초고속 통신망에서 QoS를 보장하는 다자간 실시간 통신에 관한 연구

임 선 화† · 김 문 회††

요 약

통신망이 급진적으로 고속화함으로 인해 응용 범위도 다양해지고 있으며 하나 이상의 통신 스테이션들이 참여하는 다자간 통신을 사용하는 응용 분야들도 증가하고 있다. 본 논문에서는 다중전송을 위한 서버를 두 가지 방식(중앙집중 다중전송 서버 방식과 분산 다중전송 서버 방식)으로 설계하고 구현하였다. 이 두 방식은 모두 세 가지의 동작으로 나뉘어진다: 노드의 참가/삭제 동작, 서버에게 그룹명과 메시지 전송 동작, 그리고 노드가 메시지를 읽고 공유 메모리를 클리어하는 동작. 구현을 토대로 이 두 다중전송 서버 방식을 복잡성, 확장성, 효율성, 실시간성 측면에서 비교하였다. 그 결과로서 메시지를 중앙집중 방식으로 보낼 때보다 분산 방식으로 전송하는 것이 메시지 전송에 대한 평균 전송시간이 짧음을 알 수 있었다. 마지막으로 다중전송 시 QoS를 보장하기 위한 다자간 실시간 통신 방식을 설계하였다.

Study on Multi-party Real-Time Communication with Guaranteed QoS in Information SuperHighway

Sun-Hwa Lim† · Moon-Hae Kim††

ABSTRACT

As communication networks become very fast, the scope of applications is varied and applications using multi-party communications in which more than one communication station participate, have been increased. In this paper, multicast servers are designed and implemented by two different schemes (centralized multicast server scheme and distributed multicast server scheme). Both schemes contain three operations: operation for joining/leaving of a node, operation for transmitting the name of the group and messages to the server, and operation for a node to read messages and clear shared memory. Based on their implementations, two multicast server schemes are compared in terms of complexity, extendibility, efficiency and real-time aspects. As a result, the average transfer time of the distributed multicast server scheme is shorter than that of the centralized multicast server scheme. Finally, we designed the multi-party real-time communication method to guarantee QoS in multicast.

1. 서 론

통신망이 급진적으로 고속화함에 따라 그 응용 범위가 다양해지고 기존의 실시간 분산 처리 시스템 분야에도 많은 도움을 주고 있다. 예를 들면, 영상 회의(video conferencing), 원격 의료 진단(remote medical diagnosis), 무인 자동차 시스템(intelligent vehicle system), 원격 강의(off-campus lecture) 등의 새로운 응용 분야가 나타나고 있다. 이러한 응용 분야에는 통신

※ 이 논문은 1995년도 한국학술진흥재단의 대학부설연구소 연구과제 연구비에 의하여 연구되었음
† 준 회 원 : 건국대학교 대학원 컴퓨터 · 정보통신공학과
†† 정 회 원 : 건국대학교 컴퓨터공학과 교수
논문접수 : 1998년 4월 13일, 심사완료 : 1998년 8월 27일

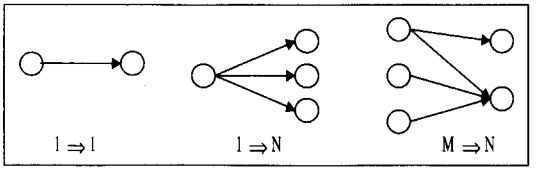
지연시간(communication delay), 손실률(loss rate) 등의 실시간성 및 신뢰성 등의 엄격한 제약이 주어진다. 또한 이들 응용 분야를 살펴보면 특정 다수의 통신 스테이션이 참여하는 다자간 실시간 통신의 특성을 가지고 있으며 이러한 다자간 실시간 통신 기능에 대한 지원은 앞으로의 통신망이 필수적으로 갖추어야 할 것으로 기대된다. 그리고 분산 응용 소프트웨어 요소들 사이에서 대부분 현재 개발된 통신 기술은 일대일 통신으로 원거리 프로시져를 호출하여 사용하였다. 그러나 일대다 또는 다대다 통신을 위한 기술은 거의 지원해 주지 않고 있다[1,3]. 그러므로 본 논문에서는 다자간 실시간 통신 기능을 통신망에 구현할 수 있는 방안들을 고려하고 분석하며 다자간 서버의 기능을 구현한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 프로세서들 간의 통신 방법과 그 중에서 일대다 연결을 위한 다중전송 방식의 필요성과 고려 사항에 관하여 알아보고 제 3장에서는 다중전송 방식을 위한 서버의 설계 및 구현과 두 방식에 대해 비교한다. 제 4장에서는 QoS를 보장하는 다자간 실시간 통신 방식의 설계에 대해 기술한다. 마지막으로 제 5장에서는 결론과 향후 연구 과제에 대하여 기술한다.

