Journal of the Industrial Technology Research Institute.
Korea Maritime University. Vol. 13. 1995.

분산 환경에서 공동 작업을 위한 응용 공유 서버의 설계 및 구현

임 재 홍 *

Design and Implementation of an Application Sharing Server for Cooperative Works in a Distributed Environment

Jae-Hong Yim

Abstract

This paper proposes an application sharing server which supports robustness, extensibility, data consistency, easy floor control and transparency for cooperative works in heterogeneous distributed environment.

The application sharing server is composed of two modules, the event converting module on each system and the event multicasting module on a server. The event multicasting module communicates with the event converting module by UDP(User Datagram Protocol). The event multicasting module cooperates with the group management server to obtain floor information and a group member list.

The application sharing server is implemented and tested over MS-Windows 3.1/PC and X-Window/UNIX workstation. The result proved the application sharing server to be excellent.

1. 서 론

컴퓨터의 전형적인 응용 형태는 단일 사용자용으로써, 워드프로세서, 스프레드쉬트, 드로잉 툴 등이 개인의 작업을 지원하기 위하여 다양하게 개발되었으나, 일반 조직의 작업 대부분이 조직원 서로간의 협동체제로 처리되고 있다는 사실은 거의 고려되지 않았다. 그러나 정보통신 기술의 발달로 PC 및 워크스테이션들간의 통신망 구축에 의한 다양한 자원들의 효율적 이용, 기능 분산 및 부하 분산 등의 부가 이익과 함께 공동 작업의 가능성에 대한 인식이 높아지기 시작하였다. 이러한

* 한국해양대학교 이공대학 전자통신공학과

가능성은 그룹웨어(groupware) 또는 CSCW(Computer Supported Cooperative Work)라 불리우는 새로운 응용 분야로 발전하게 되었으며, 회의 시스템, 공동 편집 시스템, 의사 결정 시스템 등 연구 개발이 활발히 진행되고 있다[1, 2, 4, 5, 7].

현재 많은 종류의 그룹웨어 응용 프로그램이 개발되어 있지만, 대부분 동일한 운영체제를 사용하는 환경에서만 사용가능하며, 여러 종류의 컴퓨터로 구성되는 이기종 분산 환경에서는 사용이 불가능한 실정이다. 따라서 개방 분산 환경에서 다양한 응용을 그룹웨어로 지원하기 위하여 응용 프로그램과 운영체제 양면의 특징을 가지고 있는 그룹웨어 플랫폼에 대한 개발도 진행중이다. 그룹웨어 플랫폼은 기본적으로 그룹 관리 기능, 응용 프로그램 공유 기능, 멀티캐스트 기능 등을 제공하여야 한다[4].

본 논문에서는 이기종 분산 환경에서 다양한 그룹웨어 응용 프로그램을 개발, 운용할 수 있는 그룹웨어 플랫폼의 주요 구성요소인 응용 공유 서버의 설계 및 구현에 관하여 논한다. 응용 공유 서버는 공동 작업에 참여하는 모든 사용자들이 단일한 응용 프로그램을 공유할 수 있는 환경을 제공한다. 본 논문에서의 응용 공유 서버는 이벤트 변환 모듈(event converting module)과 이벤트 멀티캐스팅 모듈(event multicasting module)로 구성되며 이들간의 통신은 UDP(User Datagram Protocol)를 통하여 이루어진다. 이벤트 변환 모듈은 이벤트 변환 기능과 발언권 제어 기능을 수행하고, 이벤트 멀티캐스팅 모듈은 이벤트 멀티캐스팅 기능과 그룹 관리 서버와의 연동 기능을 담당한다. 구현한 응용 공유 서버를 MS-Windows 3.1/PC와 X-Window/UNIX workstation 환경에서 화이트보드(whiteboard) 응용 프로그램을 사용하여 시험하였으며, 좋은 결과를 얻을 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 응용 공유 방식에 대하여, 3장에서는 응용 공유 서버의 설계 및 구현에 대하여 기술한다. 마지막 4장에서는 결론에 대하여 논한다.

