論文96-33B-7-6

이기종 환경을 위한 응용 프로그램 공유서버의 설계 및 구현

(The Design and Implementation of Application Sharing Server for Heterogeneous Environment)

李 昌 昱 *, 朴 容 震 *

(Chang-Wook Lee and Yong-Jin Park)

요 약

현재의 대부분의 네트워크 환경은 단일한 시스템으로 구성되기보다는 다양한 하드웨어와 소프트웨어 자원들로 구성되어 있으며, 이러한 이질적인 환경을 극복하기 위한 여러 가지 연구가 진행되어왔다. 본 논문에서는 상이한 운영체제를 사용하는 사용자들이 동일한 응용 프로그램을 공유할 수 있도록 하는 응용 프로그램 공유서버의 설계와 구현에 대하여 기술한다. 본 논문에서 구현하는 응용 프로그램 공유서버는 X-Window와 MS-Windows 환경에서 여러 명의 사용자들이 동시에 하나의 응용 프로그램을 공유할 수 있도록 하여 이기종 환경에서의 CSCW를 지원한다. 구현된 응용 프로그램 공유서버는 프로그램 빠른 응답성과 네트워크 부하의 최소화를 고려하였으며, 이 서버를 기반으로 다양한 응용 프로그램을 개발할 수 있는 환경을 제공한다.

Abstracts

The current network environment is composed of heterogeneous H/W and S/W resources. Integrating CSCW applications in such heterogeneous environments is highly complex. This paper describes the design and implementation of ASS(Application Sharing Server) which support sharing of a single application between different operating systems. The ASS supports cooperative work among user group using its own window system(X-window or MS-windows). We adopted improved replicated architecture, so ASS allows improved responsiveness of program and performance for managing heavy-weight data. ASS also provides development environment for various groupware application.

I. 서 론

최근 인터넷의 폭발적인 성장에 힘입어 LAN 환경은 물론 WAN 환경에 적용할 수 있는 다양한 그룹웨어 응용의 개발이 진행되어 왔다. 그러나 현재까지는 동일한 운영체제를 사용하는 시스템들 사이에서만 응용 프로그램의 공유가 지원되고 있을 뿐이며, 여러 종류의 컴퓨터로 구성된 이기종 환경에서는 그룹웨어 응용 프로그램의 공유가 불가능한 실정이다. 현재의 네트워크 환경이 하드웨어나 소프트웨어적으로 이질적인 자원들로 구성되는 것이 일반적임을 고려할 때, 이질적인 환경에서 응용 프로그램의 공유를 통한 협동작업이 가능해져야 할 것이며, 이를 위한 연구가 필요한 시점이다. 이 연구의 일환으로, 본 논문에서는 이기종 시스템으로 구성된 네트워크 환경에서 다수 사용자들이 동일한 응용 프로그램을 공유할 수 있도록 하는 응용 프로그램 공유서버를 설계하고 구현한다.

응용 프로그램을 공유할 때 가장 문제가 되는 것은 프로그램이 실제 운용될 운영체제의 이질성이다. 현재 시스템에서 사용되고있는 대표적인 운영체제로는 UNIX와 MS-Windows가 있으며, 이질적인 환경에서의 프로그램 공유를 위해서는 운영체제의 이질성에서 발생하는 문제를 해결해야 한다. 본 논문에서는 그룹웨

* 正會員, 漢陽大學校 電子工學科

(Dept. of Electronic Eng., Hanyang University)

接受日字:1996年4月26日, 수정완료일:1996年6月21日

어 응용들이 윈도우 시스템을 기반으로 하는 윈도우 응용 프로그램들이라는 사실에 착안하여 UNIX의 X-Window와 MS-Windows 환경에서 응용 프로그램을 공유할 수 있는 공유서버를 구현한다. 본 논문에서 개발하는 응용 프로그램 공유서버는 상이한 윈도우 시스템을 사용하는 사용자들간에 동일한 응용 프로그램을 공유하도록 함으로써 모든 사용자들이 익숙한 환경에서 협동작업에 참여할 수 있도록 한다. 이를 위해 구현되는 응용 프로그램 공유서버는 상이한 윈도우 시스템들 사이의 이벤트를 변환할 수 있는 기능, 다중의 사용자를 지원하기 위한 멀티캐스팅 기능, 사용자들 사이의 입력 권한을 제어하는 발언권 제어 기능을 제공한다. 구현된 응용 프로그램 공유서버는 각각의 응용 프로그램이 참가자의 시스템에서 실행되는 복제 실행 구조를 지원함으로써 네트워크의 부하를 최소화하고 사용자의 입력에 대한 빠른 응답성을 지원하는 한편, 멀티캐스팅과 발언권 제어를 위한 하나의 서버를 둠으로써 각 응용에 분산된 데이터의 일관성 유지하고 실행 중인 세션에 동적인 가입과 탈퇴를 지원한다. 또한 응용 프로그램 공유서버를 이용할 수 있도록 API를 제공함으로써 다양한 그룹웨어 응용 프로그램을 개발할 수 있는 환경을 지원한다.

