도 적용할 수 있는 장점을 갖고 있다. 이러한 장점들을 바탕으로 인스턴트 메신저를 이용한 사람-사람, 사람-사물의 실시간 대화와 협력을 일관성 있고, 호환성 있는 방법으로 사용할 수 있는 방법을 제공한다.

MyTalk는 기존 연구들에 비해서 다음과 같은 의의를 갖는다. 첫째로 MyTalk는 실시간 대화의 영역을 사람에서 임의의 엔티티(entity)로 범위를 넓혔다. 이에 따라 다양한 엔티티를 활용한 새로운 형태의 서비스들을 개발할 수 있는 기초를 제공한다. 둘째로 MyTalk는 사람-사람, 사람-사물과의 대화를 통일되고 일관된 방식으로 지원할 수 있는 방법을 제시한다. 따라서 메신저를 대화, 모니터링, 제어 등을 위한 응용프로그램의 플랫폼으로 활용할 수 있다. 셋째로 MyTalk의 엔티티 확장은 새로운 형태의 연구를 촉진시킬 것이다. 즉, 기존의 사람-사람 사이의 통신에서 발생하는 사회적 연구[3-6]가 사람-사물에서 발생하는 사회적 연구로 발전할 수 있는 기초를 제공한다.

본 논문은 2장에서는 본 연구와 유사한 관련연구들을 소개하고, 3장에서는 MyTalk 시스템에 대한 개요를 소개한다. 4장에서는 MyTalk 시스템의 구성에 대해서 기술하고, 5장에서는 구현 및 평가한 내용을 소개한다. 마지막으로 6장에서는 결론 및 향후 연구를 밝힌다.

II. 관련 연구

인스턴트 메신저와 관련된 연구들은 크게 세 가지 유형으로 분류할 수 있다. 첫째 유형은 메신저의 사회적 영향에 관련된 내용을 연구하는 것들이다. 이러한 유형의 연구들로는 메신저를 사용하는 이유[4], 업무에서 메신저를 사용하는 유형[3, 4, 11], 십대들이 메신저를 이용해서 사회 연결망을 구성하는 방법[5] 등이 있다. 이러한 연구들은 주로 사회과학 분야에서 진행되고 있으며, 메신저의 역할, 중요성, 사용 패턴 등에 대해 관심을 갖는다.

둘째 유형의 연구는 인스턴트 메신저에 새로운 형태의 사용자 인터페이스를 추가하기 위한 연구들이다. 첫째로 Ellen Isaacs[12]은 인스턴트 메신저에 음성 인터페이스, Ian Oakley[13]과 Rovers[14]은 촉각 인터페이스를 추가하려고 시도했다. 또한 Susannah McPhail [15]는 인스턴트 메신저에 물리적 인터페이스를 장착시키는 연구를 시도하였다. 둘째 연구는 데스크탑의 GUI에서 벗어나 다양한 인터페이스를 활용하기 위한 시도로 사용자 관점에서 흥미롭지만, 본 연구와는 시스템 목적과 범위의 차이가 있다. 세 번째 형태의 연구는 인스턴트 메신저의 대화 대상을 확장하기 위한 연구들이다. Adams[16]는 메신저와 메일 시스템을 연결함으로써 메신저에 메일 수신을 알리는 기능을 추가하였다. Arjun[17]는 인터넷 가전기기와 통신할 수 있는 메신저에 대한 사용 시나리오를 소개하였으며, Simon Aurell[8]는 홈 오토메이션에서 원격 기기를 제어하고 모니터링하는 인스턴트 메신저를 소개하였다. Cian Foley[18]는 사람-사람, 사람-디바이스, 사람-서비스를 연결할 수 있는 인스턴트 메신저를 소개하였다.