2. 다중전송 통신 방식

2.1. 프로세서들 간의 통신 유형

프로세서들 간의 통신 유형으로는 통신 프로세서들 간에 일대일 연결을 위한 단일전송 방식(unicast)이 있으며 이 방식은 원거리 프로시져 호출, 즉 읽기/쓰기(read/write) 동작에 의해 구현되어 진다. 그리고 일대다 연결을 위한 1⇒N 다중전송 방식은 한 노드에서 여러 개의 노드로 메시지를 전송하며 다대다 연결을 위한 M⇒N 다중전송 방식은 여러 개의 노드에서 다른 여러 개의 노드로 메시지를 전송한다. (그림 1)은 이들 전송 방식을 도식화하여 나타낸 것이다.



(그림 1) 프로세서들 간의 통신 유형
(Fig. 1) Communication types of processor-to-processor

2.2 다중전송 방식 통신의 필요성

분산 환경에서 프로세서들 간의 통신 요구의 필요성과 많은 새로운 멀티미디어의 증가로 단일전송 방식은 다자간 통신을 위해서는 비효율적이다. 그리고 M⇒N 다중전송 방식은 너무 복잡하여 실시간 통신에 대한 요구사항을 충족시키기 어렵다. 그러나 많은 프로세서들이 통신에 참여하고 있다면 M⇒N 다중전송 방식이 1⇒N 다중전송 방식보다 효율적일 것이다. 그러므로 상대적으로 프로세서들의 수가 적을 경우에는 오히려 1⇒N 다중전송 방식이 효율적이다. 또한 보다 효율적이기 위해서는 프로세서들이 동적으로 참가(join)와 삭제(leave)를 할 수 있는 기능이 1⇒N 다중전송 방식에서 지원되어야 한다[2,4,5].

2.3 다중전송 방식 통신의 고려 사항

일대일 통신에서는 최단 경로를 따라 메시지를 전송할 경우 메시지의 지연시간을 줄일 수 있다. 그러나 하나 이상의 메시지들이 공유 링크를 사용하여 네트워크를 통해 전송될 경우 트래픽의 부하가 증가하게 된다. 그러므로 시작(source) 노드에서 각각의 목적(destination) 노드로 메시지를 전송할 경우 메시지 지연시간과 네트워크 트래픽의 부하를 가능한 작도록 해야 한다. 그리고 통신을 하는 동안 메시지가 블럭킹(blocking) 되지 않도록 데드락(deadlock)의 상황을 피할 수 있도록 해야 한다[2].

2.4 실시간 통신

초고속 통신망에서의 실시간 통신에 관한 연구는 국내에서는 거의 찾아 볼 수 없고 국외의 경우는 미국의 캘리포니아주 버클리 대학, 미시간 주립 대학, ICSI(International Computer Science Institute), IBM, AT&T Bell Lab 등의 학교와 연구소에서 수행되고 있으며, 인터넷에서의 실시간 통신은 IETF AVT working group에 의해 1992년부터 실시간 전송 프로토콜(RTP)과 실시간 전송제어 프로토콜(RTCP)이 개발되고 있다[6,7]. 이 중 인터넷에서의 실시간 통신을 지원하기 위한 RTP와 RTCP는 현재 가장 많이 쓰이는 것으로 인터넷 상에서 실시간 응용을 지원하는 장점이 있지만 QoS에 대한 보장을 할 수 없는 단점이 있다. QoS를 보장하기 위해 메시지 지연시간(delay)과 메시지들 간의 지연시간 편차(jitter)에 대한 방법이 있는데 이 두 방법은 약간의 차이가 있지만 메시지의 지연시

간에 대한 한계를 극복하여 실시간적으로 통신을 하도록 하기 위한 것으로 현재 이들에 대한 연구가 계속 진행 중에 있다. QoS를 보장하는 다자간 실시간 통신은 두 단계로 나누어 지는데 다음과 같다.

* 자원 할당 단계 : 승인 제어와 자원 예약

다자간 통신 응용에서 종단점 간의 QoS 요구사항을 보장하기 위해 목적지까지 충분한 자원이 있는 가를 결정하는 자원 할당 단계가 필요하다. 만약 QoS 요구사항이 만족된다면 다자간 통신 응용은 승인되고 이러한 요구사항을 만족하기 위해 자원 할당을 예약한다. 그리고 예약 프로토콜에서 예약 설정 방향(reservation direction)에 따라 시작-초기 예약(sender-initiated reservation)과 수신-초기 예약(receiver-initiated reservation)이 있다[8]. 시작-초기 예약은 시작쪽에서 QoS 요구사항을 기술한 후 그것을 목적지로 보내고 시작에서 목적지까지의 경로를 따라 자원이 예약되는 방법이다. 수신-초기 예약은 위의 방법과 반대이다. 일반적으로 다중전송 방식은 시작에서 여러 개의 목적노드로 멀티미디어 데이터가 전송됨으로 시작-초기 예약 프로토콜 방식이 적합하다고 생각되며 시작-초기 자원 예약과 할당 프로토콜의 흐름은 (그림 2)와 같다.