2. 응용 공유 방식

응용의 공유 방식은 크게 중앙집중형(centralized) 방식과 분산형(distributed) 방식의 두 가지로 나눌 수 있다[3, 6].

2.1 중앙집중형 방식

중앙집중형 방식은 그림 1과 같이 전체 시스템에서 단지 하나의 응용 프로그램이 실행되며, 사용자 입력은 응용 공유 서버를 통하여 응용 프로그램으로 전달되어 그 처리 결과가 다시 사용자 시스템으로 전달된다. 따라서 각 사용자 시스템은 그래픽 터미날과 윈도우 서버로써 동작한다.

이 방식은 응용 프로그램과 데이타가 하나의 시스템에서 관리되므로 일관성(consistency) 유지가 용이하고, 서버 기반의 윈도우 시스템(X-window)에서 기존의 단일 사용자용 응용 프로그램을 수정없이 사용할 수 있다는 장점이 있으나, 응용 프로그램을 실행하는 시스템 고장시 운영이 불가

능하며, 응용 프로그램의 처리 결과를 모든 사용자에게 멀티캐스팅해야 하기 때문에 네트워크에 걸리는 부하가 많아진다는 단점이 있다. 또한 서로 다른 윈도우 시스템을 사용하는 경우에는 공유가 불가능하여 이기종 환경의 지원에 부적합하다

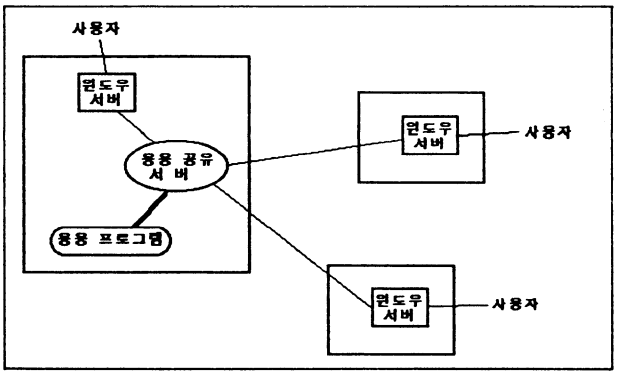

그림 1. 중앙집중형 방식

2.2 분산형 방식

분산형 방식은 그림 2와 같이 모든 시스템에 응용 프로그램이 존재하여 독립적으로 실행되고, 응용 프로그램들간의 이벤트(event) 교환에 의하여 공유가 이루어진다.

따라서 이기종 환경에 적합하며, 사용자 입력만이 멀티캐스팅되고 실행은 각 시스템에서 이루어지므로 네트워크의 부하가 적지만, 데이타의 일관성 유지가 어렵고 응용 프로그램들 사이의 동기 유지가 어렵다는 단점이 있다.

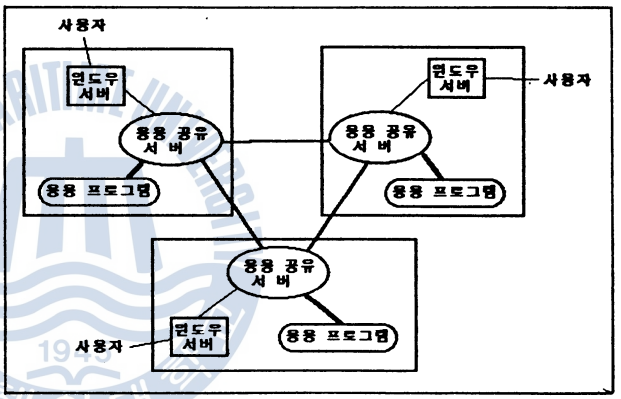

그림 2. 분산형 방식

3. 응용 공유 서버의 설계 및 구현

3.1 응용 공유 서버 모델

본 논문에서 제안하는 응용 공유 서버 모델은 데이타의 일관성 유지와 발언권 제어 등의 문제를 해결하기 위하여 그림 3과 같이 중앙집중형과 분산형을 혼합한 방식을 택하였으며, 이벤트 변환 모듈(event converting module)과 이벤트 멀티캐스팅 모듈(event multicasting module)로 구성된다.

