인터넷 상에서의 동적인 협업 환경의 지원을 위한 소프트웨어 구조

(A Software Architecture for Supporting Dynamic Collaboration Environment on the Internet)

이 장 호 †

(Jang Ho Lee)

요 약 인터넷 기반의 과학 연구 협업 환경은, 구현 경험에 의하면, 사용자가 확장할 수 있어야 하고, 작업 공간에 도구 및 객체들을 동적으로 추가할 수 있어야 하고, 작업을 개인 작업 공간과 공유 작업 공간 사이에서 이동할 수 있어야 하며, 인터넷 상에서 쉽게 접근이 가능해야 한다. 본 논문에서는 그러한 요구사항을 만족시키기 위한, Collaboratory Builder's Environment(CBE) 라고 불리는, 협업 환경을 구축하기 위한 개발 환경의 소프트웨어적 구조를 제시한다. CBE는 협업 환경을 협력적인 애플릿(collaborative applet)들로 구성함으로써, 사용자 확장성을 제공한다. 공유 작업 공간의 동적인 재구성의 지원을 위해, CBE는 애플릿, 사용자 및 임의의 데이터 객체를 포함할 수 있는 룸(room)이라는 은유적인 개념을 사용한다. 룸은 지속성을 지원함으로써, 동기적인 협업뿐만 아니라 비동기적인 협업도 지원할 수 있다. 인터넷 상에서의 접근을 위해, 룸의 구성원들은 적절한 권한의 역할(role)을 가진다. 제시된 모델의 프로토타입은 Java로 구현되었으며 Java를 지원하는 웹 브라우저를 이용하여 실행할 수 있다. 구현된 시스템은 4일간 진행된 과학적 협업 활동에서 전 세계의 79명의 우주과학자들을 포함한 95명의 사용자들에 의해 사용되었다. 그 협업 활동의 사용 분석도 제시한다.

키워드 : 그룹웨어, 컴퓨터 지원 협동 작업(CSCW:Computer Supported Cooperative Work), 공유 작업 공간, 웹 기반의 협업, 그룹 통신

Abstract Our experience with Internet-based scientific collaboratories indicates that they need to be user-extensible, allow users to add tools and objects dynamically to workspaces, permit users to move work dynamically between private and shared workspaces, and be easily accessible on the Internet. We present the software architecture of a development environment, called Collaboratory Builder's Environment(CBE), for building collaboratories to meet such needs. CBE provides user extensibility by allowing a collaboratory to be constructed as a collection of collaborative applets. To support dynamic reconfiguration of shared workspaces, CBE uses the metaphor of room that can contain applets, users, and arbitrary data objects. Rooms can be used not only for synchronous collaboration but also for asynchronous collaboration by supporting persistence. For the access over the Internet, room participants are given different roles with appropriate access rights. A prototype of the model has been implemented in Java and can be run from a Java-enabled Web browser. The implemented system had been used by 95 users including 79 space scientists around the world in a scientific campaign that ran for 4 days. The usage evaluation of the campaign is also presented.

Key words : Groupware, CSCW, shared electronic workspaces, Web-based collaboration, group communication

1. 서 론

본 논문에서는 광역 네트워크 상에서의 동기적인 과학 연구 협업을 지원하는 데 있어서의 중요점들을 조사하기 위한, UARC(Upper Atmospheric Research Collaboratory)[1]라고 불리는 실험적인다기관 테스트베드의 개발

· 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2002-003-D00254)

† 정 회 원 : 홍익대학교 컴퓨터공학과 교수
janghol@cs.hongik.ac.kr

논문접수 : 2002년 9월 26일
심사완료 : 2003년 1월 23일

및 사용경험에 대해 소개하고자 한다. NeXT 기반의 UARC 프로토타입은 사용자들에게 과학 데이타의 접근을 제공하고, DistView[2]를 이용하여, 텔레포인팅을 지원하는 동기적인 데이타 디스플레이 윈도우를 관리하며, 다중 사용자 채트(chat)와 같은 협업 도구를 제공한다.

UARC의 다양한 프로토타입의 성공적인 사용으로 말미암아 UARC에서 사용되는 설비들을 다시 디자인하여, 추가적인 협업환경을 구축하는 데 쓰일 수 있도록 하였다. 본 논문은 협업 환경의 다음과 같은 요구를 만족시키는데 쓰인 해결책들을 기술한다.

- 공유 작업공간에서의 다양한 도구[예를 들면, 특정 도메인을 위한 뷰어(viewer), 다중 사용자 채트, 공유 화이트보드(whiteboard) 등]의 지원
- 개인 및 다중 공유 작업공간의 지원과, 도구와 데이타를 작업공간사이에 동적으로 움직일 수 있는 능력
- 다양한 플랫폼으로부터 웹 상의 협업 환경으로의 접근
- 사용자 확장성의 지원. 사용자들은 협업 도구 및 URL(Uniform Resource Locator)를 협업 목적에 맞게 그들의 도구 모음(suite)에 포함시킬 수 있다.
- 관찰자(observer)와 같은 사용자 역할(role)을 지원함으로써 협업으로의 웹 상에서의 개방된 접근이 과학연구에 영향을 끼치지 못하게 한다.
- 접근 제어 (특히 웹 상에서의 사용시) 메커니즘의 제공

본 논문에 제시된 연구는 CBE(Collaboratory Builder's Environment)라고 불리는, 확장성 있는 협업 환경 구축을 위한 도구의 일부를 보여준다. CBE는 UARC 시스템의 Java 기반 버전을 구축하는데 쓰이고 있다.

확장성을 지원하기 위해, CBE는 클라이언트 애플리케이션 소프트웨어를 애플릿(작은 애플리케이션)들의 집합으로 구성하고 있다. 애플릿들에는, 특정 도메인 서비스(UARC에서의 데이타 뷰어 등)나, 도메인과는 무관한 도구(다중 사용자 채트 및 공유 화이트보드) 등, 협업에 유용한 것들이 있다. CBE 는 사용자들이 새로운 애플릿이나 URL들을 동적으로 협업에 포함시킬 수 있도록 허용한다.

CBE는 룸(room)이라는 은유(metaphor)를 써서, 다수의 애플릿과 데이타 (URL)로 구성된, 공유 작업공간을 나타낸다. 본 논문은, 애플릿들의 집합을 관리하는 상위 수준의 메커니즘을 설명하여 개인 룸과 공유 룸이 간단하게 만들어질 수 있다는 것을 보여준다. 사용자들은 동적으로 룸들을 만들 수 있다. 사용자들은 또한 현 상태 및 데이타를 가진 애플릿을 포함한 진행중인 작업들을, 다른 사용자들과의 공유 및 룸들의 사용 목적에 따라, 개인 작업공간과 공유 작업공간사이로 옮길 수 있다.

CBE는 보통 웹 브라우저를 통해 접근할 수 있는 협업 환경을 구축하기 위하여 설계되었다. 모든 클라이언트 소프트웨어인 애플릿들은 Java로 쓰여 있어서 Java를 지원하는 일반 브라우저에서 실행될 수 있다.

특히 성능 면에 있어서의 확장성을 제공하고 사용자의 룸으로의 접근을 제어하기 위하여, CBE의 접근 제어 모델은 사용자 역할(role)을 지원한다. 사용자는 각자, 공유 작업환경에서의, 관찰자(observer), 구성원(member), 관리자(administrator)와 같은 역할을 가질 수 있다. 다른 역할을 가진 사용자는, 룸으로의 다른 접근제어 권리(access control right)를 가지며, 네트워크 고장 시 다른 수준의 신뢰 보장을 제공받는다. 이 모델에서는 관찰자들이 협업을 방해할 수 없기 때문에, 과학자들이 협업 환경이 인터넷상에서 개방되는 것에 대해 수용적이 된다. 본 논문에서는 작업공간 수준의 접근 제어가 통신 구조의 하위 수준의 설계에 미치는 영향을 조사한다.

CBE에서의 룸들은 비동기적(asynchronous)[3]인 작업 및 동기적(synchronous)인 작업들을 지원하는데 사용될 수 있다. 본 논문은 사용자의 작업 공간이 여러 개의 애플릿으로 이루어진 경우의 작업공간의 공유에 관한 문제를 언급한다. CBE의 룸의 상태는 협업 세션간에 지속되므로 룸들은 비동기적인 협업에 사용될 수 있다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 먼저 관련된 협업 시스템 연구를 소개한다. 그 다음에, CBE의 기본 개념들인 룸, 애플릿 그룹, 애플릿, 사용자 그리고 그들간의 관계를 설명한다. 그리고, 현재의 설계 및 구현을 설명하고, 접근제어를 기술하며, CBE를 사용하여 구축된 실제 협업 환경의 사용 예를 보여줌으로써 그 개념을 사용자의 입장에서 설명한다. 마지막으로, 결론 및 앞으로의 연구 방향에 대해 기술한다.