본 논문의 2장에서는 그룹웨어 환경에서 응용 프로그램의 공유를 위한 접근방법과 구현된 응용 프로그램 공유서버의 구조에 설명하고 3장에서는 응용 프로그램 공유서버의 구현에 대해 기술한다. 그리고 4장에서 구현된 서버의 성능을 평가한 후 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 응용 프로그램 공유서버의 설계

1. 설계 목적 및 요구사항

본 논문에서 설계한 ASS는 네 가지 사항을 만족할 수 있는 구조를 선택하였다.

i) 발언권 제어의 용이 : 여러 사용자들이 하나의 응용 프로그램을 공유할 경우 입력의 권한에 대한 적절한 제어가 필요하며, 이러한 발언권 제어가 공유하는 프로그램의 특성에 따라 융통성 있게 지원되어야 한다.

ii) 네트워크 부하 감소 : 프로그램의 공유를 위해서는 멀티캐스팅은 필수적이다. 멀티캐스팅하는 데이터를 최소화함으로써 네트워크 부하를 감소시켜 입력에 대한 빠른 응답을 기대할 수 있다.

iii) 확장성 : 특정 시스템에 대한 의존성을 최소화하여 시스템의 확장성 및 안정성을 보장할 수 있어야 한다.

iv) 공유 데이터의 일관성 유지

v) 윈도우 시스템의 이벤트 변환 : 구현하고자 하는 공유서버는 상이한 윈도우 환경에서의 프로그램 공유를 지원한다. 따라서 서버를 통해 공유되는 프로그램은 윈도우 기반의 응용 프로그램들이다. 윈도우 시스템은 기본적으로 이벤트 구동(Event-driven) 방식을 취하고 있다. 즉 사용자의 입력을 응용 프로그램이 직접 받아서 처리하는 것이 아니라, 윈도우 시스템이 이벤트의 형식을 빌어 현재 사용자의 입력 내용을 응용 프로그램에게 알려 주면 응용 프로그램이 그 이벤트를 자신의 목적에 맞도록 처리하게 된다. 현재 이러한 이벤트는 윈도우 시스템들마다 다르게 정의되어있으며, 따라서 서로 다른 윈도우 시스템에서 운용되고 있는 응용 프로그램을 공유하기 위해서는 하나의 시스템에서 발생한 이벤트를 다른 윈도우 시스템에서 정의된 동일한 기능의 이벤트로 변환할 수 있는 기능이 필요하다.

2. 응용 프로그램의 실행 형태

사용자들이 응용 프로그램을 어떤 방식으로 공유할 것인지를 결정해주는 응용 프로그램의 실행형태는 본 논문에서 구현하고자 하는 응용 프로그램 공유서버의 구조는 물론 구현된 시스템의 성능을 결정하는 중요한 요소로 작용한다. 지금까지 응용 프로그램의 실행형태에 대해서는 여러 방식에 대한 장단점이 논의되어 왔다[1,2,3,4]. 현재까지 개발된 시스템들은 크게 다음 세 가지 모델로 분류될 수 있으며, 각각의 장단점은 다음과 같다.

1) 단일실행(centralized) 구조

전체 시스템에서 단지 하나의 응용 프로그램이 실행되며, 사용자 입력은 응용 프로그램으로 전달되고 그 실행결과는 다시 사용자 시스템으로 전달된다. 따라서 각 사용자의 시스템은 그래픽 터미널과 윈도우 서버로서 동작한다. 이 구조는 첫째, 응용 프로그램과 데이터가 하나의 시스템에서 관리되므로 접근 관리와 데이터의 일관성(consistency)의 유지가 간편하고 둘째, X-Window와 같은 서버 기반의 윈도우 시스템에서

구현이 용이하며 셋째, 서버 기반의 윈도우 시스템(X-Window)에서 기존의 단일 사용자 프로그램을 수정 없이 사용할 수 있고 넷째, 전체 시스템들 중 하나의 시스템에서만 응용 프로그램이 실행되므로 응용 프로그램이 이식되지 않은 시스템의 사용자도 그룹작업에 참가할 수 있다는 장점이 있는 반면, 응용 프로그램을 실행하는 시스템의 고장시 시스템의 운영이 불가능하고, 사용자의 입력이 원거리의 서버로 전송된 후 그 실행결과가 다시 네트워크를 통해 전송되므로 응답속도가 느릴 뿐 아니라 실행된 결과가 모든 참가자의 워크스테이션으로 재전송되어야 하므로 네트워크 부하가 커지게 되며, 서로 다른 윈도우 시스템을 사용하는 시스템들 사이에는 응용 프로그램의 공유가 불가능하므로 이기종 환경의 지원에 적합하지 않다는 단점을 갖고 있다. 이 구조를 사용하는 대표적인 시스템으로는 Rendezvous[1]를 들 수 있다.