이벤트 변환 모듈은 이벤트 변환 기능과 발언권 제어 기능을 수행하고, 이벤트 멀티캐스팅 모듈은 이벤트 멀티캐스팅 기능과 그룹 관리 서버와의 연동 기능을 담당한다. 이벤트 변환 모듈은 그림에서와 같이 각 시스템마다 존재하며, 이벤트 멀티캐스팅 모듈은 그룹에 하나 존재한다.

3.2 이벤트 변환 모듈

3.2.1 윈도우 시스템

윈도우 시스템이 기본적으로 이벤트 드리븐(event-driver) 방식을 취하고 있다. 즉, 사용자의 입력을 응용 프로그램이 직접 받아서 처리하는 것이 아니라, 윈도우 시스템이 이벤트의 형식을 빌어 입력내용을 응용 프로그램에 전달하면 이를 응용 프로그램이 처리하는 것이다. 이를 그림으로 나타내면 다음 그림 4와 같다.

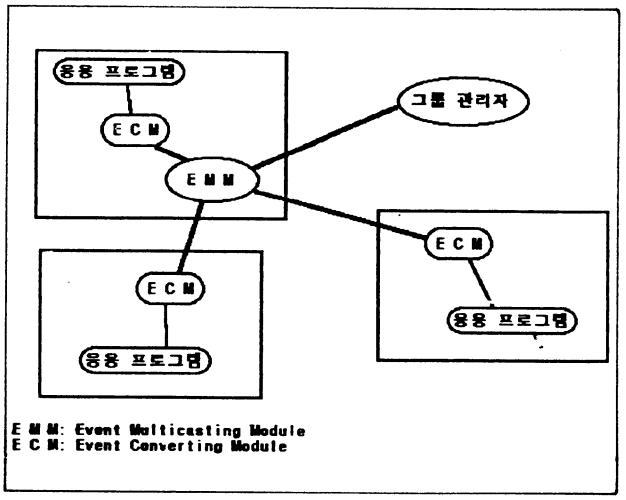

그림 3. 응용 공유 서버 모델

서로 다른 윈도우 시스템상에서 수행되고 있는 응용 프로그램들간에 이벤트를 공유하려 한다면, 한 윈도우 시스템에서 발생한 이벤트를 다른 윈도우 시스템에서는 자신의 응용 프로그램에 맞도록 이벤트를 변환한 후 전달하여야 한다. 이러한 기능을 제공하는 것이 이벤트 변환 모듈이다. 이벤트 변환 모듈의 가장 큰 특징은 어느 한 윈도우 시스템에서 발생한 이벤트를 각각의 서로 다른 윈도우 시스템에 맞는 이벤트로 변환하기 보다는, 본 논문에서 정의한 글로벌(global) 이벤트로 변환하여 다른 시스템에 보내면 각 시스템에서 자신에 맞는 이벤트로 변환하는 것이다. 이러한 경우 새로운 윈도우 시스템이 추가되어도 그 윈도우 시스템의 이벤트와 글로벌 이벤트간의 변환 기능만 추가하면 되므로 확장성 및 유지, 보수가 용이하다는 장점이 있다.

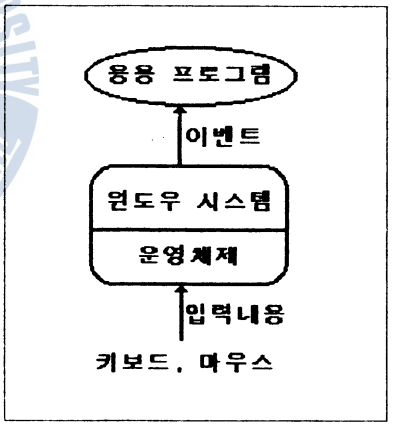

그림 4. 윈도우 시스템

3.2.2 이벤트 변환

본 논문에서는 대표적인 윈도우 시스템인 X-Window와 MS-Windows 3.1에 대하여 글로벌 이벤트를 이용한 이벤트 변환 기능을 구현한다.