2. 관련 연구

룸이라는 개념은 Henderson과 Card에 의해 제안된 적이 있었으며, 이들은 실제 세계의 룸들과 유사하게, 사용자의 데스크탑을 다수의 가상 작업 공간으로 나누는 것을 목표로 했다[4]. 그들은 룸을 단순히 한 사용자가 개별적인 단일 사용자용 애플리케이션들을 사용할 때, 자신만의 작업 공간을 용이하게 관리하기 위한 GUI 개념으로 만들었다. 반면에, 본 논문에서 제시된 CBE의 룸의 모델은 한 그룹의 여러 사용자들에게 공유된 맥락의 협업 환경을 제공하기 위한 수단이라고 할 수 있다. 따라서, CBE는 구조적으로 볼 때 각 룸마다 그 안에 속해 있는 사용자들의 애플릿들간의 그룹 통신(group communication)을 기반으로 한 상태의 공유, 작업 공간간에 협업 내용의 이동에 따

른 애플릿들의 그룹 멤버쉽의 변경, 인터넷을 통해 여러 사용자를 지원하기 위한 역할 기반의 접근 제어, 그리고 각 사용자에게 협업 환경에 대한 그룹 인지성(group awareness)을 지원한다는 점이 Henderson의 룸 모델과의 차이점이라고 할 수 있다.

협업 시스템(collaborative system)은 지원되는 협업의 종류(동기적 협업과 비동기적 협업)를 기준으로 분류할 수 있다. 표 1은 관련 협업 시스템들간의 비교를 보여준다.

동기적 협업을 지원하는 협업 시스템의 예로는, Microsoft NetMeeting[5]과 Rendezvous[6]가 있다. 이 시스템들은 기본적으로 여러 사용자의 세션(또는 회의) 관리에 필요한 플로어 제어 정책(floor control policy)을 제공한다. 특히 NetMeeting의 경우, 애플리케이션 공유, 채트, 화이트보드 및 파일 전송 등의 다양한 클라이언트 기능을 지원하고, 스크립트 API나 COM API를 통해 세션을 제어할 수 있는 구조를 제공한다. 그러나, CBE와는 달리 동시에 두 개 이상의 협업 세션을 진행시킬 수 없다. 그리고, Rendezvous의 경우, 기본적인 공유와 관련된 제약조건(constraints)을 추가하는 것이 가능한 구조로 되어 있는 유연성을 지니고 있으나, 진행 중인 세션에 나중에 온 사용자가 참가하는 것을 허용하지 않는 단점을 지니고 있다. 그리고, 근본적으로 이러한 NetMeeting과 Rendezvous와 같은 협업 시스템들은 동기적인 세션만을 구축할 수 있는 메커니즘을 제공하는 반면, CBE는 비동기적인 협업까지도 지원할 수 있는 점이 다르다. 구조적으로 볼 때, 이러한 시스템에서는 세션 관리를 위한 시설은 애플리케이션과 밀접하게 통합되어 있다. 반면에 CBE에서는 세션 관리 서비스가 애플릿과 분리되어 있으므로, 여러 애플릿들에 의해 공유될 수 있다.

TeamRooms[7], Habanero[8], Orbit[9]에서는 애플릿들을 각각 팀 룸(team room), 세션(session), 로케일(locale)이라는 개념들을 통해 다중 애플릿, 사용자, 데이타 객체를 가진 공유 작업 공간을 지원함으로써 동기적 뿐만 아니라 비동기적인 협업이 가능하며 이는 CBE의 룸의 개념과 유사하다. TeamRooms는 구조상 서버가 메시지 전달의 역할을 하면서 관리 클라이언트를 비롯한 사용자 클라이언트들에게 상태 동기화 및 상태 저장을 제공한다. 그리고 Habanero의 경우 개발자들을 위한 API를 제공함으로써, CBE와 같이 확장성 있는 구조를 지원하고 있다. Orbit의 경우, 사용자의 세션 관리 및 로케일(locale : 구성원, 그룹 뷰 등)의 정의 저장을 위한, 분산 Smalltalk로 구현된 로케일 서비스라는 서버를 통해 클라이언트를 이용한 협업을 지원한다. 그러나, 이러한 시스템들은 앞에서 설명한 동기적인 협업만을 제공하는 시스템들과 아울러, 객체 및 도구의 협업 환경으로의 동적인 추가, 그리고 협업 내용의 개인 및 공유 작업 공간 사이의 이동 등이 지원되는 않는다는 점이 CBE와의 기본적인 차이점이라고 할 수 있다. 클라이언트 구조는, TeamRooms는 Tcl/Tk를 사용하며, Habanero는 Java 애플리케이션으로 구현되어 있으므로, 사용 전에 사용자의 플랫폼에 설치가 선행되어야 하는데 반해, CBE와 Orbit의 경우는 인터넷 상에서의 협업 환경으로의 쉬운 접근을 위해, Java 애플릿으로 구현되었으므로, 클라이언트를 설치할 필요 없이 다양한 플랫폼 상에서 웹 브라우저를 통해 쉽게 협업 환경에 접근할 수 있다.

사용자 인터페이스 면에서 볼 때, TeamRooms와 같은 시스템에서는, 룸 안에 있는 모든 애플릿들이 하나의

표 1 관련 협업 시스템들

이름	NetMeeting	Rendezvous	TeamRooms	Habanero	Orbit	CBE
협업의 종류	동기적	동기적	동기적+ 비동기적	동기적+ 비동기적	동기적+ 비동기적	동기적+ 비동기적
협업의 단위	회의	회의	팀룸	세션	로케일	룸
구현	MS- Windows COM	X Window +MEL(Extension of Lisp)	Tcl/Tk	Java	분산 Smalltalk	Java
웹을 통한 접근 및 이용	부분적 지원	지원하지 않음	지원하지 않음	지원	지원	지원
협업내용의 지속성 (persistence)	지원하지 않음	지원하지 않음	지원	지원	지원	지원
협업내용의 동적인 이동	지원하지 않음	지원하지 않음	지원하지 않음	지원하지 않음	지원하지 않음	지원
사용자 역할 기반의 접근 제어	지원하지 않음	지원하지 않음	지원하지 않음	지원	지원하지 않음	지원

큰 윈도우 안에 표시되기 때문에, 고유의 윈도우 관리 기능들이 포함된 GUI를 제공한다. 반면에, CBE에서는 룸 안의 애플릿들이 세션 매니저에 따라서 다르게 디스플레이 될 수 있다. 즉, 룸 안의 애플릿 들을 어떤 형태로 표시하느냐는 세션 매니저의 정책 결정(policy decision) 부분이므로 CBE는 룸 안의 애플릿 들을 하나의 윈도우 속에 제한하지 않는다. 이러한 점은, 사용자들이 표준 혹은 자신에게 익숙한 윈도우 매니저를 사용하여 데스크탑 상에 애플릿을 배치할 수 있도록 해주는 유연성을 제공한다.

3. 설계 개념

공유 작업공간은 다음의 주요 구성요소로 이루어져 있다. 즉, 애플릿, 애플릿 그룹, URL, 룸, 그리고 사용자가 그 구성요소들이다.

3.1 애플릿

애플릿이란 사용자가 데이타를 보거나 업데이트할 수 있게 하는 전형적인 GUI 인터페이스를 제공하는 작은 애플리케이션이다. 사용자는 보통 여러 개의 애플릿을 사용하여 다른 사용자들과 협업을 하게 된다. 이러한 애플릿은 사용자의 데스크탑 상의 프로세스나 쓰레드(thread)로 구현될 수 있다. 예를 들어, 사용자들이 공유 화이트보드와 다중사용자 채트를 이용해서 다른 사용자와 협업하려고 할 경우, 공유 화이트보드 애플릿과 다중 사용자 채트 애플릿이, 협업에 참가하는 각 사용자의 데스크탑 상에 생성될 것이다. 본 논문에서 사용되는 애플릿이라는 용어는 Java에서의 애플릿보다는 광범위한 의미를 지니고 있다. 예를 들면, 본 논문에 제시된 모델에서 애플릿은 Java 애플릿뿐만 아니라 Java 애플리케이션으로도 구현 가능하다.