세 번째 유형의 연구는 대화 대상을 확장한다는 점에서 본 연구와 상당한 유사점을 갖고 있다. 그러나 본 연구와는 세 가지 측면에서 차이점을 갖는다. 첫째로 MyTalk 시스템은 컴퓨팅 능력이 없는 임의의 사물 혹은 응용프로그램에도 적용할 수 있다. 둘째로 본 연구는 사용자의 편의성을 위한 에이전트 윈도우 개념을 제공한다. 셋째로 MyTalk에서는 새로운 엔티티를 쉽게 추가할 수 있도록 프레임워크를 제공한다.

Ⅲ. 대화 및 협력의 창구로서 메신저

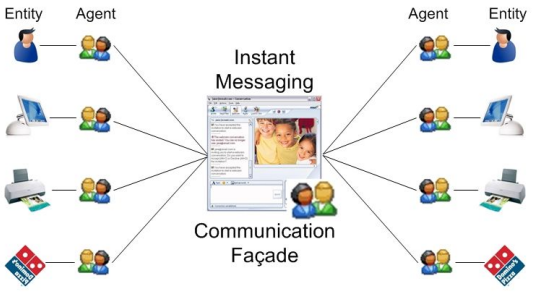
인스턴트 메신저는 기본적으로 사람과 사람 사이에서 텍스트 기반 실시간 의사소통을 지원하며, 부가적으로 파일 전송, 컨퍼런스, 화상 통신 등의 기능을 제공한다. 또한 메신저는 데스크톱은 물론 웹[26]과 모바일 플랫폼[19]에서 지원하고 있기 때문에 활용도가 더욱 높아지고 있다.

선행 연구[10]에서 밝혔듯이 인스턴트 메신저는 사람-사람, 사람-사물 사이의 통신에서 게이트웨이의 역할을 할 수 있다. 인스턴트 메신저를 통해 서로 대화할 수 있는 개체(사람 혹은 사물)를 통신 엔티티(entity)라고 부른다. 각 통신 엔티티는 인스턴트 메신저 대화를 담당할 수 있는 에이전트를 가져야 하고, 에이전트를 통해서 다른 통신 엔티티와 대화가 가능하다.

통신 엔티티는 능동 엔티티, 피동 엔티티라는 2가지 형태로 분류할 수 있다. 능동 엔티티는 일반적으로 컴퓨팅 파워를 갖고, 스스로 자체 서비스를 수행할 수 있는 엔티티를 의미한다. 사람은 메신저를 이용해서 능동 엔티티의 상태를 모니터링하거나 제어할 수 있다. 또한 능동 엔티티는 스스로 필요하다고 판단하는 경우에 사용자와 대화를 시작할 수 있다. 이러한 엔티티의 예로는 데스크톱 컴퓨터, 프린터 등이 있다.

피동 엔티티는 자체적으로 오퍼레이션을 수행할 수 없는 물리적 혹은 논리적인 엔티티를 의미한다. 피동 엔티티는 크게 3가지 유형으로 분류할 수 있다. 첫째는 컴퓨팅 파워가 없는 물리적 사물로서 사용자는 메신저를 이용해서 사물의 메타 정보나 상태 정보를 파악할 수 있다. 둘째는 데이터베이스 관련 응용프로그램이다. 이러한 형태의 경우에는 데이터 입력, 검색 등의 작업을 수행할 수 있다. 셋째는 유틸리티성 응용프로그램이다. 메신저를 이용해서 응용프로그램을 실행시키고, 사용할 수 있다. 예를 들어, 계산기 등이 이러한 부류에 포함된다.

인스턴트 메신저는 엔티티의 종류에 관계없이 동일한 형태로 서로 대화가 가능하도록 지원할 수 있다. <그림 1>은 메신저를 통해서 각 통신 엔티티들이 동일한 방법으로 통신할 수 있다는 것을 보여준다. 즉, 엔티티들은 에이전트를 통해서 텍스트 기반의 메시지를 상대방 에이전트에 전송한다. 이때 엔티티가 사람이건 프린트이건 메시지 전달 측면에서는 동일하다. 엔티티에 따라 달라지는 부분은 에이전트이다. 에이전트는 텍스트 메시지를 엔티티가 이해할 수 있는 형태로 변환해서 전달하는 역할을 한다.