2) 복제실행(replicated) 구조

모든 참석자의 시스템에서 응용 프로그램의 복사본을 실행시킨다. 응용 프로그램을 각 시스템의 환경(특히 윈도우 시스템)에 맞게 구성할 수 있다. 따라서 다른 윈도우 시스템을 사용하는 시스템들 사이에서도 응용 프로그램의 공유가 가능하므로 이질적인 환경에 적합하다. 또한 사용자들의 입력만을 멀티캐스트하며, 실제 응용 프로그램의 실행은 각 시스템에서 이루어지므로 실행결과를 멀티캐스트할 필요가 없으며, 따라서 네트워크의 부하가 적고 사용자의 입력에 대한 응답이 빠르다. 그러나 각 사용자의 시스템에서 자체적으로 응용 프로그램을 실행하므로 데이터 역시 각 시스템에 저장되어야 한다. 따라서 데이터에 대한 일관성의 유지가 어렵고, 프로그램들 사이의 동기유지가 어렵다는 단점이외에 기존의 단일 사용자 프로그램을 수정 없이 사용할 수 없고, 진행중인 세션에 새로운 사용자가 참가할 때, 지금까지 실행된 프로그램의 상태정보를 기존에 참가중인 사용자가 전송해주어야 하므로 그룹작업시 사용자의 임의 탈퇴 및 가입의 지원에 어려움이 있다. 복제실행 구조를 채택한 예로는 MMConf[5]가 있다.

3) 복합(hybrid) 구조

두 가지 모델의 극단적인 장단점을 해소하기 위한 구조로서 응용 프로그램을 사용자 인터페이스 모듈과 기능 모듈로 나누어 기능 모듈은 단일 구조로 운영하고 인터페이스 모듈을 각 사용자의 시스템에서 실행시킨다. 이질적인 환경에 적합하지만 인터페이스 모듈과 기능 모듈간의 인터페이스에 대한 규정이 필요하며 기존의 프로그램의 대폭적인 수정이 요구된다. 영국 Lancaster 대학에서 개발되고 있는 IXION[6]이 이 구조를 채택하고 있다.

본 논문에서는 상이한 윈도우 환경에서 응용 프로그램의 공유를 가능하게 하는 서버의 개발을 목표로 하며 이러한 목적에는 기본적으로 이기종 환경에 적합한 복제실행(Replicated) 구조가 적합하다. 그러나 순수한 복제실행 구조의 단점인 데이터 일관성 유지와 동기화의 문제를 해결하기 위해 클라이언트-서버 모델을 병행하는 변형된 복제실행 모델을 지원한다. 이 구조는 각 시스템이 완전한 하나의 프로그램을 수행하므로 하나의 응용 프로그램이 사용자 인터페이스 부분과 공유 프로그램 부분으로 나뉘어 실행되는 복합 구조와는 구별되며, 단지 복제실행 모드에서 멀티캐스팅을 위한 하나의 독립된 서버를 사용하는 구조이다. 즉 공유되는 응용 프로그램에 대한 모든 입력이 하나의 서버에서 ASS(Application Sharing Server)를 통해 멀티캐스트됨으로써 모든 프로그램들이 입력을 받는 순서가 동일하게 되어 동기화의 문제를 해결하며, 또한 응용 프로그램의 상태정보를 이 서버에서 관리함으로써 작업 진행도중 새로운 사용자가 가입할 경우 현재의 상태정보를 새로운 가입자에게 전달할 수 있게 되어 사용자의 임의 가입과 탈퇴를 손쉽게 지원할 수 있다.

3. 응용 프로그램 공유서버의 구조

응용 프로그램 공유서버가 상이한 윈도우 시스템간의 프로그램 공유를 지원하기 위해서는 윈도우 시스템들마다 다르게 규정되어 있는 이벤트를 변환할 수 있어야한다. 이 때 각 시스템에서 발생한 입력을 중앙의 ASS에서 각각의 윈도우 시스템에 적합한 이벤트로 변환하여 멀티캐스트하는 경우 ASS가 실행되는 시스템의 부하가 과도해지므로 이벤트의 변환은 각각의 시스템에서 자신의 시스템 특성에 맞게 수행하는 것이 효과적이다. 이를 위해 응용 프로그램이 이벤트 변환을 수행하는 것을 생각할 수 있으나 이 경우 응용 프로그램의 개발시 이벤트 변환을 고려해야하는 어려움이 있다. 따라서 본 논문에서는 이벤트의 변환을 수행하는 ASS 에이전트(Agent)를 각 시스템당 하나씩 둠으로써 프로그래머의 부담을 덜어준다. 즉 ASS 에이전트는 각 시스템에서 발생하는 입력 이벤트를 미리 정의

된 공통의 이벤트로 변환하여 ASS로 전송하고 ASS는 이를 모든 시스템으로 멀티캐스트한다. ASS 에이전트는 ASS로부터 전달받은 공통의 이벤트를 자신의 윈도우 시스템에 맞게 변환한 후 응용 프로그램에게 전달한다.

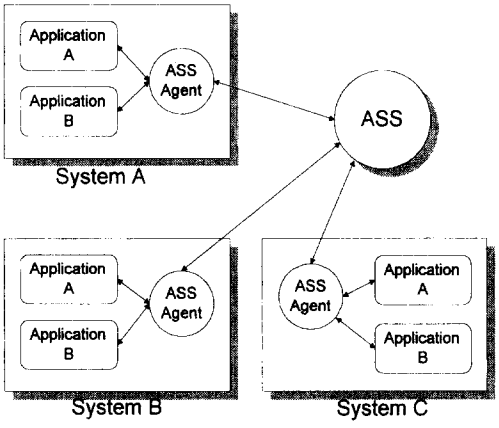


그림 1. 응용 프로그램 공유서버의 구조
Fig. 1. Architecture for Application Sharing.