(1) X-Window와 MS-Windows 3.1 이벤트 구조의 비교

다음 그림 5와 그림 6은 각각 X-Window9)와 MS-Windows 3.110)의 이벤트 구조를 나타낸다.

X-Window의 경우, XEvent 멤버중 마우스 포인터에 의하여 발생되는 이벤트는 XButtonEvent와 XMotionEvent 두 가지이다. X 서버는 포인터의 상태에 따라 두 이벤트 구조체의 type 필드에 세가지 이벤트 타입을 설정한다. 예를 들어, XButtonEvent의 type 필드에는 버튼이 눌러진 경우에

```
typedef union _XEVENT {
        int                type;
        XAnyEvent          xany;
        XButtonEvent       xbutton;
        XCirculateEvent    xcirculate;
                :
        XMotionEvent       xmotion;
                :
        } XEvent;
```

그림 5. X-Window 이벤트 구조

```
typedef struct tagMSG {
        HWND    hwnd;
        WORD    message;
        WORD    wParam;
        LONG    lParam;
        DWORD   time;
        POINT   pt;
        } MSG;
```

그림 6. MS-Windows 3.1 이벤트 구조

ButtonPress가 설정되고 버튼이 떨어진 경우 ButtonRelease가 설정된다. 포인터가 움직이는 경우에는 XMotionEvent의 type 필드에 MotionNotify가 설정된다. 이외에 이벤트가 발생한 윈도우 ID, 발생시간 및 발생위치 등 10 여가지 정도의 필드가 있어서 X 서버가 클라이언트에게 사용자의 포인터 입력 정보를 주게된다.

MS-Windows 3.1의 경우, message 필드에 발생한 이벤트의 타입이 설정되는데, 예를 들어 왼쪽 마우스 버튼이 눌러지거나 떨어질때 WM_LBUTTONDOWN과 WM_LBUTTONUP이 설정되고, 마우스가 움직이면 WM_MOUSEMOVE가 설정된다.

(2) 이벤트 전달 방법의 비교

이벤트를 윈도우 시스템상의 객체가 다른 객체에게 전달하기 위해서는 그 응용 프로그램의 윈도우 ID를 얻어야 하며, 이벤트의 수신측에서는 자신의 이벤트 버퍼에 쌓인 이벤트가 윈도우 시스템이 보낸 것인지 다른 응용 프로그램이 보낸 것인지를 구별해야 한다.

MS-Windows 3.1에서는 message를 프로그래머가 정의할 수 있도록 되어있다. 따라서 이벤트 변환 모듈이 응용 프로그램에게 혹은 응용 프로그램이 이벤트 변환 모듈에게 보낸 이벤트와 윈도우 시스템에서 이벤트 변환 모듈과 응용 프로그램에게 보낸 이벤트를 구별한다. MS-Windows 3.1에서는 이벤트를 보내는데 두 가지 API(Application Program Interface), SendMessage()와 PostMessage()를 제공한다. SendMessage()는 응용 프로그램에게 직접 이벤트를 보내고, PostMessage()는 수신할 응용 프로그램에게 이벤트를 전달하도록 윈도우 시스템에게 의뢰하는 형식을 취하므로써 수신 응용 프로그램의 부하를 줄이는 효과가 있다.

X-Window에서의 객체간 통신도 MS-Windows 3.1과 유사하지만, 다른 점은 X-Window에서는 응용 프로그램이 받은 이벤트가 다른 응용 프로그램이 보낸 것인지 X-Window가 보낸 것인지 구별할 수 있는 send_event 필드가 이벤트 구조체내에 있다. 따라서 MS-Windows에서처럼 이벤트를 따로 정의할 필요없이 수신측에서 받은 이벤트의 send_event 필드를 조사해서 FALSE이면 X-Window 시스템이 보낸 것이고, TRUE이면 다른 객체가 XSendEvent()라는 X-Window API를 이용하여 보낸 이벤트임을 구별하게 된다.