<그림 1> 통신 게이트웨이로서 인스턴트 메신저

메신저를 통해서 동일한 방식으로 엔티티들이 실시간 대화를 할 수 있다는 것은 많은 장점을 갖는다. 첫째는 다양한 통신 프로토콜을 하나로 통일할 수 있다는 의미이다. 즉, N개의 엔티티가 있을 때 서로 다른 프로토콜을 사용한다면, N x (N-1)개의 프로토콜을 지원하는 시스템을 개발해야 하지만, 메신저를 사용하는 경우에 N개의 에이전트와 1개의 메신저 프로토콜로 문제를 해결할 수 있다. 이것은 시스템의 복잡도를 줄일 수 있을 뿐만 아니라 시스템 개발의 비용 및 노력을 대폭 감소시킬 수 있다는 장점을 갖고 있다.

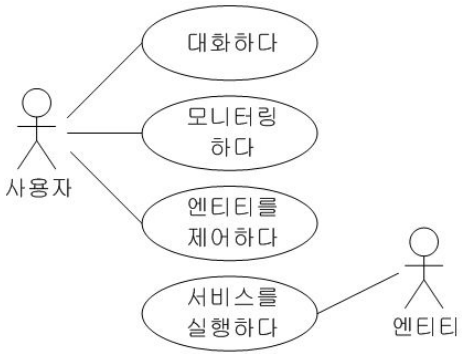
둘째는 복잡한 통신 패스를 간소화할 수 있다는 의미이다. 예를 들어, N개의 엔티티가 있는 경우에 메신저를 사용하지 않고, 개별적으로 연결하는 경우에 N x (N-1)개의 통신 패스를 메신저를 사용하는 경우에 N개로 줄일 수 있다.

셋째로 응용프로그램들을 하나로 통합할 수 있다. 별도로 존재했던 원격 모니터링, 원격 제어, 원격 엔티티에

대한 정보 검색 등의 작업을 하나의 시스템에서 작업할 수 있다. 이것은 사용자 관점에서는 편리함을 제공하고, 개발자 입장에서는 보다 쉽게 개발할 수 있는 여건을 제공한다. 즉, 개발자는 인스턴트 메신저 플랫폼을 기반으로 응용프로그램을 작성할 수 있기 때문에 보다 빨리 원하는 시스템을 작성할 수 있다.

Ⅳ. MyTalk 시스템 구성

MyTalk은 사람-사람, 사람-사물의 실시간 대화와 협력을 지원하는 인스턴트 메신저 시스템이다. MyTalk의 기본적인 기능적 요구사항은 <그림 2>와 같이 유스케이스 다이어그램으로 표현할 수 있다. MyTalk를 통해서 사용자는 엔티티와 대화, 모니터링, 제어 등의 작업을 수행할 수 있다.

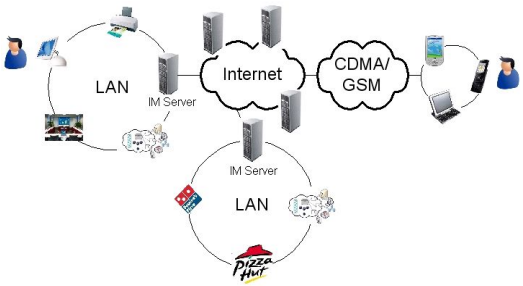


<그림 2> MyTalk의 유스케이스

MyTalk의 비기능적 요구사항들로는 기능 확장성(extensibility), 사용의 용이성, 범위 확장성(scalability) 등이다. MyTalk에서 가장 중요한 비기능적 요구사항은 기능 확장성인데, 이것은 MyTalk에 다양한 엔티티를 추가할 수 있어야 하기 때문이다.