1) ASS 에이전트의 기능
- 이벤트 변환 : ASS 에이전트는 로컬 윈도우 시스템으로부터 전달된 이벤트를 정의된 글로벌 이벤트로 변환한 후 ASS로 전송한다. 또한 ASS에서 전송된 글로벌 이벤트는 로컬 윈도우 시스템을 위한 이벤트로 변환한 후 응용 프로그램에 전달한다. 현재 변환되는 이벤트는 마우스와 키보드에서 발생되는 이벤트들이다.
- 멀티플렉싱 : 둘 이상의 응용 프로그램이 공유되는 환경을 위해 ASS 에이전트는 멀티플렉싱(multiplexing)과 디멀티플렉싱(demultiplexing)을 수행한다. 이를 위해 각 응용들은 고유의 ID를 부여받는다. 프로그램 실행 초기에 ASS 에이전트는 각 응용 프로그램의 ID와 응용 프로그램 윈도우 ID를 매핑시킨다. 이후 ASS 에이전트는 발생한 이벤트에 대한 글로벌 이벤트를 ASS로 전송할 때 이벤트를 발생시킨 응용 프로그램의 ID를 함께 전송한다. 또한 ASS로부터 글로벌 이벤트가 전송되면 응용 프로그램 ID를 확인하여 이벤트를 전달받을 응용 프로그램을 찾을 수 있다.
- 발언권 제어 : 사용자들은 ASS 에이전트가 제공하는 인터페이스를 이용하여 현재 진행중인 응용 프로그램에 대한 발언권 제어 방침을 선택할 수 있다. 공유된 응용 프로그램들에 대한 발언권 제어 정보는 ASS에서 통합적으로 관리되지만 발언권 제어는 실제적으로는 ASS 에이전트에서 이루어진다. 즉 ASS 에이전트는 시스템의 사용자가 입력 권한을 가진 경우에만 발생한 이벤트를 ASS로 전달한다. 따라서 입력 권한이 없는 사용자의 이벤트 멀티캐스팅을 차단하여 불필요한 네트워크 부하를 줄인다.
- 응용 프로그램의 실행 : ASS 에이전트는 현재 공유 가능한 응용 프로그램의 목록을 제공하며, 사용자는 이 목록으로부터 응용 프로그램을 선택함으로써 프로그램을 실행시킨다.

2) ASS의 기능
- 멀티캐스팅 : ASS는 현재 세션에 참가 중인 모든 사용자들에게 이벤트를 멀티캐스팅한다. 이를 위해 ASS는 모든 참가자들에 대한 멤버 리스트를 유지한다.
- 발언권 제어 : 각 응용 프로그램별로 설정된 발언권 제어 정보를 관리한다. 사용자가 ASS 에이전트가 제공하는 인터페이스를 이용하여 특정 응용 프로그램에 대한 발언권을 요청하면, ASS는 현재 그 응용 프로그램에 지정된 발언권 방침에 따라 요구를 처리한다. 현재 지원되는 발언권 제어 방침은 다음과 같다.
  i) 요청형 (request) : 발언을 마친 사용자는 큐에 등록된 순서대로 발언권을 요구 하는 사용자에게 발언권을 넘겨준다. ASS는 응용 프로그램마다 하나의 큐를 관리하며, 발언권을 가진 사용자가 발언권을 포기하면 큐에 저장된 다음 사용 자에게 입력을 허가한다. 이를 위해 ASS는 발언권을 가진 사용자의 ASS 에이전 트에 이벤트를 전송하도록 하고 이외의 다른 에이전트에 대해서는 모든 로컬 이벤트를 무시하도록 한다.
  ii) 방임형 (free) : 모든 사용자가 언제든지 발언권을 가질 수 있다. ASS는 모든 ASS 에이전트에 이벤트를 전송할 수 있도록 하며, 에이전트로부터의 모든 이벤 트들은 수신되는 순서대로 다른 에이전트들에게 멀티캐스팅한다.

iii) 의장형 (moderated) : 세션의 의장이 사용자에게 발언권을 부여한다. 의장은 ASS 에이전트가 제공하는 인터페이스를 이용하여 각 사용자에게 입력권한을 부 여한다. ASS는 의장이 지정한 사용자의 ASS 에이전트에게 이벤트의 전송을 허 가한다.

Ⅲ. 응용 프로그램 공유서버의 구현

1. 이벤트의 변환

1) 응용 프로그램과 ASS 에이전트사이의 이벤트 정의

응용 프로그램에 대한 입력은 ASS 에이전트로 전달되어 공통의 이벤트 형태로 변환되어 ASS로 넘겨지며, 다시 ASS에 의해 모든 ASS 에이전트에 멀티캐스팅되어 각 윈도우 시스템을 위한 이벤트로 변환된 후 응용 프로그램에 전달된다. 따라서 응용 프로그램과 ASS 에이전트는 윈도우 시스템에서 제공하는 호출함수를 사용하여 이벤트를 교환하는데, X-Window 환경에서는 XSendEvent()를, 그리고 MS-Windows에서는 SendMessage()와 PostMessage()를 사용할 수 있다. MS-Windows에서 SendMessage()는 응용 프로그램간의 직접적인 이벤트의 전달을 지원하며, PostMessage()는 윈도우 시스템을 경유하여 이벤트를 전달한다. PostMessage()를 사용할 경우 SendMessage()와 비교하여 응용 프로그램의 부하를 줄일 수 있으므로 본 구현에서는 PostMessage()를 사용하였다.