3.2 애플릿 그룹

애플릿 그룹은 공유 작업공간인 룸에서 사용되는 같은 종류의 애플릿들의 집합을 말한다. 각 사용자의 데스크탑 상의 공유 화이트보드 애플릿들은, 애플릿 그룹을 형성하여 디스플레이나 데이타가 동기화 된다. 구현의 관점에서 볼 때, 각 애플릿 그룹의 구성원은 상태 조정(state coordination)을 목적으로, 같은 멀티캐스트 통신 그룹(multicast communication group)에 가입을 하며, 이 통신 그룹의 개념은 Corona[10]에 의해 제공되는 것과 유사하다.

3.3 URL

URL(Uniform Resource Locator)은 룸에 놓여질 수 있으며, 놓여진 룸에 있는 사용자들에 의해 접근이 가능하다. 따라서, URL을 통해 웹 상의 파일 등과 같은 객체에 대한 참조를 공유할 수 있다.

3.4 룸(Room)

룸은 URL이나 애플릿 그룹 등을 포함할 수 있다. 룸은 사용자들이 협업 세션(collaborative session)을 만들 수 있도록 공유 데이타 객체나 도구(tool) 등을 제공한다. 또한, 룸은 룸 안에 들어오는 사용자들에게, 협업 인지(awareness)[11, 12]를 제공함으로써, 룸 안의 애플릿, URL 그리고 다른 사용자들의 존재를 알 수 있게 해 준다. 본 논문에서는 룸이라는 것의 개념을 공유 작업공간에서 애플릿들과 다른 객체들을 식별하기 위한 상위 수준의 그룹화 메커니즘으로서 사용한다. 이러한 개념은 [4]에서 기술된 특정한 사용자 인터페이스 은유(metaphor)의 사용을 내포하지는 않는다. 그리고, 룸의 애플릿들은 사용자의 데스크탑 상의 어느 곳에도 배치될 수 있다. 룸의 다른 표시방법 중의 하나는, 룸을, HTML페이지에서의 URL로 나타내는 것이다. 원칙적으로 본 모델에서는 룸의 개념을 어떻게 사용자들에게 표현하는 지에 관해서는 특정 협업 환경의 사용자 인터페이스 디자이너에게 맡긴다.

공유 작업공간은 다수의 룸으로 구성될 수 있다. 한 사용자가 여러 개의 룸을 동시에 들어갈 수 있다는 점에서 룸은 가상적인 개념이라고 할 수 있다. 룸에 관련된 서비스들은 사용자가 룸에 들어가기 위해서는 인증을 거치도록 할 수 있다.

그림 1은 룸의 예를 보여준다. 이 예에서는, 두 사용자 A와 B가 룸에서 협업을 하고 있다. 이 공유작업공간에서는 두 개의 애플릿 그룹(공유 화이트보드와 다중 사용자 채트)이 사용되고 있다. 이제 사용자 T가 이 룸에 들어가면, 이 두 개의 애플릿 그룹은 T에게 사용 가능해지게 된다. 즉, 이 두 개의 애플릿 그룹에 해당하는 애플릿들이 T의 데스크탑 상에 생성되어 각각 해당 애플릿 그룹에 가입함으로써, 동기적으로 그룹 내에서의 도구와 데이타를 공유하게 된다. 만약 룸이라는 장치가 없었다면, 사용자 T는 공유 작업공간을 일일이 수동으로 구성해야만 할 것이다.

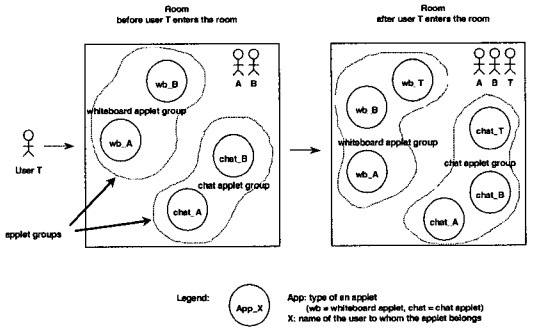

그림 1 협업 환경에서의 룸(room)

다음에는 룸들이 사용자, 애플릿 그룹 및 다른 룸들과 어떤 관계를 가지는지 설명한다.

3.5 애플릿과 룸의 관계

룸에는 애플릿 그룹이 필요한 개수만큼 포함되는 것이 허용된다. 그러나, 하나의 애플릿 그룹은 정확히 오직 한 개의 룸에만 속할 수 있다. 이것은 협업을 제한적으로 만들지는 않는다. 왜냐하면, 사용자는 동시에 여러 개의 룸에 존재할 수 있기 때문이다. 따라서, 여러 사용자들이 동시에 같은 애플릿을 보고 싶은 경우에는, 그 애플릿을, 새롭게 만든 공동의 룸으로 옮기거나, 그 애플릿을 제공하는 룸에 들어가면 된다.

다음의 연산들은 애플릿과 룸간의 결합(binding)을 바꾸기 위해 룸과 애플릿에 수행할 수 있다.

- 룸에 애플릿 넣기: 사용자가 룸에 애플릿을 만들거나 넣으면, 새로운 애플릿 그룹이 만들어지고, 애플릿은 자동적으로 그 그룹에 가입하게 된다. 이 애플릿 그룹에는 룸에 있는 사용자들이 가입할 수 있게 된다.
- 룸에서 애플릿 그룹에 해당하는 애플릿 생성하기: 이 연산은 애플릿 그룹에 속해 있는 애플릿들과 같은 유형의 애플릿을 사용자의 작업공간에 만든다. 생성된 애플릿이 애플릿 그룹에 가입하면, 그 애플릿 그룹에 속해 있는 다른 애플릿들과 상태 및 디스플레이가 동기화 된다.
- 애플릿 삭제하기 : 이 연산은 사용자의 작업공간으로부터 룸에 있는 애플릿을 제거한다. 제거된 애플릿과 같은 애플릿 그룹에 속해 있는, 다른 사용자들의 애플릿들은 계속해서 존재하게 된다.
- 애플릿 그룹 삭제하기 : 이 연산은 애플릿 그룹의 애플릿 구성원들이 속해 있는 통신 그룹(communication group)을 삭제한다. 애플릿 그룹에 속해 있던 애플릿들은 그들이 더 이상 그 그룹에 속해 있지 않다는 통지(notification)를 받아서, 보통 제거되거나 비공유 애플릿이 된다. 예를 들며, 공유 화이트보드 애플릿 그룹이 룸으로부터 제거되면, 그 그룹에 속해 있던 모든 공유 화이트보드 애플릿들은 제거되거나, 자기가 속해 있는 사용자의 비공유 애플릿으로 남는다.
- 애플릿을 하나의 룸에서 다른 룸으로 옮기기 : 이 연산의 결과로 애플릿은 자기가 속해 있던 애플릿 그룹을 떠나서 목적지에 해당하는 룸으로 넣어진다. 그리하여, 새로운 애플릿 그룹이 목적지 룸에 생성된다.
- 애플릿 그룹을 하나의 룸에서 다른 룸으로 옮기기 : 이 연산은 애플릿 그룹을 목적지 룸에 생성해서 애플릿들을 새 애플릿 그룹에 속하게 하고, 원래 룸에 있었던 애플릿 그룹은 삭제한다.

위의 연산들 중에서 가장 중요한 연산인 애플릿 생성 연산의 알고리즘과 애플릿 삭제 연산의 알고리즘을 보면 다음과 같다(여기서 언급되는 세션 매니저, 룸 매니저, 그룹 통신 서버에 관해서는 본 논문의 설계 및 구현 부분에서 좀 더 자세히 설명하기로 한다).

먼저 구체적인 애플릿 생성 알고리즘은 다음과 같다. 우선 사용자의 세션 매니저가 애플릿을 사용자의 데스크탑 상에서 실행을 시킨다. 그 다음 실행이 시작된 애플릿으로부터 생성완료를 알리는 메시지를 받게되면 세션 매니저는 애플릿에게 해당 애플릿 그룹에 가입하라는 메시지를 보낸다. 이 메시지를 받은 애플릿은 그룹 통신 서버에게 해당 애플릿 그룹 가입 요청 메시지를 보내고, 그룹 통신 서버는 애플릿 그룹 목록에 해당 애플릿을 추가하고, 그 결과를 애플릿에게 보낸다. 애플릿은 애플릿 그룹에 가입된 결과를 세션 매니저에게 알린다. 그 후 세션 매니저는 룸 매니저에게 새로운 애플릿에 대한 엔트리를 룸에 넣어 달라는 요청 메시지를 보낸다. 이를 받은 룸 매니저는 룸의 자료 구조에 해당 애플릿에 대한 엔트리를 추가하고 그 결과를 세션 매니저에게 보내게 된다.

