MissCW:다중 사용자 동기적 공동 저작 시 스템

성 미 영†

요 약

이 논문에서는 멀티미디어 회의를 하면서 공동 편집을 하는 시스템 MissCW(Multiuser Interactive System for Synchronous Collaborative Writing)를 설계하고 구현한 내용을 소개한다. 이 시스템의 문서 모델인 DMDA (Distributed Multimedia Document Architecture)는 논리 구조를 가지며 표현 스타일 객체와 표시 객체를 포함한다. 본 공동 저작 시스템의 동기성은 멀티미디어 회의와 편집 윈도우의 공유 모드로 실현되었다. 이 시스템의 공동 편집기는 분산 객체들을 논리 구조로 조합하여 하나의 문서로 만들 수 있는 구조 지향적 편집 방식을 제공한다. 미들웨어인 공유 객체 관리자(SOM ; Shared Object Manager)는 공유 객체들을 일관성 있게 유지하며 응용 프로그램이 객체들을 효율적으로 이용할 수 있게 도와준다. 이 시스템의 하부 제어 구조는 강력한 서버 없이도 구현이 가능하도록 기본적으로는 복제 구조를 채택하였으나 공유 자료의 일관성 유지를 위하여 가상 노드로의 중앙 집중 구조를 혼합 적용하였다. 가상 노드는 공유 객체 관리자의 객체 제어기에 해당하며 공유 객체 테이블(SOT ; Shared Object Table)을 다루는 모든 일을 한다.

MissCW : Multiuser Interactive System for Synchronous Collaborative Writing

Mee Young Sung†

ABSTRACT

This paper presents the design and the implementation of a MissCW(Multiuser Interactive System for Synchronous Collaborative Writing). The document model DMDA(Distributed Multimedia Document Architecture) of MissCW consists of the logical structure, presentation style object, and mark object. The synchronousness of this system can be realized by the multimedia conferencing and the shared mode of editing windows. The collaborative editor of this system proposes a structure oriented editing mechanism to combine distributed objects into one document. The middleware SOM(Shared Object Manager) maintains shared objects consistently and helps application programs use objects efficiently. The infrastructure of this system is a hybrid structure of replicated and centralized architectures, that is to maintain shared objects consistently inside of SOM and to reduce the overhead of network traffic. The central part is a virtual node which corresponds to the Object Controller of SOM with the SOT(Shared Object Table).

1. 서 론

공동 작업 중 가장 활용도가 높고 중요한 것이 공동 저작(collaborative writing)이다[1, 6]. 공동 저작은 저작자와 저작 대상과의 상호 작용과 공동 저작자들 사이의 의사 소통을 위한 상호 작용을 포함한다. 공동 저작에 있어서 사용자들이 상호 작용하는 방식에는 동기적 상호 작용(synchronous interaction)과 비동기적 상호 작용(asynchronous interaction)이 있다[1]. 동기적 상호 작용을 지원하는 공동 저작 시스템은 문서를 수정하는 내용이 다른 모든 참가자들에게 실시간으로 보여지는 시스템이다. 반대로 비동기적 상호 작용을 하는 공동 저작 시스템에서는 문서를 수정한 내용은 문서를 관리하는 공유 커널에만 기록된다.

※이 논문은 한국과학재단 핵심전문연구(과제번호 961-0902-040-2)의 일환으로 연구되었음.
† 종신회원 : 인천대학교 전자계산학과
논문접수 : 1996년 10월 5일, 심사완료 : 1996년 11월 22일

비동기적 공동 저작 환경에 두 가지 통신 기능이 추가되면 동기적 공동 저작 환경으로 바뀔 수 있다. 하나는 오디오 채널 또는 비디오 채널을 통한 사용자 사이의 직접 통신(direct communication) 기능이고 다른 하나는 공동으로 저작 중인 문서의 변경 내용이 즉시 전달되게 하는 주제별 통신(subject-based communication) 기능이다[1].

공동 저작 시스템을 설계할 때 우선적으로 결정해야 할 사항은 데이터 공유 기법이다. 분산 시스템에서 객체 또는 응용 프로그램을 공유하는 기법에는 출력 복사(protocol replication), 입력 복사(event replication), 이미지 복사(image replication), 객체 공유(shared objects) 기법들이 있다[5]. 우리가 추구하는 기법은 객체 공유 기법이다. 객체 공유 기법을 이용하는 공동 편집 시스템은 분산 객체들을 조합하여 하나의 문서를 만들 수 있기 위하여 논리 구조 지향적이고 구조 편집 지향적이어야 한다[11]. 또한 문서를 구성하는 공유 객체들은 효율적으로 접근되어야 하며 일관성 있게 유지되어야 한다.

본 논문에서는 PC LAN 상에서 동기적 공동 저작을 하는 시스템 MissCW(Multiuser Interactive System for Synchronous Collaborative Writing)를 설계하고 구현한 내용을 소개한다. 시스템은 동기적 상호 작용을 지원하기 위한 멀티미디어 회의[12] 응용과 분산 객체들을 논리 구조로 조합하는 공동 편집기 응용의 두 부분으로 구성되어 있다. 이 논문에서는 공동 편집기의 설계와 구현 내용에 초점을 맞추고자 한다. 우선 본 시스템의 기본이 되는 문서 모델에 대해 살펴본 다음 공동 편집기의 설계와 공유 객체 관리자를 알아보겠다. 이 후 구현 내용을 살펴보고 끝으로 결론을 맺겠다.