응용 프로그램은 입력된 이벤트가 ASS를 통해 전달된 것이면 바로 처리하고, 로컬 윈도우 시스템에서 발생한 이벤트는 ASS 에이전트로 넘겨준다. 따라서 발생한 이벤트가 로컬 윈도우 시스템에 의해 발생한 것인지 ASS를 통해 전달된 것인지를 구분하여 수 있어야 하며, 각각의 응용 프로그램과 ASS 에이전트가 주고받는 이벤트도 정의되어져야 한다. MS-Windows와 X-Window 시스템의 이벤트 처리 방법이 다르기 때문에 각각 다른 방법접근 방법을 사용하였다.

MS-Windows의 이벤트 구조는 그림 2와 같이 정의되어 있는데, 이중 message 필드를 이용하여 이벤트의 종류를 정의하고 이를 이용하여 발생한 이벤트를 구분할 수 있다. 본 논문에서 응용 프로그램과 ASS 에이전트가 사용할 이벤트를 그림 3과 같이 정의하였으며 각 이벤트의 기능을 표 1에서 설명하고 있다.

```
typedefs struct tagMSG {
        HWND   hwnd;    /* 이벤트가 발생한 윈도우의 ID */
        WORD   message; /* 이벤트의 타입 */
        WORD   wParam;  /* 이벤트 발생시 키의 상태 및 선택된 버튼 */
        LONG   lParam;  /* 윈도우에서 이벤트 발생 위치 */
        DWORD  time;    /* 이벤트 발생시간 */
        POINT  pt;      /* 스크린에서 이벤트 발생위치 */
        }      MSG;
```

그림 2. MS-Windows의 이벤트 구조
Fig. 2. Event structure for MS-Windows.

```
#define WM_COM_INITIAL          WM_USER + 100
#define WM_COM_ACK              WM_USER + 101
#define WM_COM_MOUSEMOVE        WM_USER + 102
#define WM_COM_LBUTTONDOWN      WM_USER + 103
#define WM_COM_LBUTTONUP        WM_USER + 104
#define WM_COM_DISCONNECTION    WM_USER + 105
```

그림 3. 응용 프로그램과 ASS 에이전트의 통신을 위한 이벤트
Fig. 3. Events for communication between Application and ASS-Agent.

표 1. 정의된 이벤트의 기능
Table 1. Function of events defined.

정의된 이벤트	기 능
WM_COM_INITIAL	ASS 에이전트는 응용 프로그램을 처음 기동시킬 때 WM_COM_INITIAL을 이용하여 에이전트의 WindowID를 로컬 윈도우 시스템에 브로드캐스팅함으로써 자신의 WindowID를 응용 프로그램에게 알린다.
WM_COM_ACK	WM_COM_INITIAL을 받은 응용 프로그램은 그 응답으로 자신의 WindowID가 담긴 WM_COM_ACK를 ASS 에이전트에게 보내어 응용 프로그램들 간의 통신 통로를 확보한다
WM_COM_MOUSEMOVE	마우스가 움직이면 WM_MOUSEMOVE 이벤트가 발생하며, 이 이벤트는 해당 응용 프로그램에서 곧바로 처리하지 않고 일단 ASS 에이전트로 보내지고 에이전트는 그 이벤트를 발생시킨 사용자가 발언권을 가졌는지를 검사한다. WM_COM_MOUSEMOVE는 응용 프로그램으로부터 에이전트로 전달되는 마우스움직임 이벤트이다.
WM_COM_LBUTTONDOWN	마우스의 왼쪽 버튼이 클릭되면 윈도우 시스템에서는 WM_LBUTTONDOWN 이벤트를 해당 프로그램에게 전송한다. 응용 프로그램은 이 이벤트를 WM_COM_LBUTTONDOWN을 이용하여 에이전트에게 전달한다.
WM_COM_LBUTTONUP	마우스의 왼쪽 버튼이 릴리즈될 때 발생하는 이벤트를 에이전트에게 전달할 때 사용한다.
WM_COM_DISCONNECTION	응용 프로그램이 프로그램의 종료를 ASS 에이전트에게 알리기 위한 이벤트이다.