다음으로, 구체적인 애플릿 삭제 알고리즘은 다음과 같다. 먼저 사용자의 세션 매니저가 애플릿에게 애플릿 그룹으로부터 탈퇴하라는 메시지를 보낸다. 이 메시지를 받은 애플릿이 그룹 통신 서버에게 애플릿 그룹 탈퇴를 요청하는 메시지를 보내면, 그룹 통신 서버는 자신의 애플릿 그룹 목록에서 해당 애플릿을 삭제하고 그 결과를 애플릿에게 보낸다. 애플릿은 탈퇴 결과를 세션 매니저에게 보낸다. 이 메시지를 받은 세션 매니저는 애플릿에게 종료 요청을 보내어 애플릿을 종료시키고, 룸 매니저에게 애플릿 엔트리를 룸에서 삭제해 달라는 요청을 보낸다. 이 메시지를 받은 룸 매니저는 애플릿을 해당 룸 자료 구조로부터 삭제하고 그 결과를 세션 매니저에게 보내게 된다.

3.6 룸과 룸의 관계

현재 룸 안에 다른 룸을 포함(nesting)시키는 것은 허용되지 않는다. 그러나, 한 룸에서 다른 룸으로의 이동(navigation)을 용이하게 하기 위해, 본 모델에서는 다른 룸으로의 링크를 쉽게 구현하는 것을 가능하게 한다. 링크는, 다른 룸으로의 참조를 관리하는 링크 애플릿이나 URL을 통해서 제공될 수 있다. 지금까지는 협업환경의 사용자들이 룸을 다른 룸에 포함시키는 것에 대한 필요를 못 느끼고 있지만, 앞으로 그러한 필요가 나타날 경우에 그러한 제공을 고려할 수 있다. 그렇게 되는 경우, AFS(Andrew File System)에서와 같은 파일 시스템에서의 디렉토리의 구조를 모델로 삼는 것이 가능하다.

3.7 사용자와 룸의 관계

한 사용자는 동시에 여러 개의 룸 안에 존재하는 것이 허용된다. 왜냐하면, 사용자를 하나의 룸에만 존재하도록 제한할 이없을뿐더러, 오히려 사용자를 여러 룸에 존재하는 것을 허용함으로써, 사용자에게 여러 가지 잠재적인 이익(예를 들어 시간을 더욱 효율적으로 사용할 수 있으며, 협업의 결과를 하나의 룸에서 다른 룸으로 옮길 수 있는 것 등)을 제공할 수 있기 때문이다.

그림 2는 사용자, 애플릿, 룸의 관계를 나타낸다. 그림 2에서 User 1은 3개의 애플릿(세션매니저, 다중사용자 채트, 공유 화이트보드)을 가지고 있다. User 1은 User 2와 같이 Room A안에서 자신들의 채트와 화이트보드 애플릿을 사용하여 서로 협업을 하고 있음을 알 수 있다. 룸과 애플릿의 관계라는 관점에서 보면, 같은 룸 안에는 각 사용자에게 속한 채트 애플릿들로 이루어진, 채트 애플릿 그룹이 존재하며, 이 그룹에 속한 채트 애플릿 끼리는 서로 데이타를 주고 받으며 상태를 공유하는 관계가 된다(화이트보드 애플릿의 경우에도 같은 관계가 성립한다). 그림 2에서 User 2는 User 1과 Room A안에서 채트 및 화이트보드 애플릿을 통해 협업을 함과 동시에, User 3과 함께 Room B안에서 채트 애플릿을 사용하여 협업을 하고 있다. 이는 한 사용자(여기에서는 User 2)가 동시에 여러 개의 룸 안에 존재하는 것이 허용되는 룸과 사용자의 관계를 보여준다. 그리고, Room A와 Room B는 수평적인 관계를 가진다.

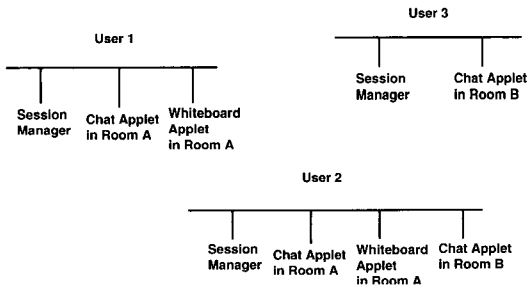

그림 2 사용자, 애플릿, 룸의 관계

각 사용자는 개인 룸을 이용하여, 다른 사용자들과 공유하지 않는 애플릿들을 보관할 수 있다.

사용자가 룸에 들어가면, 그 룸에 있는 애플릿 그룹들은 그 사용자의 세션 매니저(Session Manager)에게 접근 될 수 있게 된다. 그러한 각 애플릿 그룹에 대해 세션 매니저는 적절한 유형의 애플릿을 생성해서, 애플릿에게 애플릿 그룹을 가입하도록 요청을 보내어서, 그 사용자에게 그 룸의 공유된 맥락을 얻게 만든다. 사용자가 룸을 떠나면, 그 룸 안에서 그 사용자가 사용하던 애플릿들은 자신들의 애플릿 그룹을 떠나게 된다.

사용자들의 룸에 대한 접근을 제어하려면, 사용자들은 그 룸에 대한 역할을 할당받게 되는데, 자세한 내용은 다음에 설명한다.

3.8 룸에서의 사용자의 역할(Role)

협업 시스템에서의 접근제어(access control)는 사용자의 상호간섭을 효과적으로 제어함으로써, 사용자의 실수나 고의적인 공격으로부터 그룹 작업을 안전하게 보호하고 시스템의 한정된 자원을 일부 특권층들에게만 사용 가능하게 해주는 중요한 역할을 한다[13]. 그 예로서 Quilt 그룹 에디팅 시스템[14]에서의 사용자는 각 문서에 대해 특정한 사용자 역할(role)을 지니며, 한 범주의 사용 특권(privilege)이 각 사용자 역할에 연관되어 있다. 그 사용자 역할의 종류로는 쓰는 자(writer: 일의 내용을 바꾸도록 허용된 자), 읽는 자(reader: 문서의 내용을 변경할 수 없고 읽기만 허용된 자), 그리고 주석 하는 자(commentator: 문서에 주석만 붙일 수 읽도록 허용된 자)가 있다. Suite 시스템[13]도 마찬가지로 객체에 대한 접근 제어 권한의 집합을 하나의 사용자 역할에 할당하고 있다.

CBE는 협업이 이루어지는 작업 공간 즉 룸에 대한 접근제어를 제공하며, 이를 위해 다음과 같은 종류의 사용자 역할을 지원한다.

- 관리자(ADMIN) : 룸의 접근 제어를 변경할 수 있고, 룸 안에서 애플릿 그룹들을 생성하거나 제거할 수 있으며, 룸 안에 있는 애플릿들과 상호작용을 가질 수 있다.
- 구성원(MEMBER) : 접근 제어의 변경 권한을 제외하고는, 관리자의 모든 권한(right)을 가질 수 있다.
- 관찰자(OBSERVER) : 룸에 있는 애플릿들의 상태를 관찰하는 것만 허용되며, 어떠한 방법으로도 애플릿들의 상태를 바꿀 수 없다.
- 입장 제한자(RESTRICTED) : 룸에 들어가도록 허용되지 않는다.

각 룸은 자기 고유의 접근제어목록(ACL: Access Control List)과 연관되어 있다. 위에서 언급된 역할에 상응하는 네 수준의 특권(privilege)이 존재한다. 표 2는 각 특권 수준에서 허용된 연산들을 보여준다.

표 2에서, 특권 수준은 관리자, 구성원, 관찰자, 입장 제한자로 갈수록 낮아지며, 각 특권 수준에 허용된 연산을 자세히 보면 다음과 같다. 한 사용자가 자신의 세션 매니저를 통해 룸을 생성하게 되면, 그 사용자는 생성된 룸의 관리자 역할을 맡게 되며, 그 룸은 항상 최소한 하나의 관리하는 사용자를 갖게 되는데, 관리자가 이 역할을 나타낸

다. 관리자는 앞서 말한 바와 같이 룸, 애플릿, 접근권한에 관련된 일체의 권한을 가진다. 그 다음으로 높은 특권 수준인 구성원은 "Destroy a room" 연산을 통해 룸을 제거하거나, "Set room ACLs" 연산을 통해 룸에 관한 접근제어를 변경하는 것이 허용되지 않으며, 이러한 연산들까지 수행할 수 있는 사용자는 관리자뿐이다[여기서 ACL은 (특권수준, 사용자목록)의 쌍으로 되어 있다]. 구성원 다음의 특권 수준인 관찰자는 구성원에게 허용된 몇 가지 연산을 수행할 수 없도록 되어 있다. 예를 들면, "Add applet" 연산을 통하여 애플릿을 새롭게 생성된 애플릿 그룹에 가입시키는 것이 허용되지 않으며, "Move applet- group" 연산으로 기존의 애플릿 그룹을 다른 룸으로 옮기거나, "Delete applet-group" 연산을 통해 애플릿 그룹을 제거하는 것이 허용되지 않는다. 따라서 관찰자는 이와 같은 특권 수준으로는 이미 만들어진 애플릿 그룹에 가입하여 협업 과정을 지켜볼 수는 있지만 애플릿 그룹에 영향을 미치는 행동들은 할 수 없다. 마지막으로 입장 제한자는 "Enter a room" 연산을 통하여 룸을 들어가는 것을 포함한 모든 연산의 수행이 허용되지 않는다.