(3) 글로벌 이벤트를 이용한 이벤트 변환

X-Window와 MS-Windows 3.1의 마우스 이벤트 구조를 비교 분석해 본 결과로부터 많은 유사점이 있음을 알 수 있다. 그 결과로부터 글로벌 이벤트를 정의하고 이벤트 변환 모듈에서 로칼 윈도우 시스템의 이벤트와 글로벌 이벤트와의 변환 기능을 구현한다. 본 논문에서 정의한 글로벌 이벤트의 구조는 다음과 같다.

```
typedef         union {
 int             header;
 typeStruct      typeStruct;
 UButtonEvent    ubutton;
 UMotionEvent    umotion;
 UAnyEvent       uany;
 } UEvent;
```

그림 7. 글로벌 이벤트 구조

```
typedef         UButtonEvent {
        int     header;
        int     type;
        int     user_id;
        int     application_id;
        int     tool;
        int     point_x, point_y;
        int     state;
        int     button;
        } UButtonEvent;
```

그림 8. UButtonEvent

```
typedef         UMotionEvent {
        int     header;
        int     type;
        int     user_id;
        int     application_id;
        int     tool;
        int     point_x, point_y;
        } UMotionEvent;
```

그림 9. UMotionEvent

다음 표 1은 글로벌 이벤트와 X-Window, MS-Windows 3.1 이벤트와의 변환을 위한 관계를 나타낸다.

표 1. 윈도우 시스템 이벤트 변환 관계

	X-Window	MS-Windows 3.1	글로벌 이벤트
버튼 눌림	XButtonEvent (ButtonPress)	WM_LBUTTONDOWN	UButtonEvent (BUTTONDOWN)
버튼 놓임	XButtonEvent (ButtonRelease)	WM_LBUTTONUP	UButtonEvent (BUTTONUP)
마우스 움직임	XMotionEvent (MotionNotify)	WM_MOUSEMOVE	UMotionEvent (MOTIONEVENT)

3.3 이벤트 멀티캐스팅 모듈

이벤트 멀티캐스팅 모듈은 한 윈도우 시스템에서 발생한 이벤트를 모든 이벤트 변환 모듈에 멀티캐스팅 해 주는 기능과 그룹 관리 서버와의 연동 기능을 제공한다. 이벤트 변환 모듈과 이벤트 멀티캐스팅 모듈간 통신은 UDP(User Datagram Protocol)를 사용하며 패킷 단위로 이루어진다. 이벤트 변환 모듈과 응용 프로그램간에는 각 윈도우 시스템에서 지원하는 방식을 이용하여 구현하

였다. 각 모듈간 패킷 흐름을 그림 10에 나타내었다.그림 10에서 PASSWD_PACKET은 사용자가 그룹의 패스워드를 알아야 이벤트 멀티캐스팅 모듈에 접속할 수 있게 된다. 이벤트 변환 모듈로부터 접속 요청을 받았을때 이벤트 멀티캐스팅 모듈은 자신의 데이타베이스를 참조해서 참가자 리스트에 있으면 ACK_CONNECT_FROM_EMM을 보내고 만약 없으면 보내지 않는다. 참가자 리스트에 있는 경우에 한해서 그룹 패스워드를 요청한다. 발언권 정보(FloorInformation) 패킷은 이벤트 멀티캐스팅 모듈이 그룹 관리 서버로부터 받은 패킷을 그대로 멀티캐스팅한다. 패킷의 구조는 그림 11에 나타낸 바와 같다.