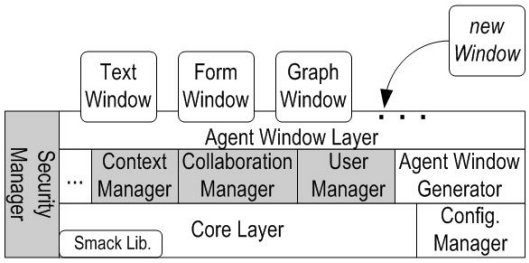
- 기능 확장성 : 새로운 통신 엔티티를 쉽게 추가할 수 있어야 한다.
- 사용의 용이성 : MyTalk 시스템은 쉽게 사용할 수 있어야 한다.
- 범위 확장성 : MyTalk 시스템은 대규모 사용자와 엔티티를 지원할 수 있어야 한다.

MyTalk 시스템은 peer-to-peer 시스템이기 때문에 시스템 아키텍처는 서버측과 클라이언트 측으로 구분할 수 있다. MyTalk의 서버측은 peer-to-peer 아키텍처를 지원하기 때문에 범위 확장성을 만족시킬 수 있다. <그림 3>은 MyTalk의 서버측 구성을 보여준다. 랜으로 연결된 환경에서 사용자는 데스크톱 컴퓨터를 통해서 대화가 가능하고, 외부에서는 CDMA 혹은 GSM 망을 사용하는 모바일 단말기를 통해서 대화가 가능하다.



<그림 3> 메신저 서버 구성

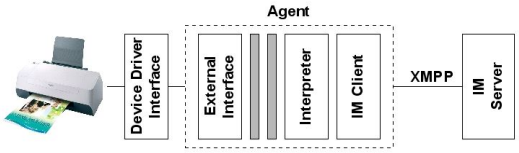
클라이언트측은 기능 확장성과 사용의 용이성을 지원하는 구조를 가져야 한다. 클라이언트는 사용자를 위한 MyTalk Client와 엔티티를 대표하는 에이전트로 구성된다. <그림 4>는 MyTalk Client의 구조를 보여준다. MyTalk Client는 복잡성을 줄이기 위해서 레이어 구조를 갖는다. 가장 하단에는 통신과 구성을 관리하기 위한 Core Layer가 존재하고, 중간에는 다양한 인공지능적인 기능들을 제공하기 위한 Smart Layer가 존재한다. 가장 상단은 Agent Window Layer로서, 에이전트 윈도우를 관리하기 위한 레이어이다. 그림에서 음영으로 표시된 부분은 현재 구현 중에 있는 부분들이다.



<그림 4> MyTalk Client 구조

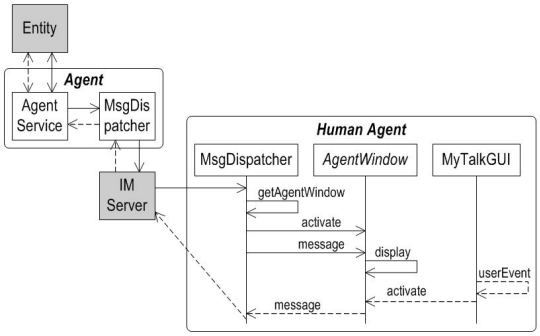
MyTalk는 사용자의 용의성을 위해서 에이전트 윈도우 개념을 제공한다. 에이전트 윈도우는 에이전트의 기능에 따라 사용자가 사용하기 적합한 형태의 GUI를 의미한다. 에이전트 윈도우의 예로는 일반적인 텍스트 기반의 대화창, 데이터를 입력할 수 있는 폼 형태의 윈도우, 모니터링 정보를 시각적으로 보여주는 그래프 윈도우 등이 존재할 수 있다. 새로운 엔티티가 추가되는 경우에 이를 위한 에이전트 윈도우가 동적으로 설치 혹은 자동적으로 생성되어야 하는데, MyTalk에서는 이를 위해 2가지 방법을 제공한다. 첫 번째는 Agent Window Layer에 Jar 형태로 작성된 에이전트 윈도우를 동적으로 플러그인 시킬 수 있는 방법이다. 두 번째는 엔티티를 위한 윈도우를 자동적으로 생성하는 방법이다. 이는 Agent Window Generator를 통해서 자동적으로 생성한다[22].