표 2 각 특권 수준(Privilege Level)에 허용된 연산

연산	특권 수준			
	관리자	구성원	관찰자	입장 제한자
Enter a room	허용	허용	허용	허용 않음
Look up a room	허용	허용	허용	허용 않음
Leave a room	허용	허용	허용	허용 않음
Destroy a room	허용	허용 않음	허용 않음	허용 않음
Join applet-group	허용	허용	허용	허용 않음
Add applet	허용	허용	허용 않음	허용 않음
Move applet-group	허용	허용	허용 않음	허용 않음
Delete local applet	허용	허용	허용	허용 않음
Delete applet-group	허용	허용	허용 않음	허용 않음
Get room ACLs	허용	허용	허용	허용 않음
Set room ACLs	허용	허용 않음	허용 않음	허용 않음

CBE와 같은 동기적 협업 및 비동기적 협업을 동시에 지원하는 TeamRooms, Orbit, Habanero와 같은 관련 시스템들의 접근 제어는 다음과 같이 되어있다.

우선 TeamRooms에는 사용자의 접근 권한 구분이 없기 때문에, 팀 룸을 만든 자가 일반 사용자에 비해 특권이 없으며, 아무 사용자나 팀 룸에 들어가거나, 팀 룸의 상태를 바꾸거나, 팀 룸을 제거할 수도 있다. Orbit 또한 객체에 대한 접근 제어는 로케일 단위로만 이루어지는 개방된 접근을 허용하기 때문에, 어떤 구성원도 로케일 안의 객체들을 보거나 수정하거나 제거할 수 있다. Habanero는 사용자 역할(role)을 지원한다. 세션을 만든 사용자는 세션 참여가 허용된 자들의 목록과 같은 세션의 특성을 정할 수 있으며, 세션 참여자들은 능동적 참여자, 수동적 관찰자 등과 같은 역할을 지닐 수 있다. 따라서, Habanero의 이러한 사용자 역할 기반의 접근 제어 방식은 CBE의 경우와 유사하다고 할 수 있다.

3.9 애플릿, 애플릿 그룹 그리고 룸의 지속성(Persistence)

협업 환경의 일에 관한 경험에 의하면, 비동기적인 협업(asynchronous collaboration)[3]에 대한 지원이 필수적으로 보인다. 왜냐하면 스케줄링의 문제로 인하여 반드시 모든 참여 예정자가 동시에 협업에 참여할 수는 없기 때문이다. 따라서, 본 논문에서 제시된 모델은 동기적인 협업 세션들간의 협업 상태의 지속을 제공한다.

현재의 구현된 시스템에서, 룸들은 그 안에 사용자들이 없어도 지속적으로 존재한다. 따라서 룸들은, 사용자들간에 공유되는 정보들의 저장소로 이용될 수 있다. 그리고, 룸이 활동적으로 많이 사용되지 않는 경우, 사용자가 지정한 특정 시간이 지나면 자동적으로 삭제되는 유연성도 본 모델은 수용할 수 있다.

비동기적인 협업을 지원하기 위해, 애플릿들은 그들의 상태가 세션들간에 지속될 수 있도록 설계되어야 한다.

본 시스템에서, 사용자의 요청을 하면, 룸 매니저(Room Manager)가 애플릿의 식별자, 애플릿이 속한 애플릿 그룹의 이름, 그리고 애플릿의 상태를 저장하여 보관하게 된다. 그리고, DistView[2]라는 메커니즘을 통하여, 새로운 애플릿이 애플릿 그룹에 들어가게 되면, 그 애플릿 그룹에 속한 애플릿들이 공유하고 있는 상태가 새로 들어오는 애플릿에 의해서도 접근하여 공유되게 된다.

사용자가 아무 사용자도 없는 룸에 들어가게 되면, 그 안에 있는 애플릿 그룹이, 자신의 가장 최근에 저장되었던 상태와 함께 접근 가능하게 된다. 그렇게 되면, 그 룸에 들어온 사용자의 세션 매니저는 그 애플릿 그룹에 속하게 될 애플릿들을 생성하고, 가장 최근에 저장되었던 상태로 초기화시킨다.

4. 설계 및 구현

본 시스템의 클라이언트 및 서버의 개발 환경으로는, Sun사의 Solaris 운영체제가 탑재된 UltraSPARC 워크스테이션 상에서 Java 소프트웨어 개발 도구(Java SDK v1.2)를 이용하였다(서버 수행 환경도 동일하다).

제시된 클라이언트 구조에서는 그림 3과 같이 사용자의 데스크탑 상의 여러 애플릿들을 조정하기 위해 세션 매니저(Session Manager)를 사용한다.

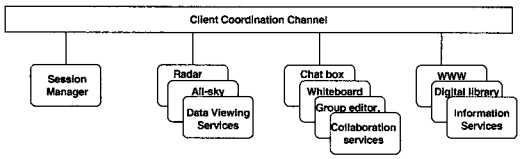

그림 3 클라이언트 구조

세션 매니저는 한 사용자당 하나가 존재한다. 세션 매니저는 다양한 여러 그룹 인지 애플릿에 대한 공통의 인터페이스를 제공하며, 협업 환경의 상태에 관한 질의에 필요한 설비를 갖추고 있다. 그룹 인지 애플릿들은 세션 매니저가 요구하는 인터페이스를 제공하기만 하면, 사용자에 의해 시스템에 추가되는 것을 허용한다. 이 경우 추상 클래스(abstract class)가, 각 애플릿 등이 지원해야 할 메쏘드(method)들을 명기하고 있다. 메쏘드들의 예를들면, 세션 매니저가 애플릿에게 특정한 애플릿 그룹을 가입하거나 탈퇴하라는 요청, 아이콘화 하라는 요청, 종료하라는 요청 등이 있다. 세션 매니저는 애플릿들과 서로 통신하며 애플릿들의 메쏘드가 지원하는 요청을 보낼 수도 있으며, 그룹 인지의 목적으로 협업 환경의 상태에 관한 정보(애플릿이 속해있는 룸에 관한 정보 즉, 룸에 있는 사용자 등)를 애플릿에게 보낼 수도 있다. 애플릿은 이러한 정보들을 이용하여 사용자에게 추가적인 상황을 제공할 수 있다.

사용자는, Java를 지원하는 웹 브라우저(인터넷 익스플로러나 넷스케이프 등)만 있으며, 협업 환경의 월드 와이드 웹(World Wide Web)상의 홈페이지에 접근하여, 세션 매니저를 자신의 플랫폼에 다운로드 받을 수 있다. 세션 매니저는 사용자들을 위한 공유 작업 공간을 구축하기 위해 웹 브라우저를 이용하여, 그룹 인지 Java 애플릿(다중 사용자 채트 애플릿이나 공유 화이트보드 애플릿)을 다운로드 한다.

4.1 실행 시의 통신 구조

그림 4는 협업 환경을 제공하는 시스템의 실행 시의 통신 구조를 보여 준다. 각 사용자의 작업 공간은 한 개 이상의 애플릿들(다중 사용자 채트, 공유 화이트보드 그리고 과학적인 데이타를 보여주는 뷰어 등의, 작은 개개의 애플리케이션)로 구성되어 있다. 다른 사용자들에게 속한, 같은 종류의 애플릿들은, 그룹 통신 서버를 통해 서로 통신을 하며, 그룹에게 공통의 인터페이스를 제공한다.

룸 매니저(Room Manager)와 그룹 통신 서버는 Java 애플리케이션으로 구현되어 있으며, 서버 기계 상에서 실행된다. 세션 매니저는 각 사용자의 기계 상에서 실행되는 Java 애플릿으로 구현된다. 룸 매니저 및 그룹 통신 서버와 같은 기계 상에서 실행되는, HTTP 서버인 httpd를 통해, 사용자들은 Java를 지원하는 웹 브라우저를 이용하여, 웹 상의 협업 환경 홈페이지를 방문하여 세션 매니저를 그들의 데스크탑에 다운로드 받을 수 있다.