2. 문서 모델

이 장에서는 MissCW의 배경이 되는 문서의 논리적 분해(document decomposition) 개념과 분산 멀티미디어 문서 모델인 DMDA(Distributed Multimedia Document Architecture)에 대해 알아보도록 하겠다.

2.1. 문서의 논리적 분해

분산 환경에서 한 문서를 공동으로 다루기 위해서는 문서의 내용을 분해(decomposition)할 필요가 있다. 공유 커널을 이용하는 비동기적 공동 편집 시스템에서, 만일 하나의 문서를 하나의 객체로 한다면 N가지 수정에 대하여 다른 사이트와의 불일치 수정을 하려면 $N(N-1)/2$ 가지 작업을 해야 한다. 그러나 한 문서가 k개($k \ll N$)로 나뉘어 있고 각 부분으로의 접근 확률이 같다면 불일치 수정은 $N(N-k)/2k$ 가지 작업으로 줄어든다[1][1]. 한편 동기적 공동 편집 시스템에서는 하나의 문서가 하나의 객체인 경우는 $N(N-1)$ 가지 작업이 되며 한 문서가 k개로 분해된 경우는 $N(N-k)/k$ 가지 작업이 된다. 그러므로 문서 내용은 적절하게 분해되어야 하며 문서는 분산된 객체들을 찾아가는 포인터들을 모아 놓은 복합 객체가 되어야 한다.

문서는 독립된 여러 개의 조각들로 분해될 수 있다. 이는 여러 공동 저작자가 하나의 데이터를 동시에 접근하는 현상을 현저히 줄일 수 있게 한다. 결국 문서의 분해는 동시성 제어의 크기(granularity of concurrency control)를 줄이는 효과가 있다. 이 크기가 작을수록 여러 저작자가 동시에 작업하는 기회가 늘어난다. 그러나 이 경우 잠금(locking) 등의 동시성 제어 프로시주어를 자주 수행하는 부담이 생긴다.

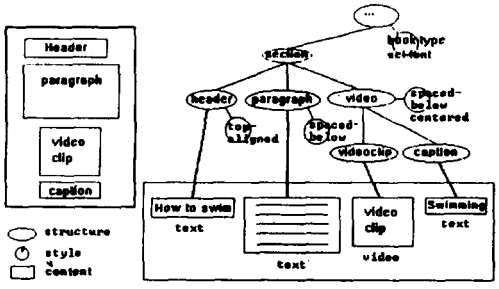
문서의 분해는 일반적으로 장, 절, 소절, 문단, 그림 등의 논리적 단위로 분해될 수도 있고 페이지, 프레임 등의 물리적 단위로 분해될 수도 있다. 문서를 페이지 단위로 분해하면 하나의 절이 두 페이지에 걸쳐 있을 수 있는 경우 그 절을 편집하는 사람은 그 다음 절 또는 그 이전의 절을 편집하는 사람과 동시에 한 페이지에 접근할 가능성 많아진다. 그러므로 문서 접근의 동시성 제어를 위한 부담을 줄이려면 문서는 논리적인 단위로 분해되어야 한다.

문서는 저작자가 정하는 단위의 독립된 조각들로 분해될 수 있고 하나의 조각은 다시 동적이고 순환적

1. 문서의 한 부분은 N번의 수정 작업 중 N/k번 수정되었다. 결국 문서의 한 부분은 자신을 제외한 $(N/k-1)$회 즉 $(N-k)/k$회의 불일치가 가능하다. 그러므로 N회 수정 작업에 대하여 일치와 불일치의 확률이 같다면 $N(N-k)/2k$회의 불일치 수정을 해야 한다.

으로(recursive) 분해될 수 있다. 결과적으로 문서 구조는 내포된 계층 구조(nested hierarchical structure)를 이루게 된다. 그림 1은 한 멀티미디어 문서가 논리적으로 분해된 모습을 보여 준다. 이 예의 문서 구조는 아래와 같은 트리의 리스트 표현법으로 컴퓨터 내부에 표현될 수 있다.

(section(header, paragraph, video(videoclip, caption)))



(그림 1) 멀티미디어 문서 구조의 한 예
(Fig. 1) Document example of a multimedia

2.2. 분산 멀티미디어 문서 구조(DMDA)

본 공동 편집기의 문서 모델인 DMDA(Distributed Multimedia Document Architecture)[11]는 분산 환경에서 멀티미디어 문서를 효율적으로 생성, 저장, 관리할 수 있기 위한 문서 표현 모델이다. DMDA 모델은 ODA(Open Document Architecture)[10]의 영향을 받았으며 문서의 논리적 분해(logical decomposition) 개념에서부터 출발한다.

DMDA의 주요 내용은 다음과 같다.

- 논리 구조(logical structure)와 표현 스타일 객체(presentation style object)

ODA의 전신인 CCITT T.73 Document interchange protocol for telematic services)은 텔레마틱 문서 교환을 위한 프로토콜이었으며 여기에는 배치 구조만이 있었다. T.73의 영향을 받은 ODA에는 문서의 논리 구조와 배치 구조가 있다. 배치에 관한 정보의 전달은 필요한 것이나 하나의 문서에 대해 두 개의 구조를 유지하는 것은 문서 표현 자체를 복잡하고 비효율적으로 만든다. ODA에서도 논리 구조만을 가진 문서를 교환하는 것을 지원한다. 그래서 DMDA에는 그림 1에서 보는 것 같이 단지 논리 구조만 있고 배치에 관한 정보는 표현 스타일 객체에 담아 표현 처리 과정에서 이용한다. 표현 스타일 객체들은 그것이 연결된 논리 객체의 계층 구조에 준하여 속성을 계승받는다.