X-Window에서는 입력된 이벤트가 다른 응용 프로그램으로부터 보내진 것인지, X-서버로부터 보내진 것인지를 구별할 수 있는 필드가 이벤트 구조체 내에 정

의되어 있다. 즉, 응용 프로그램에서 보내진 이벤트일 경우 그림 4의 이벤트 구조 중 send-event 필드의 값이 1로 설정되므로 X-Window에서는 MS-Window에서와는 달리 ASS와 ASS 에이전트를 위한 이벤트를 따로 정의할 필요가 없다. 응용 프로그램은 send-event 필드를 조사하여 로컬 윈도우 시스템에서 발생한 이벤트이면 이를 ASS 에이전트에게 넘겨주며, ASS 에이전트가 전송한 이벤트로 판별되면 그 이벤트를 즉시 처리하게 된다.

```
typedef struct {
        int             type;
        unsigned        long serial;
        Bool            send_event;
          .
          .
        unsigned        int state;
        unsigned        int button;
        Bool            same_screen;
        } XButtonEvent;

typedef struct {
        int             type;
        unsigned        long serial;
        Bool            send_event;
          .
          .
        unsigned        int state;
        unsigned        int is_hint;
        Bool            same_screen;
        } XMotionEvent;
```

그림 4. X-Window 시스템에서의 이벤트 구조
Fig. 4. Event Structures for X-Window.

2) 글로벌 이벤트의 정의

윈도우 시스템들은 프로그램에 대한 입력을 이벤트로 처리한다는 공통점이 있지만 이벤트의 종류 및 처리 방식에는 차이가 있다. 따라서 상이한 시스템들 사이에 이벤트를 전송하기 위해서는 이벤트의 변환이 필요하며, 이 변환은 각 윈도우 시스템들의 유사한 이벤트들 사이에 이루어진다. 본 논문에서는 이벤트의 변환에 따른 부하를 최소화하기 위하여 멀티캐스팅되는 모든 이벤트들을 공통의 이벤트로 변환하여 전송한 후 각 시스템에서 이 이벤트를 자신의 시스템에 맞는 이벤트로 변환하도록 하였다. 이를 위해 그림 5와 같은 글로벌 이벤트를 정의하였다. 공유된 프로그램으로 전달되는 로컬 이벤트는 일단 ASS 에이전트로 전달되며, ASS 에이전트는 이를 글로벌 이벤트로 변환하여 ASS로 전송하고 ASS는 이 글로벌 이벤트를 다른 모든 ASS 에이전트에게 멀티캐스팅한다.

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

typedef  UButtonEvent {
        int   header;   /* 제어용 이벤트와 정보교환 이벤트의 구별 */
        int   type;     /* 이벤트 타입 */
        int   user_id;  /* 이벤트를 발생시킨 사용자 구별 */
        int   application_id;
                        /* 이벤트가 발생한 응용 프로그램 구별 */
        int   tool;     /* 이벤트 발생시 응용 프로그램이 사용하는 도구*/
        int   point_x,point_y;
                        /* 윈도우내에서 이벤트 발생 위치 */
        int   state;    /* 이벤트 발생시 키의 상태 */
        int   button;   /* 이벤트를 발생시킨 키의 번호 */
        } UButtonEvent ;

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

그림 5. 글로벌 이벤트를 위한 구조체
Fig. 5. Structure defined for global events.

표 2. 글로벌 이벤트와 로컬 이벤트의 관계
Table 2. Mapping relation between global and local events.

글로벌 이벤트	로컬 이벤트		기 능
	X-Window	MS-Windows	
UBottonEvent (BUTTONDOWN)	XButtonEvent (ButtonPress)	WM_LBUTTONDOWN	버튼 클릭
UBottonEvent (BUTTONUP)	XButtonEvent (ButtonRelease)	WM_LBUTTONUP	버튼 릴리즈
UMotionEvent	XMotionEvent (MotionNotify)	WM_MOUSEMOVE	마우스 이동

ASS 에이전트는 ASS로부터 글로벌 이벤트를 전달받으면 이를 다시 로컬 윈도우 시스템에 적합한 로컬 이벤트로 변환하여 응용 프로그램에게 전달한다. 따라서 ASS 에이전트는 그림 5에서 정의된 글로벌 이벤트와 로컬 이벤트 사이의 변환을 수행해야 한다. 표 2는 글로벌 이벤트와 로컬 이벤트 사이의 매핑을 보여준다.

2. 응용 프로그램과 ASS 사이의 통신

1) 응용 프로그램과 ASS 에이전트의 통신

응용 프로그램과 ASS 에이전트는 그림 3에서 정의

한 이벤트를 사용하여 통신한다. ASS 에이젼트는 응용 프로그램이 처음 실행될 때 WM_COM_INITIAL 이벤트를 사용하여 자신의 WindowID를 응용 프로그램에게 알려주며, 응용 프로그램은 그 응답으로 자신의 WindowID가 담긴 WM_COM_ACK 이벤트를 ASS 에이젼트에게 보내어 둘 사이의 통신 통로를 확보한다 (그림 6). 이 후 응용 프로그램은 로컬 윈도우 시스템에서 발생한 이벤트를 ASS 에이젼트로 넘겨주고, ASS 에이젼트는 멀티캐스팅되어온 글로벌 이벤트를 로컬 이벤트로 변환하여 응용 프로그램으로 전달한다. 응용 프로그램이 종료될 때 WM_COM_DISCONNECTION 이벤트를 ASS 에이젼트로 전달한다.

표 3. ASS와 ASS 에이젼트의 통신을 위한 프리미티브

Table 3. Primitives for communication between ASS and ASS-Agent.