MyTalk에서 에이전트는 사람-사물의 대화에서 핵심적인 역할을 한다. 이것은 에이전트가 통신 엔티티에 대한 언어를 이해할 수 있도록 해석해주고, 엔티티의 정보를 전달해주며, 엔티티를 제어할 수 있도록 지원하기 때문이다. 에이전트는 IM 클라이언트 모듈, 인터프리터, 외부 인터페이스 모듈 등으로 구성된다. <그림 5>는 프린터 에이전트 구조를 보여준다. IM 클라이언트 모듈은 메신저 통신에 대한 기능을 제공하고, 인터프리터는 메시지를 해석하고, 이를 프린터가 이해할 수 있는 형태로 외부 인터페이스 모듈로 전달한다. 외부 인터페이스 모듈은 디바이스 드라이버를 통해서 직접적으로 프린터를 모니터링 혹은 제어한다.



<그림 5> 에이전트 구조

효과적인 대화를 위해서는 엔티티와 에이전트 윈도우는 서로 쌍이 맞아야 한다. 즉, 프린터와 대화하기 위해서는 프린터를 위한 에이전트 윈도우를 사용해야 한다. MyTalk Client는 엔티티에 따라서 해당 엔티티에 해당되는 에이전트 윈도우를 활성화시키는 기능을 갖추고 있다. 이때 엔티티와 MyTalk Client 사이에는 <그림 6>과 같은 형태로 메시지 전송과 작업이 진행된다. 그림에서 실선 화살표는 엔티티에서 사용자 에이전트에 메시지를 전송하는 경우를 보여주고, 점선 화살표는 사용자의 입력이 엔티티에 전달되는 경로를 보여준다. MsgDispatcher 클래스는 인스턴트 메신저 서버로부터 받은 메시지를 적합한 에이전트 윈도우에 전달하는 역할을 한다. 그림에서 음영으로 표시된 부분은 외부 시스템을 의미한다.



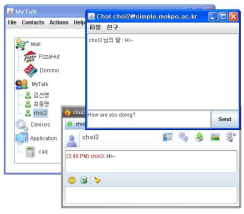
<그림 6> 엔티티와 MyTalk Client의 메시지 전송

V. 구현 및 평가

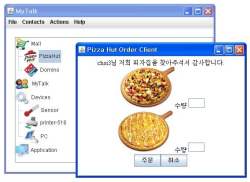
MyTalk 시스템은 Win32 플랫폼에서 주로 자바 언어를 이용해서 개발하였다. 메신저 프로토콜은 Jabber[20]

의 XMPP(Extensible Messaging and Presence Protocol) [23]를 사용하였고, 메신저 서버는 Openfire[21]를 사용하였다. 클라이언트 측에서 XMPP와 관련된 라이브러리는 Smack 라이브러리[24]를 사용하였다. 프린터 에이전트는 윈도우 MFC를 사용하였고, 센서 에이전트는 TinyOS 기반의 Hybus의 Hmote[27]를 사용하였다.