- 동적 배치(dynamic layout)

각각의 논리 구조 객체에 연결된 표현 스타일 객체 안의 정의가 변화함에 따라 한 문서의 내용을 여러 모양으로 표현(presentation)할 수 있다. 표현 스타일에는 페이지 단위의 출력, 섹션 단위의 출력, 전 문서의 스크롤 등의 배치 모드(layout mode)와 신문, 책, 논문, 보고서, 편지, 사전, 공문, 시나리오 등의 문서 타입이 정의될 수 있으며 각 문서 타입 고유의 표현 객체 속성에 따른 배치가 일어난다. 이를 위해서는 배치 좌표를 다른 객체와의 상대적인 좌표로 정의하는 상대 공간 합성(relative spatial composition)이 필요하다. 만일 책 모양(book design) 데이터 베이스를 유지할 수 있다면 표현 객체에 정의된 속성은 책 모양 데이터 베이스와 연계되어 문서 전체를 동적으로 재배치하도록 운영할 수 있다.

- 표시 객체(mark object)

ODA는 원래 문서의 교환을 목적으로 정의된 것이라 공동 작업에 대한 배려가 없다. 공동 작업을 하기 위해서는 문서 위에 어떤 표시를 하거나 간단한 글 또는 그림을 그릴 필요가 있다. 이렇게 문서 위에 덧붙여진 내용은 표시 객체에 담겨진다. 표시 객체는 다른 공유자에게 전송되기도 하고 나중에 다시 참조할 수 있기 위해 표시가 행해진 논리 객체에 연결된다.

- 프로화일(profile)

문서 파일을 저장시키면 저자, 제목, 주제, 키워드, 요약문, 생성 날짜, 버전과 전문 탐색(full-content retrieval)을 위한 인덱스 등이 프로화일에 수동 또는 자동으로 저장된다.

- 사용자 관점(user view)

문서의 논리 구조는 사용자가 원하는 섹션만 읽거나, 그림들만 보거나, 제목에 특정 단어가 들어 있는 부분만 필터링하여 살펴보는 것을 가능하게 한다.

- 버전 관리(version control)

문서의 논리 구조는 버전 관리를 쉽게 해준다. 문서 내용의 일부가 수정되었을 경우 문서 전체의 새로운 복사본을 만들 것이 아니라 단지 바뀐 부분에 해당하는 단말의 내용조각으로의 연결만을 새것으로 바꾸면 된다.

• 접근 제어(access control)

분산 시스템에서 문서를 공유할 때 가장 문제가 되는 것이 자료의 일관성(consistency) 유지와 보안 문제이다. 동시 접근 제어의 단위를 논리 구조의 단말 객체로 하여 한 순간에 한 문서에 여러 공동 편집자가 동시에 접근할 수 있게 할 수 있다. 문서의 논리 구조는 문서 전체 뿐 아니라 각 논리 객체 단위로 접근 권한을 부여할 수 있게 한다. 또한 논리 객체 단위로 저자 이름을 부여하여 저작권(authority)도 제어할 수 있다.

DMDA는 간결하고 효율적으로 공동의 문서를 표현하기 위해 논리 구조만 있고 배치 정보는 표현 스타일 객체에 담겨 논리 구조에 연결된다. 또한 DMDA는 표현 스타일에 따른 동적 배치가 가능하며 문서 위에 표시한 내용을 담는 표시 객체(mark object)를 포함한다. DMDA는 자료의 일관성 유지를 위하여 논리 구조의 객체 단위로 동시성을 제어하는 공동 작업을 위한 객체 지향적 분산 멀티미디어 문서 모델이다.

참고 문헌 [1]에 소개된 Duplex에서도 문서의 분해를 강조하고 있다. Duplex는 비동기적인 문서 공동 편집 모델인데 반해 MissCW는 동기적 문서 공동 편집 모델이다. Duplex는 공유 객체들을 담고 있는 커널을 가지고 있으며 Duplex의 한 문서는 그 문서가 만들어진 사이트의 지역 데이터와 커널의 객체들만을 포함한다. 반면 MissCW의 한 문서는 모든 사이트의 모든 객체들을 포함할 수 있다.

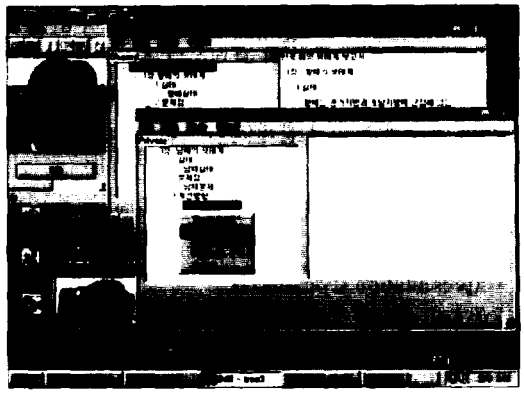
3. 공동 편집기

공동 편집기는 모든 시공 모드[3]에서 이용될 수 있는 중요한 도구이다. 본 공동 편집기는 구조 편집기(structure editor), 내용 편집기(content editor), 객체 등록기(object register), 객체 열람기 (object browser) 그리고 페이지 출력기(page formatter)로 구성된다. 공동 편집기가 사용하는 객체들은 5장에서 소개될 공유 객체 관리자(Shared Object Manager;SOM)에 의해 관리된다.