프리미티브	기 능
REQUEST_CONNECT_FROM_AA	ASS 에이젼트가 ASS와의 접속 요청
ACK_CONNECT_FROM_ASS	ASS 에이젼트의 요청에 대한 ASS의 응답
PASSWD_PACKET	사용자가 그룹의 멤버임을 확인하는 패스워드 정보
CAN_SERVE, CANNOT_SERVE	사용자의 패스워드를 확인한 후 서비스의 승낙 및 거부
FloorInformation	현재 입력권한을 가진 사용자에 대한 정보 (응용 프로그램의 ID, 현재 입력 권한을 가진 사용자 ID)
DISCONNECT_FROM_ASS	세션의 종료를 ASS 에이젼트에 통보
DISCONNECT_FROM_AA	진행중인 세션으로부터의 탈퇴를 ASS에 통보

2) ASS와 ASS 에이젼트의 통신

ASS와 ASS 에이젼트의 통신은 TCP 패킷을 이용한다. 표 3은 ASS와 ASS 에이젼트 사이의 통신에 사용되는 프리미티브와 그 기능을 설명한다.

ASS 에이젼트와 ASS는 표 3에 정의된 프리미티브들을 이용하여 세션의 설정 및 종료, 발언권 제어 등과 관련된 제어정보를 교환한다(그림 6). 세션이 진행 중인 동안에는 ASS와 ASS 에이젼트사이에 교환되는 데이터는 글로벌 이벤트이다.

Ⅳ. 평가 및 고찰

본 논문에서 구현한 응용 프로그램 공유서버를 평가하기 위해 ASS가 제공하는 API를 이용하여 화이트 보드(White B/D) 형태의 응용 프로그램을 구현하였다. 화이트 보드 프로그램은 X-Window용과 MS-Windows용을 구현하였으며, 각각의 윈도우 환경에서 실행된다.

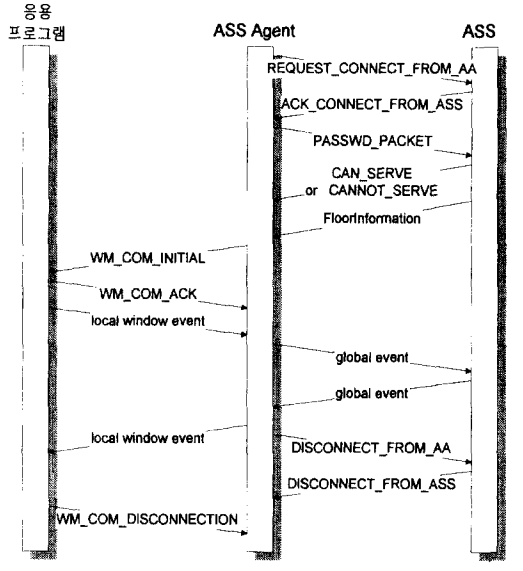


그림 6. 응용 프로그램과 ASS 사이의 데이터 흐름
Fig. 6. Data flow between application and ASS.

실행환경은 FDDI및 Ethernet으로 연결된 3 대의 펜티엄 PC 와 SUN 워크스테이션을 이용하였다. PC에서는 MS-Windows용의 화이트 보드 프로그램을 실행하고 SUN 워크스테이션에서는 X-Window용 프로그램을 실행하였으며, ASS는 워크스테이션에서 실행함으로서 PC에 주어지는 부하를 최소화시켰다. 또한 ASS 에이전트의 멀티플렉싱 기능을 시험하기 위하여 각 시스템에서는 세 개의 화이트 보드를 실행시켰으며, 각 프로그램마다 서로 다른 발언권 제어를 적용하였다. 실행 결과 공유된 프로그램들은 사용자의 입력에 대한 빠른 응답 능력을 보여주었으며, 또한 하나의 ASS에서 모든 이벤트를 멀티캐스팅함으로써 모든 프로그램들간의 데이터의 동일성이 유지되었다. 그러나 하나의 ASS를 사용할 경우 데이터 동일성 유지라는 장점을 얻게되는 대신 하나뿐인 ASS에 전체 시스템이 의존하게 됨으로써 시스템 운용의 신뢰성 문제와 ASS의 과도한 부하에 의한 병목현상을 유발할 수 있다. 특히 ASS의 부하에 의한 병목현상은 멀티미디어의 처리가 요구되는 환경에서는 중요한 문제점으로 작용한다. 즉 오디오와 비디오와 같은 연속매체 데이터의 처리가 필요한 응용 프로그램을 공유해야 하는 환경에서 본 논

문에서와 같이 하나의 ASS를 사용하게 되면 ASS에 부하가 과도하게 커지게 되어 오디오와 비디오의 실시간 처리를 지원할 수 없게되거나 협동작업의 참가자 수를 제한하는 요인이 된다. 따라서 본 논문에서 구현한 ASS는 텍스트와 이미지 기반의 응용 프로그램의 공유에 적합한 구조와 기능을 제공한다.