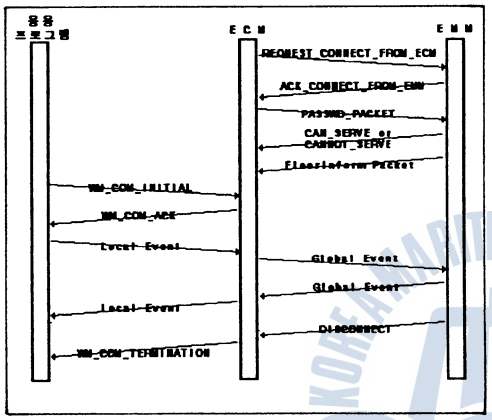

그림 10. 모듈간 패킷 흐름

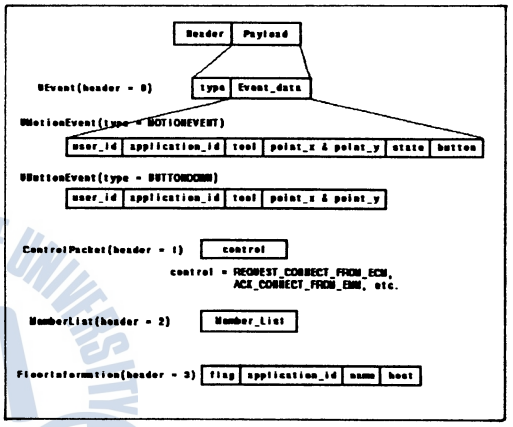

그림 11. 패킷 구조

3.4 고 찰

응용 공유 서버를 MS-Windows 3.1/PC와 X-Window/UNIX workstation 환경에서 구현하여 그룹웨어의 기본적인 도구인 화이트보드를 이용하여 시험하였으며, 좋은 결과를 얻을 수 있었다.

그림 12와 그림 13은 각각 UNIX와 PC Window 3.1 환경에서의 사용자 인터페이스를 나타낸다.

제안된 응용 공유 서버의 장점은 다음과 같이 요약된다.

i) 시스템의 안정성: 기본적으로 분산형 모델을 지향하므로 전체 시스템이 안정적이다. 즉, 하나의 시스템에서 이벤트 변환 및 네트워크 부하가 편중되는 것을 방지함으로써 전체 시스템의 기능 저하를 예방할 수 있다.

ii) 발언권 제어의 용이함 및 네트워크 부하의 감소: 각 이벤트 변환 모듈에서 이벤트 멀티캐스팅 모듈로부터 받은 발언권 정보는 동일하므로 발언권 제어가 일률적으로 이루어지며, 발언권이 없는 회의 참가자의 입력이 원천적으로 봉쇄되어 네트워크에 불필요한 부하를 가중시키지 않는다.

iii) 확장성 및 유지, 보수의 용이함: MS-Windows 3.1, X-Window 뿐만 아니라 OS/2 Window, Windows 95 등의 다른 윈도우 시스템으로 개방 환경을 확장하고자 할 때, 전체 응용 공유 서버를 수정할 필요없이 이벤트 변환 모듈 부분만 확장하면 되므로 확장성 및 유지, 보수가 용이하다.

iv) 공유 데이타의 일관성 유지: 이벤트 멀티캐스팅 모듈에서는 각 시스템으로 이벤트만 멀티캐

스팅 하므로 네트워크 부하가 적어 실시간 전송이 가능하며, 응용 프로그램의 실행은 각 시스템에서 이루어지므로 그룹 구성원들의 공유 데이타에 대한 일관성을 유지할 수 있다.