공동 편집기는 여러 개의 자식 윈도우를 생성할 수 있으며 각 윈도우는 서로 다른 문서를 편집할 수 있는 독립적인 환경이 된다. 공동 편집기의 각 윈도우에는 두 가지 편집 모드가 있다. 하나는 동기적인 공동 작업을 위한 공유 모드(shared mode)이고 다른 하나는 비동기적 공동 작업을 위한 사유 모드(private mode)이다. 사용자는 편집 도중 이 편집 모드를 바꿀 수 있으며 공유 모드로 설정되는 즉시 그 윈도우에서 편집하는 내용은 다른 참가자에게 전송되어 보여지게 된다. 이 장에서는 공동 편집기의 각 구성요소들을 살펴보고 분산 객체 관리자는 4장에서 자세히 다루겠다.

3.1. 구조 편집기

그림 2에 나타나 있는 것처럼 구조 편집기로는 트리의 레벨을 들여 쓰기로 나타내는 목차와 비슷한 형태의 트리 구조를 편집할 수 있다. 구조 편집기를 이용하여 문서 구조를 구상하고 각 단말에 내용 조각을 연결시킨다. 단말의 내용이 이미 만들어진 것일 경우 객체 열람기를 호출하여 원하는 객체를 선택하여 그 단말의 내용 조각으로 연결한다. 그렇지 않은 경우에는 내용 편집기를 불러 그 내용 조각을 새로 편집한다. 이 때 새로 만들어진 내용 조각은 객체 등록기의 도움을 받아 4.1절에서 소개되는 공유 객체 테이블(Shared Object Table)에 등록될 수 있다.

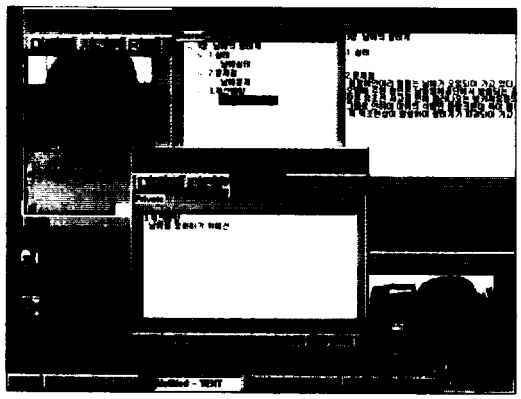


(그림 2) 구조 편집의 한 예
(Fig. 2) An example of structure editing

3.2. 내용 편집기

내용 편집기는 그림 3에서 보는 것처럼 구조의 단말을 선택한 상태에서 그 객체의 타입에 따른 개별 미디어 편집기가 불려진다. 편집한 결과는 그 단말의 내용으로 연결된다. 현재 텍스트 편집만이 가능하나

멀티미디어 회의 응용의 전자 칠판[12]에서 구현한 그래픽 편집기와 연결하고 다른 도구로 만들어진 이미지나 비디오 내용을 포함하도록 하여 멀티미디어 문서 공동 편집기로 만드는 작업을 진행하고 있다.



(그림 3) 내용 편집의 한 예
(Fig. 3) Editing example of content

3.3. 객체 등록기

공동 편집기에서 객체가 새로 생길 때마다 공유 객체 테이블에 등록시킬 수 있다. 기본적으로 객체 이름(완전 논리 객체이름; FLON, 4.1절 참조)들은 중복되어서는 안된다. 객체 등록이 수행될 때마다 같은 객체 이름이 존재하는지 조사하여 이름이 중복되는 경우 편집자에게 알려 조치하게 한다.

3.4. 객체 열람기

문서의 내용을 편집할 때 이미 만들진 내용을 이용할 수 있기 위하여 공동 편집기가 유지하고 있는 객체들을 조회할 수 있어야 한다. 이를 위하여 객체 열람기는 공유 객체 테이블의 항목들을 열람하고 테이블의 한 항목을 선택하여 그 내용을 볼 수 있게 해준다.

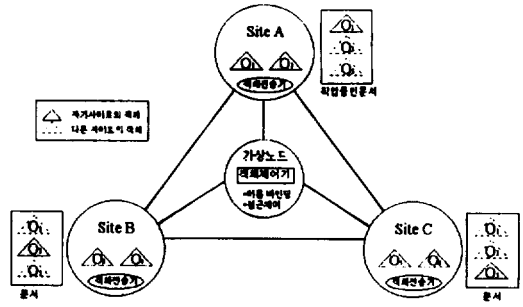
3.5. 페이지 출력기

페이지 출력기는 문서 구조에 정의된 대로 내용 조각들을 찾아와 페이지로 변환시켜 준다. 현재 구현된 상태는 텍스트 출력만이 가능하나 그래픽, 이미지, 사운드, 비디오 등을 포함하도록 하려면 그림의 자리 차지 등 배치 알고리즘의 많은 부분이 보완되어야 한다.