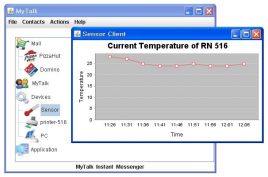
MyTalk 시스템을 사용하는 경우에 다양한 형태의 엔티티와 대화를 할 수 있다. <그림 7>은 MyTalk 메신저가 실행되는 화면이다. (a)는 텍스트 기반의 대화창 윈도우이다. 사람간의 대화에 주로 사용되며, 그림 (a)에서는 MyTalk와 Spark [25] 사용자의 대화를 보여준다. (b)는 온라인 주문 시스템과의 대화를 보여준다. 주문 내용은 주문 시스템의 에이전트로 전달된다. (c)는 센서 에이전트가 현재 방의 온도를 메신저를 통해서 전달하고, 센서 에이전트 윈도우는 이 내용을 그래프 형태로 출력한다. (d)는 로컬 컴퓨터의 응용프로그램을 등록하고, 필요시에 쉽게 실행시키는 것을 보여준다.



(a) 사람간의 실시간 대화



(b) 주문 시스템과 대화



(c) 센서 모니터링

(d) 응용프로그램

<그림 7> MyTalk 실행 화면

MyTalk 시스템의 평가는 유용성을 기반으로 테스트하였다. <표 1>은 MyTalk 시스템에서 구현 및 테스트한 엔티티들의 기능과 장점을 기술한 것이다. 엔티티의 종류와 사용 형태가 많아질수록 인스턴트 메신저를 이용해서 사람-사물의 협력과 대화를 활용할 수 있는 방법들이 더욱 개발될 것이다.

<표 1> MyTalk 에이전트의 기능 및 평가

엔티티 종류	기능	장점 및 비고
데스크톱	원격 컴퓨터 접근 및 제어	메신저를 지원하는 임의의 플랫폼(예: 모바일 단말기, 웹, 다른 플랫폼의 컴퓨터)과 장소에서 접근 및 사용 가능
프린터	원격 프린터 접근	다른 네트워크에서도 사용 가능
주문 등록	데이터 입력 및 검색	편리한 접근 및 사용
센서	센서 값 모니터링	편리한 모니터링
응용 프로그램	응용프로그램 실행	편리한 접근 및 사용

VI. 결론 및 향후 연구

유비쿼터스 환경에서 사람-사람, 사람-사물의 실시간 대화와 협력은 매우 중요한 요소이다. 특히 사람과 사물의 대화와 협력은 더욱 활성화되고, 일반적으로 사용하게 될 것이다. 현재 통신 엔티티에 대한 원격 정보 검색, 모니터링, 제어 등의 점차 일반화되고 있으며, 이를 이용하는 시스템들이 점차 많아지고 있다.

본 논문에서는 인스턴트 메신저를 이용해서 사람-사람, 사람-사물의 실시간 대화와 협력을 지원할 수 있는 MyTalk 시스템의 구현 내용에 대해서 소개하였다. MyTalk는 통신 엔티티를 대표하는 에이전트를 두고, 에이전트를 통해서 실시간 대화를 제공할 수 있다. MyTalk는 확장성과 사용의 용이성을 위해서 에이전트 윈도우의 개념을 도입하였으며, 에이전트를 쉽게 개발할 수 있도록 라이브러리들을 제공한다. 향후 현재 MyTalk 시스템

에서 구축이 미진한 컨텍스트 기능과 협력 부분에 대한 내용들을 연구 및 구현할 것이다.

참고문헌

[1] Internet of Things 2008, available at http://www.iot2008.org/

[2] ITU Internet Reports 2005: The Internet of Things, available at http://www.itu.int/osg/spu/publications/internetofthings/

[3] Eulynn Shiu and Amanda Lenhart, How Americans use instant messaging, Pew Internet & American Life Project, Sep., 2004, available at http://www.pewinternet.org/.

[4] Ellen Isaacs, et al., "The character, functions, and styles of instant messaging in the workplace," ACM Conf. on Computer Supported Cooperative Work, 2002, pp. 11-20.

[5] Rebecca E. Grinter and Leysia Palen, "Instant messaging in teen life," ACM Conf. on Computer Supported Cooperative Work, 2002, pp. 21-30.

[6] Elaine M. Huang, Daniel M. Russell, and Alison E. Sue, "IM here: public instant messaging on large, shared displays for workgroup interactions," Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, 2004, pp. 279-286.