그리고 자원 예약을 위한 접근 방법[11]으로는 종단점 간의 전체 QoS 요구사항을 시작에서 목적지까지 연결된 경로 상에서 각 링크에서 필요로 하는 지역(local) QoS로 나눌 수 있다. 즉, 세 노드 사이에 인접한 두 링크(*l1*, *l2*)가 있을 경우 종단점 간의 QoS는 Q_{l1}과 Q_{l2}의 합으로 나타낼 수 있다.

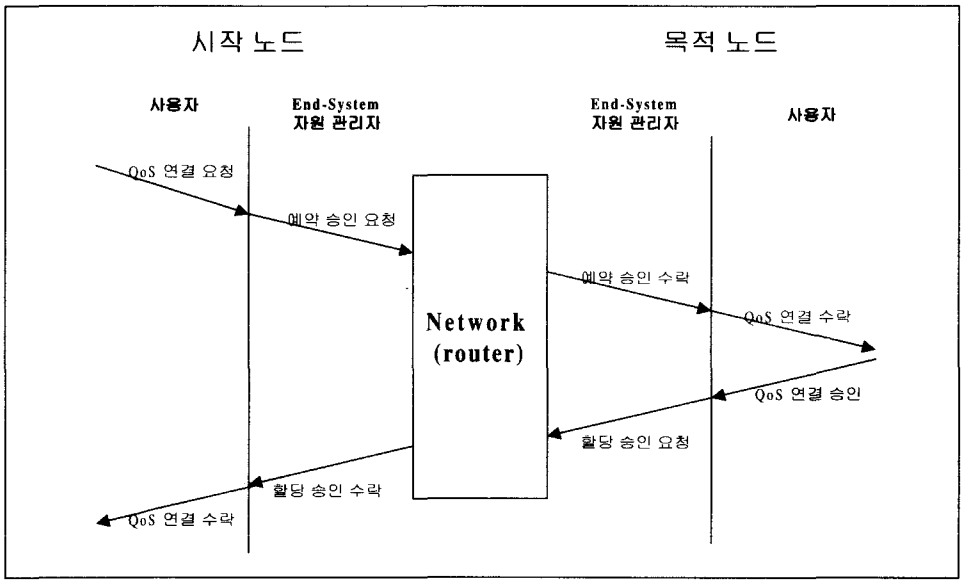
$$A \xrightarrow{l1} B \xrightarrow{l2} C$$

$$Q(A, C) \approx (Q_1(A, B) + Q_2(B, C) = Q_{l1} + Q_{l2}$$

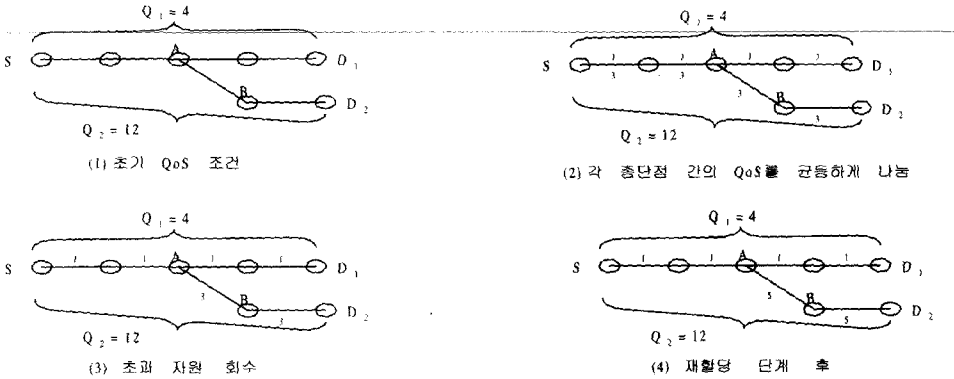
이 방법에는 QoS 요구사항을 링크들 간에 균등하게 나누는 균등 나눔 방법과 네트워크 상에서 로드에 따라 자원을 다르게 나누는 비균등 나눔 방법이 있다.

* 자원 재할당 단계 : 초과로 할당된 자원 해제

멀티캐스트 트리에서 지나치게 할당된 자원은 트래픽이 발생하지 않도록 하면서 자원을 반환해야 한다. 즉 서로 다른 목적지들이 같은 경로를 공유하면서 자원의 양이 다른 QoS를 요구할 경우 멀티캐스트 트리 내에 할당된 자원의 일부를 회수하여 자원을 다시 할당한다. (그림 3)은 우선 종단점 간의 QoS 요구사항을 링크들 간에 각각 균등하게 나눈 후 공유 링크에는 최소한의 QoS를 할당하여 각 링크의 QoS를 재할당한 예를 보여주고 있다. (그림 3)에서의 Q_i는 메시지 지연 시간에 대한 QoS를 의미한다.