4. 공유 객체 관리자

공유 객체 관리자(Shared Object Manager; SOM)는 미들웨어로서 객체들을 일관성 있게 유지하며 응용 프로그램이 객체들을 효율적으로 이용할 수 있도록 도와주는 일을 한다. 공유 객체 관리자는 객체 제어기(object controller)와 객체 전송기(object transferor)로 구성된다. 그림 4에서 보는 바와 같이 객체 제어기는 가상 노드(virtual node)를 형성하여 총체적인 제어를 담당한다. 가상 노드는 물리적으로는 한 사이트의 공유 객체 관리자 내부에 존재한다. 그리고 각 사이트에는 객체 전송기가 있어 객체의 내용을 빠르게 전송하는 일을 한다.

하나의 공동 문서를 구성하는 객체들은 여러 사이트에 분산되어 있을 수 있다. 그러므로 한 저작자가 자신의 윈도우를 공유 모드로 세트하는 순간 각 사이트의 공동 편집기는 그 문서를 구성하는 객체 중 다른 사이트에 있는 객체의 내용을 실시간으로 전송 받아야 한다. 이 작업을 객체 제어기와 객체 전송기가 어떻게 도와주는지 알아보자.



(그림 4) 문서와 공유 객체 관리자
(Fig. 4) Document and Shared object marager

4.1. 객체 제어기

객체 제어기는 공유 객체 테이블의 내용을 참조하거나 변경하는 모든 일을 한다. 즉, 객체의 논리 이름을 물리 이름으로 사상(mapping)하는 이름 바인딩을 하며 허가권과 동시성 제어 규칙에 따라 객체들의 접근을 제어한다. 또한 한 사이트의 객체 내용이 변경된 경우 다른 사이트에 있는 그 객체의 사본을 갱신하는 일을 관리한다. 그밖에 새로운 객체의 등록도

담당한다.

객체 제어기는 공유 객체에 대한 정보를 담은 공유 객체 테이블(Shared Object Table;SOT)을 이용하는데 만일 SOT를 모든 사이트에 복제한다면 N가지 수정에 대해 N(N-1)가지 불일치 수정 작업을 해야 한다. 객체 제어기에 있어서는 SOT를 중앙의 가상 노드에만 유지하여 일관성을 보장하는 중앙 집중형 제어 기법을 적용하였다.

• 공유 객체 테이블

객체들을 잘 관리하기 위해서는 객체들을 명확하게 식별하는 방법이 필요하다. 그 방법은 객체들에 고유한 이름을 지어 주는 것이다. 이름에는 논리 객체이름 (Logical Object Name)(논리 이름)과 완전 논리 객체이름(Full Logical Object Name)(완전 이름), 그리고 물리 객체이름(Physical Object Name)(물리 이름)이 있다. 논리 이름은 문서 구조에 나타나는 인사말, 본문1, 날짜 등 문서 작성자가 결정하는 객체이름이다. 모든 논리 객체는 그것이 만들어지는 순간 컴퓨터 내부에서 완전 논리 객체이름이 부여된다. 완전 이름은 편지1. 서론. 인사말과 같이 문서 구조의 루트로부터 그 객체까지의 경로상에 있는 객체이름들을 접합(concatenation)한 것이다. 물리 이름은 그 논리 객체의 실제 내용이 들어있는 파일의 경로 이름이다. 물리 객체이름은 그 객체가 가지고 있는 컴퓨터의 IP 주소와 실제 파일 경로 이름을 접합한 형태(IP주소:: 화일경로)이다.

공유 객체 테이블의 한 객체에 대한 항목들은 아래와 같다.

논리이름	완전이름	물리이름	객체 타입	접근 제어	접근 권한	객체 설명
인사말	편지1.본론.인사말	134.75.190.33:: c:\doc\hello.txt	텍스트	잠금	RW	저작자,...
본론	편지1.본론.	134.75.190.33:: c:\doc\main.str	구조	잠금	RW	저작자,...

• 이름 바인딩

한 응용 프로그램에서 이름 바인딩에 대한 요구가 오면 이름 바인더는 논리 이름에 해당하는 물리 이름을 찾아낸다. 이름 바인딩 요구는 완전 이름일 수도 논리 이름일 수도 있는데 구조 편집기의 문서 구조로부터 생긴 요구는 완전 이름이고 객체 등록기나 객체 열람기에서는 논리 이름으로 요구가 온다. 완전 이름으로 요구가 온 경우 요구된 이름의 끝 부분과 테이블의 논리 이름 필드와 일치하는 항목에 대하여 완전 이름을 비교한다. 논리 이름으로 요구가 온 경우는 논리 이름 필드가 같은 항목들을 모두 찾아내어 사용자로 하여금 그 중에서 고를 수 있도록 조치한다. 이렇게 논리 이름이 찾아지면 객체 제어기는 요구된 연산에 따라 적절한 처리를 한다. 만일 읽기나 쓰기 등의 객체 전송 연산이었으면 4.2절에서 설명한 방법으로 처리한다.

• 접근 제어

접근 제어는 크게 두 가지로 구별할 수 있다. 하나는 공유 객체에 대해 읽기 또는 쓰기 접근을 하는 것을 제어하기 위한 접근 권한(access permission)이다. 다른 하나는 쓰기 접근 권한이 있더라도 여러 저작자가 동시에 쓰기 접근을 한 경우 객체의 일관성을 유지하기 위한 동시성 제어(concurrency control)이다.