[7] Ross S. Lunetta and Christopher D. Elvidge, Remote Sensing Change Detection, CRC Press, 1999.

[8] Simon Aurell, "Remote Controlling Devices Using Instant Messaging," Proc. of the 2005 ACM SIGPLAN workshop on Erlang, 2005, pp. 46-51.

[9] Alejandro Jaimesa and Nicu Sebeb, "Multimodal human‐computer interaction: A survey," Proc. of IEEE International Workshop on Human-Computer Interaction, IEEE, 2005, pp. 15-21.

[10] Jongmyung Choi and Chaewoo Yoo, "Connect with Things through Instant Messaging," Proc. of Internet of Things, LNCS 4952, Springer, 2008.

[11] Bonnie A. Nardi, Steve Whittaker, and Erin Bradner, "Interaction and outeraction: instant messaging in action," ACM Conf. on Computer Supported Cooperative Work, 2000, pp. 79-88.

[12] Ellen Isaacs, Alan Walendowski, and Dipti Ranganthan, "Hubbub: a sound-enhanced mobile instant messenger that supports awareness and opportunistic interactions," Proc. of Human Factors in Computing Systems, ACM, 2002, pp. 179-186.

[13] Ian Oakley and Sile O'Modhrain, "Contact IM: Exploring Asynchronous Touch Over Distance," Proc. of CSCW, 2002.

[14] A.F. Rovers and H.A. van Essen, "HIM: a framework for haptic instant messaging," Proc. of Human Factors in Computing Systems, ACM, 2004, pp. 1313-1316.

[15] Susannah McPhail, "Buddy Bugs: A Physical User Interface for Windows Instant Messenger," Western Computer Graphics Symposium (Skigraph'02), 2002.

[16] Adams, "You Have Mail!," 2001, available at, http://www.openp2p.com/.

[17] Arjun Roychowdhury and Stan Moyer, "Instant Messaging and Presence for SIP Enabled Networked Appliances," Proc. of Internet Telephony Workshop, 2001.

[18] Cian Foley, Eamonn de Leaster, Sven van der Meer, and Barry Downes, "Instant Messaging as a

Platform for the Realisation of a true Ubiquitous Computing Environment," Proc. of eChallenges, 2005.

[19] Ellen Isaacs, Alan Walendowski, and Dipti Ranganathan, "Mobile instant messaging through Hubbub," Comm. of ACM, Vol. 45, No. 9, 2002, pp. 68-72.

[20] Jabber: Open Instant Messaging and a Whole Lot More, Powered by XMPP, available at http://www.jabber.org/.

[21] Openfire, available at http://www.jabber.org/ servers/openfire.

[22] Jongmyung Choi, "Vocabulary-Driven Dynamic Generation of GUIs for Human-Thing Communication," Proc. of APIS, 2008.

[23] Extensible Messaging and Presence Protocol (XMPP), 2004, available at http://www.xmpp.org/.

[24] Smack API, available at http://www.igniterealtime. org/

[25] Sparck IM Client, available at http://www.igniterealtime.org/

[26] MSN Web Messenger, available at http://webmessenger. msn.com/.

[27] Hmote, available at http://www.hybus.co.kr/.

■ 저자소개 ■

최 종 명
Choi Jong Myung

2004년 3월~현재 국립목포대학교 정보공학부 컴퓨터공학 교수
2003년 8월 숭실대학교 컴퓨터학과 (공학박사)
1996년 8월 숭실대학교 전자계산학과 (공학석사)
1992년 2월 숭실대학교 전자계산학과 (공학사)

관심분야 : 프로그래밍 언어, 유비쿼터스 컴퓨팅, 컨텍스트-인지 시스템
E-mail : jmchoi@mokpo.ac.kr

논문접수일 : 2009년 1월 9일
수 정 일 : 2009년 1월 25일
게재확정일 : 2009년 2월 5일