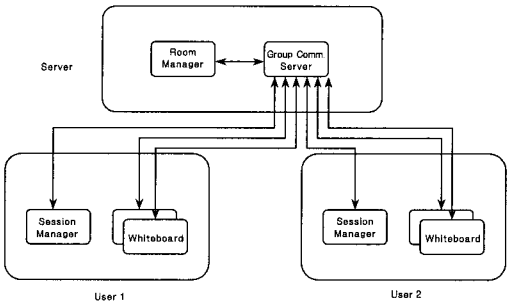

그림 4 협업 제공하는 시스템의 실행시의 통신 구조

다음에 다중 사용자 채트와 같은 그룹 인지 애플릿이 사용자의 데스크탑에서 시작되는 데 필요한 수행 절차를 설명하기로 한다. 먼저, 세션 매니저가 URL을 웹 브라우저에게 주어서, HTTP 서버로부터의 애플릿 다운로드를 요청한다. URL에는 Java 애플릿 클래스의 이름, Java 애플릿의 고유 식별자, 세션 매니저의 고유 식별자, 그리고 그룹 통신 서버의 주소 (포트 번호 포함) 등을 포함한다. 그 다음에는 httpd 서버가 CGI 스크립트를 서버 쪽에서 수행하여 HTML 문서를 생성해내는데, 이 문서에는 애플릿을 실행하기 위해 필요한 여러 인자들(애플릿 식별자, 세션 매니저 식별자, 그룹 통신 서버의 포트)이 포함되어 있다. 이 생성된 HTML 문서는 웹 브라우저에게 전달된다.

이러한, 애플릿에 대한 참조를 포함하고 있는 HTML 문서를 받으면, 웹 브라우저는 HTML 문서를 표시하게 됨으로써, 애플릿이 HTTP 서버로부터 다운로드 된다. 그리하여, 애플릿은 사용자의 기계 상에의 세션 매니저에게 메시지를 보내어 (단, Java 애플릿에 관한 보안 제약 때문에 그룹 통신 서버를 통해서 메시지를 보낸다.) 자신이 가입해야 할 애플릿 그룹 식별자를 얻어서, 그 애플릿 그룹에 가입하게 된다. 이 때, 이 애플릿이 자신이 가입한 애플릿 그룹에 속해 있는 다른 애플릿과 공유 작업 공간을 구성하게 된다.

다음에는, 앞에서 설명한 바와 실행이 시작된 그룹 인지 애플릿들의 그룹 인지 알고리즘에 대해 설명한다.

먼저 룸에 속한 한 애플릿 그룹은 해당하는 통신 그룹이 그룹 통신 서버에 존재한다. 그리고, 애플릿 그룹에 속한 각각의 그룹 인지 애플릿들은 이 통신 그룹의

구성원이 된다. 그리하여, 애플릿 그룹에 속한 한 그룹 인지 애플릿의 상태가 변하게 되면 이는 메시지를 통해 그룹 통신 서버에게 전달된다. 상태의 변화를 전달 받은 그룹 통신 서버는 이러한 사실을 상태 갱신 메시지를 통해서 해당 통신 그룹의 구성원들에게 보내게 된다. 상태 갱신 메시지를 받은 각 그룹 인지 애플릿들은 자신의 상태를 갱신함으로써 사용자들에게 동기화된 디스플레이를 제공할 수 있으므로 항상 그룹 인지성이 유지되게 된다.

4.2 실행 시 통신의 사례

그림 5는, 사용자가 룸에 들어가는 경우의, 애플릿, 세션 매니저 그리고 룸 매니저 사이의 메시지 통신을 보여준다.

사용자가 세션 매니저에게 "Enter a room(룸에 들어가기)"를 요청하면, 이벤트들이 다음의 순서로 일어난다.

1. 세션 매니저가 "Enter a room" 연산을 수행하면, 룸 매니저에게 메시지가 보내진다.
2. 룸 매니저는 룸의 접근 제어(access control)를 체크해서, 그 룸에 대한 사용자의 권한 수준(privilege level)을 검사한다. 만약에 사용자가 그 룸에 들어가는 것이 허용되면, 룸 매니저는 룸에 관한 데이타 구조를 갱신하여 사용자를 룸 안에 있는 상태로 만들고, 이 연산의 결과를 세션 매니저에게 보낸다.
3. 사용자가 룸 안에 들어가면, 세션 매니저는 "Look up a room(룸을 둘러보기)" 연산수행하여, 룸 매니저로부터 룸 안의 사용 가능한 애플릿 그룹들을 가져온다.

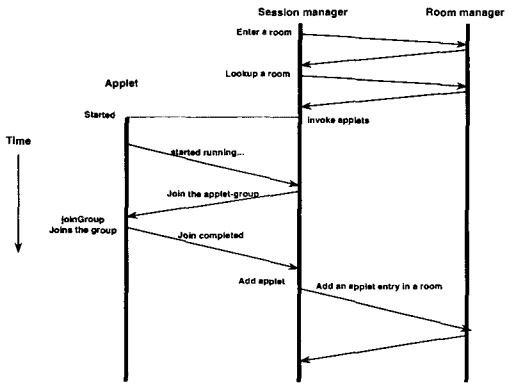

그림 5 사용자가 룸에 들어가는 경우의 애플릿, 세션 매니저, 룸 매니저간의 통신

4. 사용자가 선택한 각각의 애플릿 그룹에 대하여 세션 매니저는 그 애플릿 그룹이 속한 애플릿을 인스턴스화(instantiate)시킨다.
5. 사용자의 입력에 따라 세션 매니저에 의해 생성된 각 애플릿은 세션 매니저에게 수행이 시작되었음을 알리는 메시지를 보낸다.
6. 이 메시지를 받으면, 세션 매니저는 애플릿에게, 어떤 애플릿 그룹을 가입해야 할 지를 알려준다.
7. 애플릿이 이 메시지를 받으면, "joinGroup(그룹 가입하기)" 함수(primitive[10])를 호출하여 해당 애플릿 그룹에 가입하고 그 그룹 식별자를 세션 매니저에게 보내 자기가 애플릿 그룹에 성공적으로 가입했음을 알린다.
8. 이 메시지를 애플릿으로부터 받으면, 세션 매니저는 "Add applet(애플릿 추가하기)" 연산을 수행하여 룸 매니저에게 룸 데이타 구조 중에서 애플릿 그룹에 그 애플릿을 추가 하도록 지시한다.
9. 룸 매니저는 사용자의 룸에 대한 특권 수준을 검사한다. 사용자가 그 룸에 애플릿을 생성하도록 허용되면, 룸 매니저는 룸 자료 구조를 변경하고 룸 뷰(view)에 대한 변화를 모든 세션 매니저에게 멀티캐스트한다. 이러한 메시지를 받은 세션 매니저들은 자신의 룸 뷰를 업데이트한다.

"Leave a room(룸을 나가기)" 연산은 기본적으로 룸 자료 구조의 애플릿 그룹으로부터 애플릿 엔트리를 제거하고 애플릿으로 하여금 애플릿 그룹에게 탈퇴하게 하는 것으로 이루어진다.

제시된 모델의 기본 프로토타입은 Java로 구현되었다.

4.3 접근 제어(Access Control)의 수행

앞에서 설명한 "Enter a room(룸에 들어가기)" 프로토콜을 확장하여, 안전하지 않은 환경에서 접근 제어를 수행하도록 한 프로토콜을 다음에 설명한다. 이 프로토콜에서는 룸 매니저가 모든 사용자로부터 신뢰를 받고 있는 주체(principal) 이라고 가정한다. 이 프로토콜은 공개 키 암호 시스템을 위해 디자인 된 것으로서, 우리는 모든 사용자들이 상대방 및 룸 매니저의 공개키를 안전하게 입수하고, 사용자들과 룸 매니저가 신뢰할 수 있는 암호화 소프트웨어를 갖고 있다고 가정한다. 모든 주체들은 자신의 개인키를 비밀리에 보관한다.