분산 시스템에서의 동시성 제어는 일반적으로 직렬화(serialization) 기법과 잠금(locking) 기법을 이용하고 있다[4]. 직렬화 기법은 여러 이벤트들로 구성된 하나의 원자적 연산(atomic operation) 사이에 다른 연산의 이벤트가 끼여들지 않고 순서대로 실행되도록 스케줄링하는 기법이다. 낙관적 직렬화(optimistic serialization) 기법은 이벤트들이 섞이더라도 불일치를 찾아내고 수정할 수 있도록 지원하는 기법이다. 잠금 기법은 특정 공유 객체에 대해 접근 특권을 얻어 일정 시간 동안 상호 배제적으로 사용하는 기법이다. 낙관적 잠금(optimistic locking) 기법은 특권을 얻기를 기다리는 동안 미리 그 객체에 대해 조작할 수 있도록 지원하는 기법이다. 직렬화 기법은 스케줄링의 부담으로 처리 시간이 느리고 특히 낙관적 직렬화를 위한 연속적인 undo 처리는 복잡하고 많은 비용이 든다. 여기에서는 구현이 간단한 낙관적이지 않은 잠금 기법을 채택하였고 점차적으로 undo가 가능하게 하여 낙관의 정도를 높여 가려 한다.

접근 제어는 공유 객체 테이블에 등록되어 있는 문서 구조의 객체 단위로 이루어진다. 이는 문서가 논리적으로 분해되어야 하는 중요한 이유이다. 문자 또

는 문장 단위로 접근을 제어하는 일은 매우 복잡하고 통신 부담이 커지며 문서 단위 또는 페이지 단위로 하는 것은 공동 편집의 의미가 없다. 텍스트 문서의 경우 최종 분해 단위는 문단이 적당하다고 여겨진다.

공유 객체에 대한 모든 쓰기 연산은 우선 공유 객체 테이블의 접근 권한 필드를 조사하여 쓰기가 가능한지 확인한 다음 접근 제어 필드의 잠금을 요청하여 접근 특권을 얻어야만 가능하다. 공유 객체에 대한 모든 읽기 연산은 접근 권한이 허락하면 잠금을 요청하지 않아도 된다.

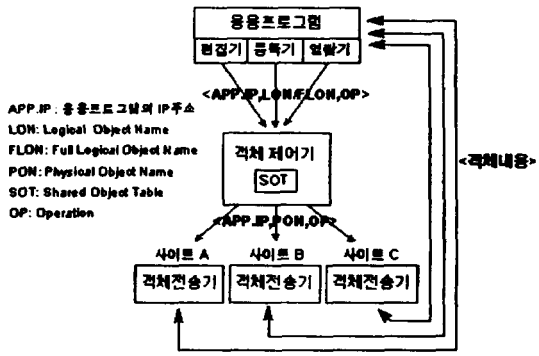
• 멀티캐스팅

한 객체의 편집이 끝나 저장이 일어나면 그 객체의 내용은 객체를 이용하고 있는 모든 사이트로 멀티캐스트되어야 한다. 우선 객체 제어기는 멀티미디어 회의 시스템의 회의 관리자에게 메시지를 보내 방금 저장된 객체를 이용하는 사이트들을 알아낸다. 객체 제어기는 저장이 일어난 사이트의 객체 전송기에 그 객체를 이용하는 모든 사이트들의 IP 주소를 전달한다. 그러면 객체 전송기는 저장된 객체의 내용을 이들 사이트들로 직접 전송한다.

4.2. 객체 전송기

객체 전송기는 요구된 객체의 내용을 응용 프로그램에게 직접 전송하는 일을 한다. 객체 관리자를 구성하는 객체 제어기는 오직 한 사이트에만 존재하였으나 객체 전송기는 모든 사이트에 존재한다. 그림 5는 객체 전송을 위한 제어 흐름을 개괄적으로 보여준다. 한 응용 프로그램이 어떤 객체의 내용을 읽어야 하는 경우 응용 프로그램 클라이언트는 객체 제어기 서버에게 응용 프로그램의 IP 주소와 논리 객체이름 그리고 연산 종류를 전달한다. 객체 제어기 서버는 이름 바인딩을 하여 객체의 물리 이름을 찾는다. 객체 제어기 클라이언트는 물리 이름 안에 밝혀진 IP 주소의 컴퓨터에서 수행 중인 객체 전송기 서버에게 그 물리 객체 이름과 그 객체를 요구한 응용 프로그램의 IP 주소를 연산 종류와 함께 전달한다.

연산 종류가 읽기 연산인 경우에는 객체 전송기 클라이언트는 객체를 요구하는 응용 프로그램 서버와 소켓 연결을 하고 요구된 객체 내용을 응용 프로그램의 IP 주소에 직접 전달한다. 만일 쓰기 연산인 경우에는 반대로 응용 프로그램으로부터 객체 내용을 가져와 파일에 기록한다. 이 때 객체 전송기는 객체가 갱신되었음을 객체 제어기에 알려 객체 제어기로 하여금 멀티캐스팅 할 수 있게 한다. 이렇게 객체 내용을 객체 제어기를 거치지 않고 응용 프로그램에 직접 전달하는 이유는 객체 제어기를 거치는 경우 통신 트래픽이 두 배가되므로 이를 줄이기 위해서이다.



(그림 5) 객체 전송의 제어 흐름
(Fig. 5) Data flow of object transger

5. 구 현

이 장에서는 MissCW의 구현 내용을 간략하게 살펴보도록 하겠다.