(그림 2) 시작-초기 자원 예약과 할당 프로토콜 흐름
(Fig. 2) Sender-initiated resource reservation and allocation protocol flow



(그림 3) 자원 재할당의 예
(Fig. 3) An example of resource reallocation

3. 다중전송을 위한 서버의 설계 및 구현

다중전송을 하기 위한 방식으로 중앙집중 다중전송 서버 방식과 분산 다중전송 서버 방식을 설계하였다. 중앙집중 다중전송 서버 방식은 시스템 전체 데이터의 전송과 목적 노드들을 관리하는 서버와 간단히 데이터 전송과 수신만을 수행하는 노드들로 구성된다. 이 방식은 시스템에서 서버의 역할이 큰 비중을 차지하기 때문에 서버에 오류가 발생할 경우 전체 시스템이 동작하지 못하게 된다. 이러한 문제점을 해결하기 위해 서버를 모든 기계에 분산시켜 하나의 시스템에 대한 과부하를 줄일 수 있는 분산 다중전송 서버 방식을 설계하였다.

3.1 중앙집중 다중전송 서버 방식

중앙집중 다중전송 서버 방식에서는 서버와 데몬 노드가 있으며 많은 목적 노드들이 논리적 그룹을 가지고서 다중전송 통신을 수행하고 있다. 서버는 모든 목적 노드들의 참가/삭제 동작과 메시지 전송을 관리하기 위해 서버 테이블을 가지고 있으며 서버 테이블의 데이터 구조는 다음과 같다.

```
typedef struct Server_Table {
    char *machine; /* 기계 이름 */
    char *group; /* 그룹 이름 */
    int destnum; /* 목적 노드 수 */
}S_table;
```

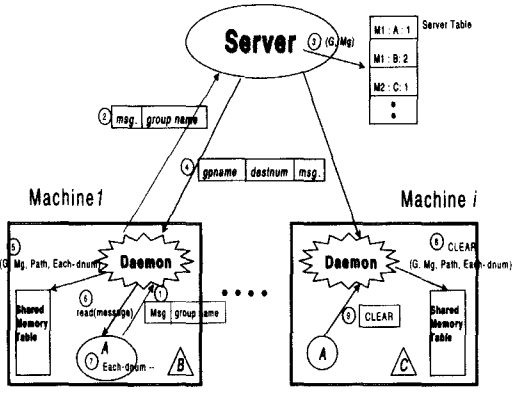
그리고 각 기계마다 있는 데몬 노드는 서버가 보낸 그룹과 동일한 그룹을 갖는 목적 노드들이 새로운 메시지를 읽을 수 있도록 하기 위하여 공유 메모리 레코드를 관리하며 데이터 구조는 다음과 같다.

```
typedef struct record_type {
    char *gp; /*그룹 이름 */
    char *mesg; /* 메시지 이름 */
    char *path; /* 메시지가 들어 있는 경로 */
    int each-dnum; /* 메시지를 받을 목적 노드의 수 */
}recordtype;
```

다음은 각 노드들의 관계를 나타낸 것이다.

- 시작 노드와 데몬과의 관계
  임의의 시작 노드는 자신이 속해 있는 데몬에게 메시지 파일과 그룹 이름을 전송한다. 데몬은 시작 노드로부터 받은 위의 정보를 서버에게 전달하고 서버는 메시지 파일과 그룹 이름을 저장한다.
- 서버와 데몬과의 관계
  서버는 데몬으로부터 받은 그룹 이름과 서버 테이블에 있는 그룹 이름이 같고, 이 그룹에 있는 목적 노드의 수가 0보다 큰 각 데몬에게 그룹 이름, 목적 노드 수, 메시지 파일를 전송한다. 데몬은 자신이 가지고 있는 공유 메모리로 구성된 데몬 레코드에 서버로부터 받은 그룹 이름, 목적 노드 수, 메시지 파일을 저장한다.
- 데몬과 목적 노드와의 관계
  목적 노드는 똑같은 메시지 파일을 받지 않기 위

해 기존에 있는 메시지 파일의 이름과 데몬의 공유 메모리 레코드에 저장되어 있는 새로운 메시지 파일의 이름이 같은 지를 비교한다. 만약 메시지 파일 이름이 같지 않고 공유 메모리 레코드에 있는 그룹 이름이 자신의 그룹 이름과 동일하다면 데몬에 저장되어 있는 메시지 파일의 내용들을 읽은 후 목적 노드의 수를 감소시킨다. 메시지를 마지막으로 읽은 목적 노드는 목적 노드의 수를 감소시켜 만약 목적 노드의 수가 0이 되면 이 목적 노드는 데몬에게 공유 메모리 레코드를 클리어 하도록 CLEAR 메시지를 전송한다. 데몬은 목적 노드로부터 CLEAR 메시지를 전송받고 목적 노드의 수가 0인 것을 확인한 후 공유 메모리 레코드를 클리어하고 다시 서버로부터 새로운 메시지 파일을 전송 받는다.