안전한 "Enter a room" 프로토콜은 Needham-Schroeder의 인증 프로토콜[15]의 변형이다. 이 프로토콜은 신뢰를 받는 주체가 (이 경우에는 룸 매니저) 상호 작용하는 한 쌍의 사용자를 위한 키를 할당한다. 우리는 이 프로토콜을 수정하여 서로 다른 접근 권리를 가진 한 그룹의 사용자들이 서로 상호 작용할 수 있도록 하였다. 특권

에 있어서의 다양성 때문에, 개인/공개키 쌍과 같은 비대칭적인 비밀들이, 주체를 인증하는 데 사용된다. 따라서, CBE의 인증 기법은 기존의 공개키 암호화 기법을 기반으로 하여 Needham-Schroeder의 인증 프로토콜을 그대로 응용한 형태라고 할 수 있다. 대칭적인 키를 사용하여 성능을 더 향상시킬 수 있으나, 그렇게 되면 분배 프로토콜이 더욱 복잡해진다.

다음에 4단계로 구성된 "Enter a room" 프로토콜에서의 접근 수행을 설명하기로 한다.

1 단계에서, 세션 매니저는 룸 매니저에게 "Enter a room"이라는 메시지를 보낸다. 이 메시지에는 사용자의 이름, 룸의 이름, 그리고 임의의 숫자(random number)가 포함된다. 사용자의 세션 매니저는 자신의 개인키를 이용하여 개인키 변환과정을 통하여 이 메시지의 디지털 서명을 만든 후, 그것을 룸 매니저의 공개키를 이용하여 암호화한다. 그리하여, 룸 매니저만이 이 메시지를 읽을 수 있게 된다.

2 단계에서, 룸 매니저가 메시지를 해독하여, 메시지의 서명을 확인하고 해당되는 룸의 접근제어목록을 검사한다. 서명이 확인되고 사용자가 룸에 들어가는 것이 허용되면, 룸 매니저는 그 룸의 사용자를 위한 단기의 개인/공개키 쌍을 (각각 개인 룸 키, 공개 사용자-룸 키라고 지칭) 결정한다. 룸 매니저는, 룸 식별자, 키 쌍, 사용자의 임의의 숫자를 포함하는 응답 메시지를 보낸다.

룸 매니저는 그 메시지에 자신의 디지털 서명을 첨부하고 사용자의 공개키로 암호화하여, 그 사용자의 세션 매니저가 그 키의 쌍을 얻을 수 있게 한다. 오직 룸 매니저만이 1 단계의 메시지를 해독하여 그 임의의 수를 얻을 수 있었을 것이므로, 사용자는 자신이 룸 매니저와 교신하고 있다고 안전하게 가정할 수 있다.

3 단계에서는 사용자의 세션 매니저가, 룸을 살펴보는 연산을 수행함으로써 룸의 상태를 요청하는 메시지를 보낸다. 세션 매니저의 메시지는 단순히 사용자의 이름 및 룸의 식별자를 포함한다. 세션 매니저는 자신의 새로운 개인키를 사용하여 만들어진 디지털 서명을 메시지와 함께 룸 매니저에게 보낸다.

4 단계에서는, 룸 매니저가 서명을 검증한다. 서명이 검증되면, 룸 매니저는 사용자를 룸에 들어가도록 허용한다. 사용자의 이름, 역할 그리고 공개 사용자-룸 키를 포함하는 엔트리가 룸 데이타 구조에 추가된다. 다른 사용자들은 이 공개키를 룸 매니저로부터, 룸 매니저에 의해 서명된 메시지를 통해서 받아서, 그 사용자로부터 보내진 서명된 메시지를 검증할 때 사용한다. 공개키를 저장하는데 룸 매니저를 사용함으로써, Needham-Schroeder 프로토콜에 대한 재공격(replay attack)을 피할 수 있게 된다.

4.4 CBE로 구현된 시스템의 사용 예와 평가

그림 6은 CBE를 사용하는 현재의 Java기반 시스템에서의 다중 애플릿 협업 환경의 사용을 보여준다. 세션 매니저(Session Manager)는 사용 가능한 룸들의 목록을 보여준다(그림에서 좌측 상단). 세션 매니저의 인터페이스는 다양한 명령들(사용자가 하나 또는 그 이상의 룸에 참여하기, 룸을 생성하기, 룸을 나가기, 애플릿이나 URL들을 룸에 첨가하기, 애플릿이나 URL들을 동적으로 한 룸에서 다른 룸으로 옮기기 등)을 제공한다.

사용자들은 룸들에 있는 다른 사용자들과 협업하기 위해 그 룸들에 들어갈 수 있다. 사용자들은 들어간 룸에 있는 URL들을 보거나 애플릿들을 사용할 수 있다. URL들은 웹 브라우저를 통해 디스플레이 될 수 있는 임의의 객체들(예를 들면, 파일, 이미지, 일인용 애플릿 등)에 대한 참조(reference)를 가진다. URL에 관한 한, URL에 대한 뷰가 아니라 URL에 대한 참조만이 룸 안에 있는 사용자들간에 공유된다.

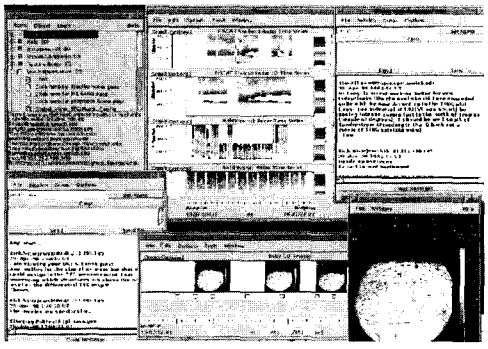

그림 6 CBE를 사용한 다중 애플릿 협업 환경의 사용 예

룸은 동기적인 협업(synchronous collaboration)이 끝났을 때에도 여전히 존재할 수 있다는 의미에서 지속적이라고 할 수 있다. 룸에 들어있는 URL들은 협업 세션간에 저장되어 사용될 수 있다. 또한, 다중 사용자 채트와 같은 애플릿들도 자신의 상태를 룸에 지속적으로 저장해 놓을 수 있다. 그리하여, 나중에 같은 종류의 애플릿인 다중 사용자 채트가 다시 수행되었을 때, 가장 최근에 저장된 상태로 다시 시작될 수 있다. 따라서, 룸은 비동기적인 협업을 지원하는데 사용할 수 있다.

인터페이스는 사용 가능한 애플릿이라든가, 룸에 존재하는 사용자들과 같은, 다양한 그룹 인지 기능을 제공한다. 사용자들은 다른 사용자의 개인 룸에 들어갈 수는

없으나, 메시지를 보낼 수는 있다. 애플릿들은 데스크탑 상의 어느 곳이든지 배치할 수 있다. 애플릿들은 자신이 어느 룸에 속해 있는지를 사용자에게 표시해 준다.

사용자는 공유 화이트보드와 같은 그룹 인지 애플릿을 공유 룸으로부터 개인 룸으로 옮길 수 있다. 이러한 이동은 다른 사용자에게는 영향을 미치지 않는다. 다만, 사용자의 애플릿이 다른 사용자와의 자기 상태 변화에 대한 통신을 중단할 뿐이다. 사용자는 그룹 인지 애플릿을 개인 룸에서 공유 룸으로 옮길 수도 있다. 이와 같은 경우에는, 공유 룸으로 옮겨진 애플릿의 상태가 그 룸의 다른 사용자들에게 보여지며, 그 룸에 나중에 새로 들어온 사람들도 그 애플릿과 상호 작용을 할 수 있다.

CBE는 단지 협업 환경을 구출하는데 필요한 기반(infrastructure)을 제공할 뿐이지, 사용자 인터페이스에 대해 특정한 제약을 가하지는 않으므로, 설계자가 사용자 요구사항에 맞게 사용자 인터페이스를 만드는 것이 용이하다.

이러한 CBE 모델의 응용 분야로는, 이미 예로 제시된 원격 과학 연구 협업 시스템 개발 외에, 원격 회의[16], 원격 진료[17], 원격 교육[18], 그리고 다중사용자 게임[19] 등을 포함한 지리적으로 떨어져 있는 사람들 간의 상호 작용 및 협력을 지원해 주는 시스템의 개발이 포함된다.

위에서 설명한 시스템은 4일 동안 UARC 프로젝트에 관련된 여러 나라의 우주 과학자들에 의해 사용되었다. 총 95명의 사람들이 참여하였으며, 이 중에서 79명(83%)은 우주 과학자들이었고, 나머지는 컴퓨터 공학자들을 포함한 다른 분야의 사람들이었다. 주된 협업 시간은 미국동부시간으로 오전 10시부터 6시까지로 하였다. 실험 기간 동안에 생성된 룸의 갯수는 41개였으며, 이 중에서 주된 협동 장소로 쓰였던 한 룸에는, 39명까지 동시에 참여하였다. 그러나, 상당수(66%)의 룸들의 경우는 5명 이하의 사용자들에 의해 동시 참여되었다. 그림 7은 한 사용자가 한 세션에서 최대 몇 개의 룸에 동시에 참여했는지와 그러한 세션이 몇 개인지를 나타낸다 (그림에서 "active rooms"란, 한 사용자에 의해 동시에 참여된 룸들을 말한다). 여기서, 세션이란 한 사용자가 로그인해서 로그아웃할 때까지의 기간을 말한다.