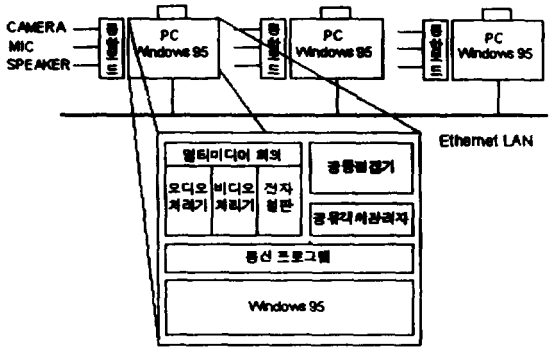
5.1. 시스템 구성과 구현 환경

본 시스템의 전체 구조는 그림 6과 같다. 이 시스템은 크게 멀티미디어 회의 부분과 공동 편집기 부분으로 나뉘어져 있다. 공동 편집기 하부에는 미들웨어 객체 관리자가 있고 하부에는 통신 모듈이 있다.

그림 6의 한 컴퓨터의 구현 환경을 표 1에 요약한다.

5.2. 제어 구조

공동 작업 시스템의 하부 제어 구조에는 중앙 집중형 구조(centralized architecture)와 복제형 구조(replicated architecture)가 있다[2, 3, 4]. 중앙 집중 구조는 모든 자료를 중앙이 통제하므로 자료의 일관성을 유지하기 쉽고 대부분의 회의 제어를 서버가 담당하므로 사



(그림 6) 시스템 구성
(Fig. 6) System structure

〈표 1〉 시스템 구현 환경

구분		내용
H/W	시스템	펜티엄 75 MHz CPU 16 MB 메모리 500 MB 하드 디스크
	오디오/비디오 통합보드	초당 300 프레임 캡쳐 24 비트 칼라 오버레이 16 비트 PCI 사운드
	네트워크	10 Mbps Ethernet LAN
	기타	카메라, 마이크, 스피커
S/W	운영체제	윈도우즈 95
	프로그래밍 환경	Visual C++ 4.0
	망 프로토콜	TCP/IP

용자들의 그룹 응용 소프트웨어가 간단해지는 장점이 있다. 반면에 모든 사용자들이 중앙 사이트를 통하여 통신해야 하기 때문에 통신 부담이 큰 단점이다. 반대로 복제 구조의 경우는 각 사용자들이 통신 제어를 담당해야 하므로 사용자 사이트의 그룹 응용 소프트웨어의 부담이 커지는 단점이 있다. 그러나 상호 작용하는 사이트들 끼리만 자료를 전송하면 되므로 통신 부담이 적어서 열린 시스템의 그룹 작업에 적당하다[3]. 여기에서는 강력한 계산 능력을 가지는 서버 없이도 구현이 가능하도록 기본적으로는 복제 구조를 채택하였다.

그림 6에서 보는 바와 같이 이 시스템에는 멀티미디어 회의[12]와 공동 편집기의 두 응용 프로그램이 있다. 이 두 프로그램은 모든 사이트에 복제된다. 그러므로 이 시스템의 각 컴퓨터는 서버와 클라이언트 역할을 모두 해야 한다. 공동 편집기는 복제형 구조이다. 그러나 하부의 분산 객체 관리자는 혼합형 구조(hybrid architecture)이다. 공유 자료의 일관성 유지를 위하여 객체 제어기는 가상 노드로의 중앙 집중 구조이며 통신 부담을 줄이기 위하여 객체 전송기는 복제형 구조이다.

5.3. 세부 구현 사항

구조 편집기의 트리 구조 편집 구현은 윈도우즈 95가 지원하는 기본 제어 클래스 CTreeCtrl을 이용하였다. 그리고 객체 관리자의 공유 객체 테이블은 윈도우즈 95의 CObList 클래스를 이용하여 리스트 구조로 하였다. 그 결과 구현이 간단할 뿐 아니라 윈도우즈 95와 일관된 사용자 인터페이스를 제공할 수 있었다.

6. 결 론

MissCW는 분산 객체를 논리 구조로 조합하는 편집 방식을 가진 동기적 공동 저작 시스템이다. 이 시스템의 동기성은 멀티미디어 회의와 편집 윈도우의 공유 모드로 구현되었다. MissCW의 의미는 문서를 더 이상 페이지의 연속으로 생각하지 말고 하나의 문서 즉 하나의 정보 덩어리는 분산 객체들을 논리 구조로 조합해 놓은 것으로 생각한다는데 있다. 또한 문서의 수정을 문서의 논리 객체 단위로 가능하게 하여 동시에 가능한 한 많은 저작자가 한 문서에 접근할 수 있게 한다는데 있다.

이 시스템의 공동 편집기는 미들웨어인 공유 객체 관리자의 도움을 받아 객체들을 효율적으로 이용하며 일관성 있게 유지한다. 공유 객체 관리자는 공유 객체 테이블에 등록된 문서 구조의 객체 단위로 접근과 전송을 제어한다. 이 시스템의 하부 제어 구조는 강력한 서버없이도 구현이 가능하도록 기본적으로는 복제형 구조이나 공유 자료의 일관성 유지를 위하여 가상 노드로의 중앙 집중형 구조를 혼합 적용하였다. 가상 노드는 공유 객체 관리자의 객체 제어기에 해당하며 공유 객체 테이블을 다루는 모든 일을 한다.