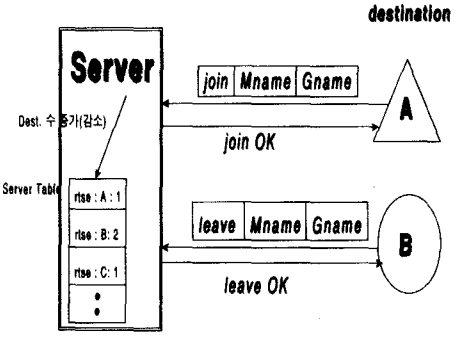
(그림 4) 중앙집중 방식에서의 메시지 전송 동작
(Fig. 4) Operation for transmitting messages in the centralized scheme

(그림 4)는 위의 동작들을 나타낸 것이며 단계별로 기술하였다.

① 임의의 시작 노드는 메시지 파일과 그룹 이름을 데몬에게 전송
② 데몬은 서버에게 메시지 파일과 그룹 이름을 전송
③ 서버는 메시지와 그룹 이름을 저장
④ 서버는 각 기계에 있는 데몬에게 그룹 이름, 목적 노드 수, 메시지를 전송
⑤ 데몬은 서버로부터 받은 것들을 공유 메모리 레코드에 저장
⑥ 목적 노드는 공유 메모리 레코드에 있는 그룹이 자신의 그룹과 같으면 메시지를 읽음
⑦ 목적 노드는 메시지를 읽은 후 목적 노드 수 감소
⑧ 마지막 목적 노드는 데몬에게 CLEAR 메시지 전송
⑨ 데몬은 공유 메모리 레코드를 클리어

- 서버와 목적 노드와의 관계

목적 노드는 참가와 삭제를 동적으로 하기 위해 서버에게 자신이 속해 있는 기계 이름과 그룹 이름을 전송한다. 서버는 목적 노드로부터 받은 기계 이름과 그룹 이름을 서버 테이블에 저장되어 있는 것과 비교하여 일치할 경우 참가일 때는 목적 노드 수를 증가시키고 삭제일 때는 목적 노드 수를 감소시킨다. 서버는 참가와 삭제 동작이 올바르게 수행되어졌으면 'JOIN OK' 또는 'LEAVE OK' 신호를 목적 노드에게 전송한다. 다음 (그림 5)는 목적 노드의 참가와 삭제에 대한 동작을 나타낸 것이다.



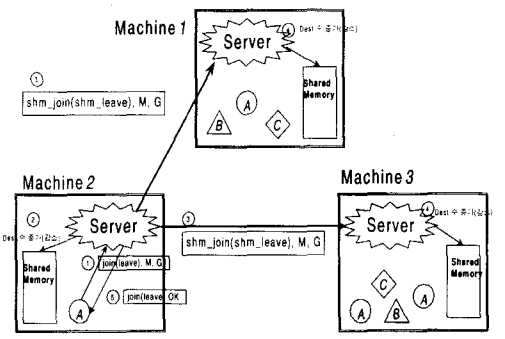
(그림 5) 중앙집중 방식에서의 참가/삭제 동작
(Fig. 5) Operation for joining/leaving of a node in the centralized scheme

3.2 분산 다중전송 서버 방식

분산 다중전송 서버에서는 각각의 물리적 기계마다 서버가 있어 중앙집중 다중전송 서버의 구조에서 전체적인 서버의 역할과 각 기계마다 있는 데몬의 역할을 동시에 수행한다. 그리고 서버마다 각각의 공유 메모리 테이블을 가지고 있으며 이 테이블에는 목적 노드들에 대한 정보들이 저장되어있다. 또한 목적 노드들이 새로운 메시지를 읽을 수 있도록 하기 위하여 공유 메모리 레코드를 관리한다. 분산 다중전송 서버 방식의 기능은 크게 세 가지로 나누어진다.

● 목적 노드의 참가와 삭제를 위한 동작

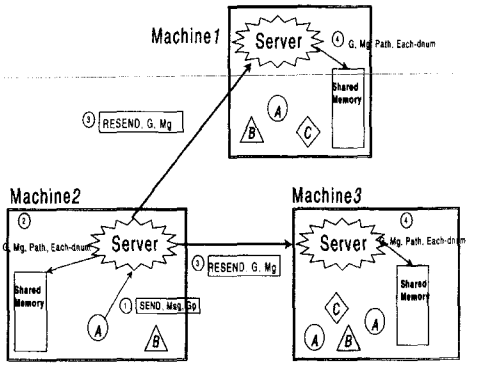
목적 노드는 참가와 삭제를 하기 위해 서버에게 기계 이름과 그룹 이름을 전송한다. 서버는 목적 노드로부터 받은 기계 이름과 그룹 이름을 서버가 가지고 있는 공유 메모리 테이블에 저장되어 있는 것과 비교한다. 만약 일치할 경우 참가일 때는 목적 노드 수를 증가시키고 삭제일 때는 목적 노드 수를 감소시킨다. 그리고 다른 서버에 있는 공유 메모리 테이블의 정보들과 자신의 공유 메모리 테이블의 정보들을 일치시키기 위해 목적 노드의 참가와 삭제에 관한 정보를 다른 서버에게 알려 각각 기계마다 가지고 있는 공유 메모리 테이블의 정보들을 일치시키도록 한다. (그림 6)는 이 과정을 보여주고 있다.