현재 구현된 프로그램 공유서버는 X-Window와 MS-Windows간의 이벤트 변환만을 구현하였으므로 Windows-NT, Windows 95를 위한 이벤트 변환 기능의 추가가 필요하다. 또한 ASS에 지금까지 수행된 이벤트를 저장함으로써 세션의 진행 중에 새로운 사용자가 참가했을 경우 이전까지 수행된 이벤트를 한꺼번에 전달함으로써, 사용자들이 세션에 동적으로 참가하고 탈퇴하는 기능을 추가할 예정이다.

V. 결 론

본 논문에서는 이기종 환경을 지원하는 응용 프로그램 공유서버를 설계하고 구현에 대하여 기술하였다. 본 논문에서 구현한 응용 프로그램 공유서버는 상이한 윈도우 시스템을 사용하는 사용자들간에 동일한 응용 프로그램을 공유하도록 함으로써 모든 사용자들이 익숙한 환경에서 협동작업에 참여할 수 있도록 한다. 이를 위해 구현되는 응용 프로그램 공유서버는 상이한 윈도우 시스템들 사이의 이벤트를 변환할 수 있는 기능, 다중의 사용자를 지원하기 위한 멀티캐스팅 기능, 사용자들 사이의 입력 권한을 제어하는 흐름제어 기능을 제공한다. 구현된 응용 프로그램 공유서버는 각각의 응용 프로그램이 참가자의 시스템에서 실행되는 복제 실행 구조를 지원함으로써 네트워크의 부하를 최소화하고 사용자의 입력에 대한 빠른 응답성을 지원하는 한편, 멀티캐스팅과 흐름제어를 위한 하나의 서버를 둠으로써 각 응용에 분산된 데이터의 일관성 유지하고 실행중인 세션에 동적인 가입과 탈퇴를 지원할 수 있다. 또한 응용 프로그램 공유서버를 이용할 수 있도록 API를 제공함으로써 다양한 그룹웨어 응용 프로그램을 개발할 수 있는 환경을 지원한다.

참 고 문 헌

[1] John F.Patterson, Ralph D.hill, Steven L.Rohall, "Rendezvous : An Architecture for Synchronous Multi-user Application," CSCW'90, pp. 317-328, 1990.

[2] Ahuja.S, Ensor R.Horm D., "The Rapport Multimedia Conferencing System," Proc. of Conf on Office Information Systems, pp. 1-8, 1988.

[3] Walter Reinhard, Jean Schwerizer, and Gerd Volksen, Michael Weber, "CSCW Tools: Concepts and Architectures," IEEE COMPUTER, pp. 28-36, May 1994.

[4] Jian Zhao, H.Ulrich Hoppe, "Supporting Flexible Communication in Heterogeneous Multi-User Environments," Proc.of 14th International Conference on Distributed Computing Systems, pp. 442-449, 1994.

[5] T.Crowley et al., "MMConf:An Infrasturacture for Building Shared Multimedia Applications," Proc. ACM CSCW 90, ACM Press, New York, pp. 329-342, 1990.

[6] G.Coulson, N. Winlliams and G. S. Blair, "Group presentation of multimedia applications in IXION," Computer Communication, Vol.14, No4, pp. 205-215, May 1991.

[7] Chang-Wook Lee, Yong-Jin Park, "Open Platform for Group-working in ODP environment", ICCC 12th, vol 2, August 1995.

[8] Adrian Nye, Xlib Programming Manual for Version 11 Vol 1. Oreilly & Associates.

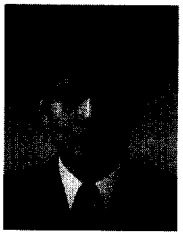
[9] James L. Conger, Windows API Bilble : The Definitive Programmer's Reference, Waite Group Press, 1992.

[10] Mon-Song Chen, Harrick M.Vin, Tsipora Barzilai, "Designing a Distributed Collaborative Environment," IEEE GLOBCOM 92, pp. 213-219, 1992.

[11] R.E. Newman-Wolfe, C.L.Ramirez, H.P elimuhardircum, "A Brief Overview of the DCS Distributed Conferencing System", USENIX Summer'91, pp. 437-451, 1991.

[12] C.Ellis, S.Gibbs, and G.Rein, "Groupware: Some Issues and Experiexnces," Comm. ACM, Vol. 34, No. 1, pp. 38-58, Jan. 1991.

저 자 소 개



李 昌 昱(正會員)

1990년 한양대학교 전자공학과 졸업(공학사). 1992년 한양대학교 대학원 전자공학과 석사학위 취득. 1992년 ~ 현재 한양대학교 대학원 전자공학과 박사과정. 관심분야는 그룹웨어, 멀티미디어 통신, 이동데이터 통신

朴 容 燮(正會員)

1969년 와세다 대학원 전자통신학과 졸업. 1971년 와세다 대학원 석사학위 취득. 1978년 와세다 대학원 박사학위 취득. 1979년 ~ 현재 한양대학교 전자공학과 교수. 1983년 ~ 1984년 Univ. of Illinois, Urbana 전산학과 방문 부교수. 1991년 ~ 1992년 영국 Kent 대학 방문 교수. 1994년 ~ 1995년 개방형 컴퓨터통신 연구회장. 관심분야는 컴퓨터 통신, 분산 시스템, 이동데이터 통신