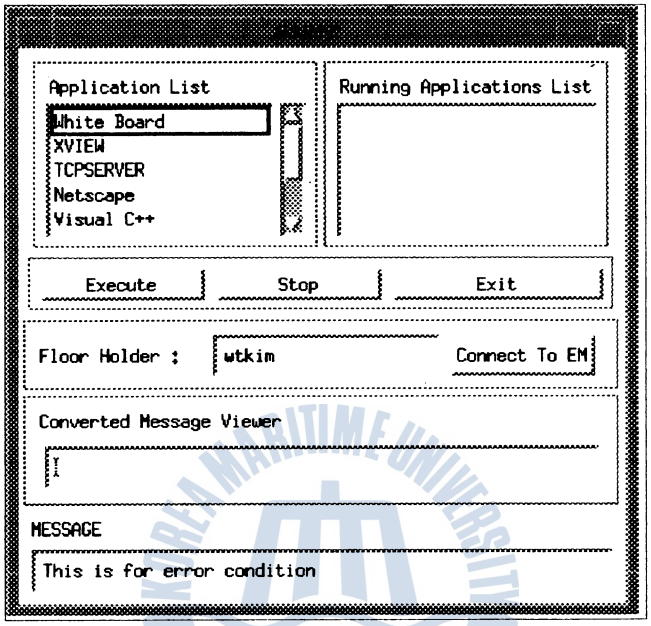

그림 12. UNIX 환경에서의 사용자 인터페이스

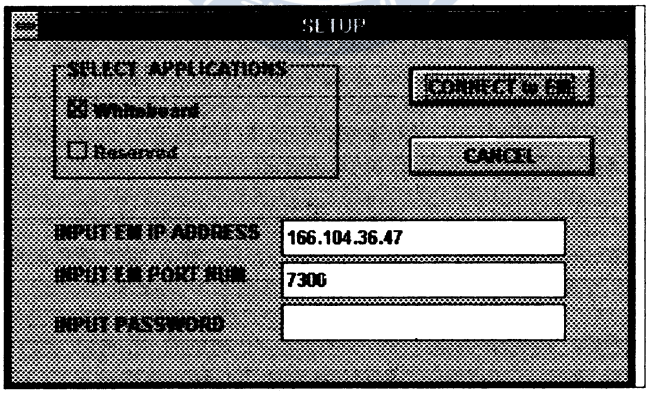

그림 13. PC 환경에서의 사용자 인터페이스

4. 결 론

본 논문에서는 이기종 분산 환경에서 공동 작업을 지원하는 응용 공유 서버의 설계 및 구현에 관하여 기술하였다. 구현한 응용 공유 서버를 MS-Windows 3.1/PC와 X-Window/UNIX workstation 환경에서 그룹웨어의 기본적인 도구인 화이트보드를 이용하여 시험하였으며, 좋은 결과를 얻을 수 있었다. 제안된 응용 공유 서버는 기존의 응용 공유 서버들이 해결하지 못하던 이기종 환경에서의 응용 프로그램 공유를 실현하였으며, 구조적으로 발언권 제어, 데이타의 일관성 유지 및 이벤트의 멀티캐스팅에 적합한 구조를 가진다.

향후 과제로써 PC에서 수행될 수 있는 이벤트 멀티캐스팅 모듈의 구현과 Windows 95나 OS/2 윈도우 시스템 등과 같은 다른 운영체제상에서 운영될 수 있는 응용 공유 서버를 구현해야 할 것이다.

참고 문헌

1) John F. Pattreson, "The Good, the Bad and the Ugly of Window Sharing in X," Bellcore.
2) Kieran Taylor, "Desktop Videoconferencing Not Ready for Prime Time," ACM Data Communication, Apr., 1995.
3) Ian Smith and Elizabeth Mynatt, "What You See Is What I Want: Experinces With The Virtual X Shared Window System," Technical Report, Georgia Tech. College, 1994.
4) Jonathan Grudin, "Computer-Supported Cooperative Work: History and Focus," IEEE Computer, May, 1994.
5) T. Ohmori, K. Maeno, S. Sakata, H. Fukuoka and K. Watabe, "Distributed Cooperative Control for Application Sharing Based on Multiparty and Multimedia Desktop Conferencing System: MERMAID," C & C System Research Lab, NEC.
6) Hussein Abdel-wahab and Kevin Jeffay, "Issues Problems and Solutions in Sharing X Clients on Multiple Displays," North Carolina University.
7) Ivan Tou, Steven Berson and Gerald Estrin, "Prototyping Synchronous Group Applications," IEEE Computer, May, 1994.
8) Hoyoung Hwang, "Design and Implementation of a Shared Workspace supporting Centralized Conference System," ICOIN-9, 1995.
9) Adrian Nye, Xlib Programming Manual for Version 11, Oreilly & Associates.
10) James L. Conger, Windows API Bible: The Definitive Programmer's Reference, Waite Group Press, 1992.