(그림 6) 분산 방식에서의 참가/삭제 동작
(Fig. 6) Operation for joining/leaving of a node in the distributed scheme

● 서버에게 그룹과 메시지 전송을 위한 동작

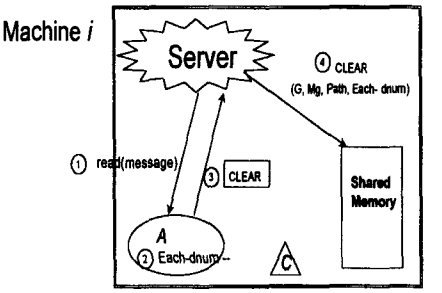
임의의 시작 노드는 서버에게 메시지와 그룹 이름을 전송한다. 만약 두개 이상의 시작 노드가 서버에게 메시지를 전송하면 서버는 먼저 도착한 시작 노드의 메시지와 그룹 이름을 큐에 저장하기 위해 세마포어에서 락(lock)을 건다. 저장이 끝나면 다른 시작 노드가 메시지와 그룹 이름을 저장하도록 락(lock)을 푼다. 서버는 시작 노드로부터 받은 그룹 이름과 공유 메모리 테이블에서 일치하는 그룹 이름을 찾아 이 그룹에 속해 있는 목적 노드의 수가 0보다 큰 값을 가지고 있는 다른 서버가 있다면 이 서버에게 그룹 이름과 메시지를 전송한다. (그림 7)은 메시지 전송 과정을 보여준다.



(그림 7) 분산 방식에서의 메시지 전송 동작
(Fig. 7) Operation for transmitting messages in the distributed scheme

● 공유 메모리 클리어 동작

(그림 8)에서 보여주듯이 목적 노드는 새로운 메시지를 서버로부터 읽고 난후 목적 노드의 수를 감소시킨다. 만약 목적 노드의 수가 0이 되면 서버에게 공유 메모리를 클리어 하도록 CLEAR 메시지를 전송한다. 서버는 이 메시지를 받은 후 공유 메모리를 클리어하고, 다시 서버로부터 새로운 메시지를 전송 받는다.



(그림 8) 읽기 · 클리어 동작
(Fig. 8) Operation for a node to read messages and clear shared memory

3.3 다중전송 방식을 위한 서버의 구현

위에 언급한 두 방식의 다중전송 서버가 구현되었다. 이 두 방식의 시스템 구현 환경은 Sparc Station 5와 Sparc Station 20 workstation이며 운영 체제로는 SunOS Release 4.1.3을 사용한다. Workstation 상에서 통신을 위한 프로토콜은 TCP/IP을 사용하였고 소켓(socket)을 사용하여 프로세스들 간의 통신을 하며 메

모리를 효율적으로 사용하기 위하여 공유 메모리(shared memory) 방식을 채택하였다. 또한 공유 메모리에 메시지가 중복 저장되는 것을 막기 위해 세마포어 기법을 사용하였다[9,10]. 먼저 중앙집중 방식에서는 하나의 서버를 실행시킨 후 각각의 데몬들을 실행시키고 목적 노드들이 참가를 하기 위해 참가 메시지를 서버에게 전송하게 되면 목적 노드는 실행이 시작된다. 목적 노드들 중 특정 그룹에게 메시지를 전송하고자 하는 노드(시작 노드)는 데몬에게 메시지를 전송한다. 데몬은 서버에게 전송하고 서버는 특정 그룹을 가지고 있는 다른 데몬에게 메시지를 전송한다. 목적 노드는 이 메시지를 읽게 된다. 그리고 목적 노드가 그룹에서 삭제를 원할 경우 삭제 메시지를 서버에게 전송하게 되고 목적 노드의 실행은 끝나게 된다. 분산 방식에서는 각 기계에 있는 서버를 실행시킨다. 목적 노드가 서버에게 참가 메시지를 전송하게 되면 서버는 다른 서버에게도 목적 노드의 참가를 알리고 목적 노드의 실행이 시작된다. 그리고 한 노드(시작 노드)가 특정 그룹에게 메시지를 전송하면 서버는 이 그룹을 갖는 각 서버에게 메시지를 전송하고 목적 노드는 메시지를 읽게 된다. 그리고 목적 노드가 그룹에서 삭제를 원할 경우 삭제 메시지를 서버에게 전송하고 서버는 다른 각 서버에게 전송한 후 목적 노드의 실행은 끝나게 된다.