그림 7에서 보는 바와 같이 가장 흔한 경우가 사용자들이 한 세션 동안에 동시에 두 개의 룸에 참여하는 경우이다(39%, 즉 총 233개의 세션 중에서 90 세션). 사용자들이 한 개 이상의 룸에 동시에 참여하는 것이 81%(총 233개의 세션 중에 189 세션)로서, 오직 하나의 룸에만 참여하는 경우(19%, 44 세션)보다 훨씬 많음을 알 수 있다. 좀 더 자세히 보면, 사용자들이 상당수(67%, 155 세션)의 경우 두 개에서 네 개까지의 룸에서 동시에 참여하고 있는 것을 알 수 있다. 이러한 결과는 한 사용자가 여러 룸에 동시에 참여하는 다중 협업(multiple collaboration)을 지원하는 CBE의 룸 모델이 유용하다는 것을 보여 준다.

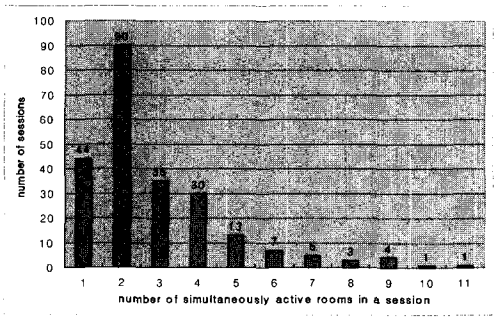

그림 7 한 세션에서 동시에 참여한 룸의 수와 그에 대한 세션 수

5. 결 론

본 논문은 인터넷 상에서의 동적인 협업 환경(collaboration environment)을 지원하는 CBE라고 불리는 소프트웨어 구조의 설계 및 구현을 제시하였다. 제시된 구조는 룸, 사용자, 애플릿, 애플릿 그룹 등의 개념들을 기반으로 하고 있다. 우리는 이러한 개념들의 의미 체계를 제시하고, 그들간의 연관성을 설명하였으며, 룸에 대한 접근 제어 모델을 보였다.

우리는 또한 이러한 상위 수준의 개념들을 기본 구조의 하위 계층으로 사상(mapping)시키는 문제를 연구하였다. 특히, 룸 수준의 그룹화 메커니즘을 통신 수준의 그룹에 관한 연산으로 사상시키는 방법을 설명하였다. 그리고, 그룹 통신 프로토콜에 디지털 서명을 이용함으로써 룸에 대한 접근 제어를 실행하는 방법을 제시하였다.

제시된 모델의 특징은 객체 및 도구의 협업 환경으로의 동적인 추가, 그리고 협업 내용의 개인 및 공유 작업 공간 사이의 이동, 그리고 인터넷 상에서의 접근 제어의 제공이라고 할 수 있다.

다중 사용자 채트, 공유 화이트보드, 그리고 UARC(Upper Atmospheric Research Collaboratory)용 디스플레이 등을 포함한 여러 애플릿들을 갖춘, 제시된 모델의 프로토타입 버전이 Java로 구현되었으며, Java를 지원하는 브라우저로 다운로드를 통해 수행될 수 있다.

큰 규모의 협업 환경이 실용적이 되기 위해서는, 본 논문에서 제시된 모델을 지원하는데 있어서 추가적으로 언급되어야 할 문제들도 있다. 예를 들면, 확장성 있는 통신

프로토콜을 지원함으로써, 많은 수의 관찰자가 지원되며, 룸의 정규 멤버들에 의해 느껴지는 성능이 관찰자들에 의해 느껴지는 성능보다 더 높은 우선 순위를 가질 수 있게 하는 것 등이다. Corona 시스템[4]이 그와 같은 문제를 해결하는데 목적을 두고 있다. 그 외에도, 앞으로 계속될 과제로는, 세션 매니저에 의해 제공되는 사용자 인터페이스의 개선, 그리고 사용자의 역할에 따라 다른 수준의 성능을 제공하며 오류에도 수행될 수 있도록 룸 매니저를 확장하는 일이 있다.

참 고 문 헌

[1] R. Clauer et. al., "UARC: A Prototype Upper Atmospheric Research Collaboratory," EOS Transactions on American Geophysical Union, Vol 74, 1993.

[2] A. Prakash and H. Shim, "DistView:Support for Building Efficient Collaborative Applications Using Replicated Objects," Proceedings of the Fifth Conference on Computer Supported Cooperative Work, October 1994.

[3] N. Preguiça, J. Martins, H. Domingos, and S. Duarte, "Data Management Support for Asynchronous Groupware," Proceedings of the Eighth Conference on Computer Supported Cooperative Work, pp. 69-78, December 2000.

[4] D. Henderson and S. Card, "Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface," ACM Transactions on Graph, Vol. 5, No. 3, pp. 211-243, July 1986.

[5] Microsoft Windows NetMeeting, http://www.microsoft.com/netmeeting, 2000.

[6] J. Patterson, R. Hill, S. Rohall, and W. Meeks, "Rendezvous: An Architecture for Synchronous Multi-User Applications," Proceedings of the Third Conference on Computer Supported Cooperative Work, pp. 317-328, October 1990.

[7] M. Roseman and S. Greenberg, "TeamRooms: Network Places for Collaboration," Proceedings of the Sixth Conference on Computer Supported Cooperative Work, pp. 325-333, November 1996.

[8] L. Jackson and E. Grossman, "Integration of synchronous and asynchronous collaboration activities," ACM Computing Surveys, 31, June 1999

[9] T. Mansfield, S. Kaplan, G. Fitzpatrick, T. Phelps, M. Fitzpatrick, and R. Taylor, "Toward locales: Supporting collaboration with Orbit," Journal on Information and Software Technology, Vol. 41, No. 6, pp. 367-382, April 1999.

[10] R. Hall, A. Mathur, F. Jahanian, A. Prakash, and C. Rasmussen, "Corona: A Communication Service for Scalable, Reliable Group Collaboration Systems," Proceedings of the Sixth Conference on Computer Supported Cooperative Work, pp. 140-149, November 1996.

[11] M. Boyle, C. Edwards, and S. Greenberg, "The Effects of Filtered Video on Awareness and Privacy," Proceedings of the Eighth Conference on Computer Supported Cooperative Work, pp. 1-10, December 2000.

[12] S. Teasley, L. Covi, M. Krishnan, and J. Olson, "How Does Radical Collocation Help a Team Succeed?," Proceedings of the Eighth Conference on Computer Supported Cooperative Work, pp. 339-346, December 2000.

[13] P. Dewan and H. Shen, "Controlling Access in Multiuser Interfaces," ACM Transactions on Computer-Human Interaction, Vol. 5, No, 1, pp. 34-62, March 1998.

[14] M. Leland, R. Fish, and R. Kraut, "Collaborative Document Production Using Quilt," Proceedings of the Second Conference on Computer-Supported Cooperative Work, pp. 206-215, 1988.

[15] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks," Communications of ACM, Vol. 21, No. 12, pp. 993-999, December 1978.

[16] M. Chen, "Leveraging the Asymmetric Sensitivity of Eye Contact for Videoconference," Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 49-56, 2002.

[17] W. Raghupathi and J. Tan, "Strategic IT Applications in Health Care," Communications of the ACM, Vol. 45, No.12, pp. 56-61, December 2002.

[18] J. Cadiz et. al., "Distance Learning Through Distributed Collaborative Video Viewing," Proceedings of the Eighth Conference on Computer Supported Cooperative Work," pp. 135-144, December 2000.

[19] M. Bylund and F. Espinoza, "Testing and demonstrating context-aware services with Quake III Arena," Communications of ACM, Vol. 45, No. 1, pp. 46-48, January 2002.

이 장 호

1990년 서울대학교 컴퓨터공학과 학사
1992년 서울대학교 컴퓨터공학과 석사
2000년 미국 University of Michigan 컴퓨터공학과 박사. 2000년 미국 IBM Watson 연구소 Postdoctoral Researcher
현재 홍익대학교 컴퓨터공학과 전임강사
관심분야는 CSCW(Computer Supported Cooperative Work), 분산 시스템, 모바일 컴퓨팅