이 시스템의 가장 큰 장점은 네트워크 상에 분산되어 있는 객체들을 모아 하나의 문서로 합성할 수 있다는 점이다. 다시 말하면 이 시스템은 물리 이름(IP

주소와 파일 경로 이름을 접합한 형태)으로 식별될 수 있는 모든 네트워크 상의 객체들을 문서의 구성 요소로 이용할 수 있게 한다. 만일 이 시스템을 웹 환경에서 동작하도록 확장한다면 인터넷 상에서 웹 브라우저로 검색을 하다가 발견한 웹 페이지, 사진, 그림, 비디오, 소리 자료들을 논리 구조의 단말들에 연결하여 하나의 문서로 합성하는 응용도 가능할 것이다.

한편, 멀티미디어 회의를 위한 데이터 프로토콜인 ITU(International Telecommunication Union)의 T.120 시리즈[7]에는 문서의 구조에 대한 언급이나 문서 공유 프로토콜에 대한 특별한 APE가 정의되어 있지 않다. 현재의 T.120에서 문서를 공유하는 방법은 정적 이미지와 주석을 위한 T.126 APE[9]의 프로토콜에 맞추는 것이다. 이 방법으로는 공유 작업 공간에서 텍스트와 이미지 만을 포함하는 문서를 동기적으로 보여줄 수 있다. 만일 본 시스템을 GAT[8]의 개념에 맞게 문서 공유를 위한 ARM(Application Resource Manager)과 ASE(Application Service Element)를 정의하여 하나의 비표준 응용 프로토콜(non-standard application protocols)로 만든다면 이 경우의 결과물을 T.120의 멀티미디어 문서 공유를 위한 APE 확장으로 제안할 수 있을 것이다.

본 시스템의 문제점을 개선하고 기능을 확장시키기 위한 남은 연구는 아래와 같다.

- 공동 저작 스케줄링, 공동 저작 열람, 공동 저작 합병/분리, 공동 저작 히스토리 기능 등의 공동 작업 관리 기능이 아직 미약하므로 이 부분에 대한 보완이 필요하다.
- 공유 객체 관리에 있어서 중앙 집중 방식을 혼합 적용한 결과 공유 객체 테이블을 가지고 있는 사이트의 처리 부담이 커졌다. 관리 사이트의 통신 부담을 줄이는 기법이 대한 연구가 필요하다.
- 접근 제어에 있어서 잠금을 승인 받을 때까지 계속 기다리는 것은 바람직하지 않다. 낙관의 정도를 높이는 작업이 필요하다.
- 인터넷의 웹 브라우저가 광범위하게 사용되고 있음을 감안하여 웹 상에서 바로 실시간 공동 저작이 가능하도록 본 시스템을 확장하는 지속적인 연구가 진행되어야 하겠다.

참 고 문 헌

[1] F. Pacall, A. Sandoz, A. Schiper, "Duplex : A Distributed Collaborative Editing Environment in Large Scale", Proceedings on CSCW '94, ACM Press, pp165-173, October 22-26, 1994.

[2] Stephen Jabele, Steven Rohall, Ralph L. Vinciguerra, "High Performance Infrastructure for visually-Intensive CSCW Applications", Proceedings on CSCW '94, ACM Press, pp395-403, October 22-26, 1994.

[3] Walter Reinhard, Jean Schweitzer, Gerd Volksen, "CSCW Tools : Concepts and Architectures", IEEE Computer, Vol. 27, No. 5, pp28-36, May 1994.

[4] Saul Greenberg and David Marwood, "Real Time Groupware as a Distributed System : Concurrency Control and its Effect on the Interface", Proceedings on CSCW '94, ACM Press, pp165-173, October 22-26, 1994.

[5] J. D. Palmer, "Computer-Supported Cooperative Work", IEEE Computer, Vol. 27, No. 5, pp15-17, May 1994.

[6] M. E. Hodges, R. M. Sasnett, "Multimedia Computing, Case Studies from MIT Project Athena", 302 pages, Addison-Wesley, 1993.

[7] ITU-T Draft Recommendation T.120, Data Protocols for Multimedia Conferencing, 21 pages, 1996. 8.

[8] ITU-T Draft Recommendation T.121, Generic Application Template, 42 pages, 1996. 8.

[9] ITU-T Draft Recommendation T.126, Multipoint Still Image and Annotation Protocol, 137 pages, 1996. 8.

[10] International Organization for Standardization, "Information technology-Open Document Architecture(ODA) and Interchange Format-Temporal relationships and non-linear structures(ISO/IEC DIS 8613-14 : 1993)", Geneva, 1993.

[11] 성미영, "분산 멀티미디어 환경에서의 공동 저작", 제4회 멀티미디어 산업기술 학술대회 발표 논문집, pp117-126, 1995. 12. 1.

[12] 성미영, 장성룡, 임성근, 김충인, 박성균, "멀티미디어 회의 시스템과 공동 편집기의 구현", 1996

년도 한국정보과학회 가을 학술발표논문집, Vol. 24, No. 1, pp. 1357-1360, 1996. 10. 25-26.

성 미 영

1982년 서울대학교 식품영양학과(학사), 계산통계학과(계산학 전공) 부전공
1987년 프랑스 INSA de Lyon 전산학과(공학석사)
1990년 프랑스 INSA de Lyon 전산학과(공학박사)
1990년~1993년 한국전자통신연구소 컴퓨터연구단 선임연구원
1993년~현재 인천대학교 전자계산학과 조교수
관심분야: 멀티미디어 시스템, 멀티미디어 공동 작업 시스템, 공동 저작/편집 시스템, 문서 구조