3.4 중앙집중 방식과 분산 방식의 비교

다음과 같이 네 가지 측면에서 두 방식을 비교할 수 있으며 목적 노드를 확장할 경우 두 가지 방식에 대한 복잡성과 확장성에 관한 시간적인 관계는 (그림 9)와 같다.

- 복잡성

  중앙집중 방식이 분산 방식보다 단순하다. 중앙집중 서버에서는 목적 노드들에 대한 정보가 한 서버에 의해서만 관리되나 분산 방식에서는 목적 노드들에 대한 정보가 여러 서버들에게 복제됨으로 이들에 대한 무결성을 보장해야 한다.

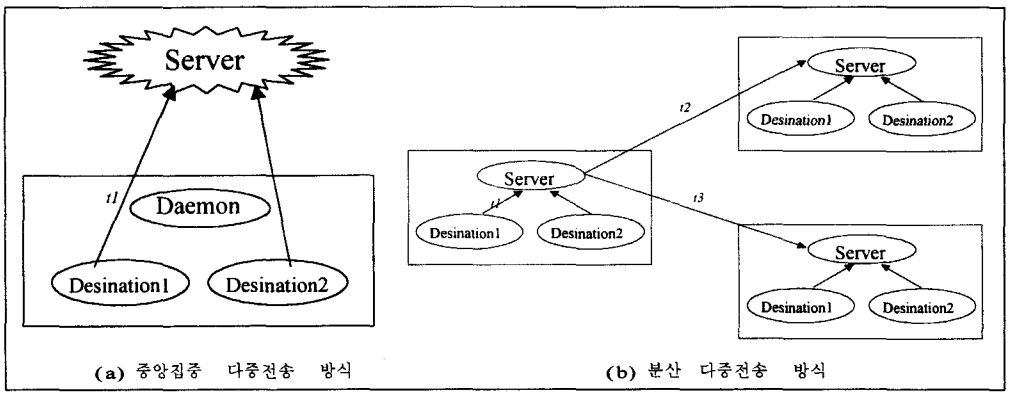
- 확장성

  두 방식이 확장성 측면에서는 비슷하나 분산 방식에서는 추가되는 노드의 서버가 다른 기존의 서버들에게 통지를 하여야 하므로 중앙집중 방식이 다소 단순하게 확장될 수 있다. 그리고 (그림 9)에서 처럼 노드를 추가할 경우 중앙집중 방식이 분산 방식보다 시간이 적게 걸림을 알 수 있다.

- 효율성

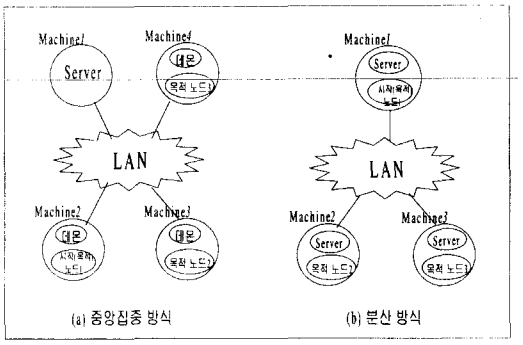
  다음은 중앙집중과 분산 다중전송 서버 방식을 각각 구현한 후 메시지 전송시간에 대한 효율성을 비교하였다. (그림 10)은 두 방식에 대한 실험 환경을 나타낸 것으로 Machine1에 있는 시작 노드는 목적 노드의 역할도 수행한다.

<표 1>은 중앙집중 방식과 분산 방식에서 메시지를 10개, 50개, 100개를 연속적으로 보내어 Machine1에 있는 시작 노드에서 Machine1, Machine2, Machine3에 있는 목적 노드까지 메시지가 도착한 시간을 구한 것이며 각 기계마다 전송시간에는 약간의 차이가 있었다.



(그림 9) 중앙집중과 분산방식에서 목적노드를 확장할 경우 걸리는 시간: $t1 < t1 + t2 + t3$

(Fig. 9) Transfer time in case of extending a destination in centralized and distributed methods : $t1 < t1 + t2 + t3$



(그림 10) 실험 환경
(Fig. 10) Experimentation environment

〈표 1〉 전송시간
〈Table 1〉 Transfer time

메시지 수 \ 구간 (second)		시작노드 목적노드1	시작노드 목적노드2	시작노드 목적노드3
10개	중앙	1	2	2
	분산	2	3	4
50개	중앙	37	37	37
	분산	14	30	30
100개	중앙	87	87	88
	분산	55	75	75

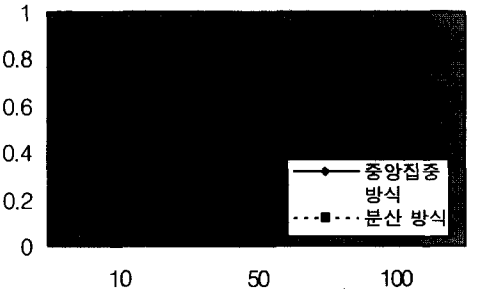
메시지 수를 n이라 하고, 각 메시지마다 시작 노드에서 목적 노드까지의 도착시간을 λ라고 하면 하나의 메시지에 대한 평균 전송시간은 다음 식 (1)과 같이 나타낼 수 있다.

$\lambda_{(n=10,\ 50\ 또는\ 100)} =$
time (시작노드→목적노드) / 메시지갯수 $_{(n=10,\ 50\ 또는\ 100)}$ (1)

그리고 시작 노드에서 모든 목적 노드까지 하나의 메시지에 대한 전체 평균 도착시간은 다음 식 (2)와 같이 나타낼 수 있다.

$$\lambda = \lambda_{10} + \lambda_{50} + \lambda_{100} \ / \ 3 \qquad (2)$$

(그림 11)에서 시작 노드에서 목적 노드까지의 전체 평균 전송시간을 분석해 본 결과, 처음 메시지를 10개 보내었을 때, 중앙집중 방식이 분산 방식보다 메시지 전송시간이 짧음을 알 수 있었다. 그러나 메시지 수를 증가시킬수록 점점 분산 방식이 중앙집중 방식보다 전송시간이 작아졌다. 그 이유는 첫번째, 중앙집중 방식에서는 모든 메시지가 서버를 통해 전달되므로 그만큼 서버의 부하가 증가하게 되어 메시지 전송시간이 길어지게 되지만, 분산 방식에서는 서버의 부하를 분산시키기 때문에 성능 저하를 막을 수가 있었다. 그리고 두번째, 중앙집중 방식은 시작 노드에서 중간 노드인 서버를 거쳐서 데몬에 도착한 메시지를 목적 노드에 전송하지만, 분산 방식에서는 시작 노드를 갖는 서버가 다른 서버에게 직접 메시지를 전송하기 때문에 중앙집중 방식으로 보낼 때보다 분산 방식으로 전송하는 것이 평균 전송시간을 짧게 하였다. 이를 통해 메시지의 전송량이 적을 경우에는 중앙집중 방식이 더 효율적이지만 많은 메시지를 전송할 경우에는 분산 방식이 더 효율적이라는 것을 알 수 있었다. 물론 이 실험에서는 메시지를 10, 50, 100개 정도로서 시뮬레이션을 하였지만 컴퓨터 수의 증가와 많은 정보를 주고 받는 현재로서는 중앙집중 방식보다 분산 방식이 더 효율적일 것이다.



(그림 11) 시작노드에서 목적노드까지의 메세지 평균 전송시간
(Fig. 11) Messages average transfer time from source to destination

- 실시간성

  실시간 통신에 대한 QoS 파라메터에 대한 관리는 중앙집중 방식이 편리하나 이들에 대한 보장에 관한 사항은 보다 연구가 필요하다. 이는 자원 할당 방식과도 밀접한 관계가 있으며 현재로는 효율성이 높은 분산 방식이 실시간성 보장면에서 보다 유리할 것으로 판단된다.

4. QoS를 보장하는 다자간 실시간 통신 방식 설계

지금까지 중앙집중 다중전송 서버와 분산 다중전송 서버를 설계하고 구현해 보았다. 그러나 현재 운영체제의 환경에서 의미상으로는 다중전송 방식이지만 소켓을 사용하여 메시지를 전송하기 때문에 서버 간에 단일전송 방식으로 구현할 수 밖에 없었다. 이것은 엄밀히 따지면 다중전송 방식은 아니다. 다중전송 방식을 구현하기 위해서는 IP 다중전송을 사용해야 한다. 그리고 오디오나 비디오를 포함한 많은 다자간 통신 응용은 네트워크 상에서 종단점 간의 지연 또는 손실률에 대한 QoS 보장을 요구하며, 이를 위해서는 자원 예약을 위한 채널 설정(channel establishment)이 필요하다[11]. 이와 같은 관점에서 QoS를 보장하는 다자간 실시간 통신 방식을 설계하였다. 이 방식은 (그림 12)와 같이 크게 GRM(Global Resource Manager), Router Server, LRM(Local Resource Manager) 그리고 목적 노드들로 구성되며, LRM과 목적 노드들은 각각 LAN으로 구성되어 있다. GRM은 모든 목적 노드에 대한 채널의 상태를 관리하며, LRM은 LAN에 속해 있는 목적 노드에 대해서만 채널의 상태를 관리한다.

4.1 자원 예약을 위한 채널 설정과 데이터 전송 동작

다음은 각 노드들이 하는 역할을 (그림 12)의 번호와 연계하여 기술하였다.

● 시작 노드가 하는 일

(1) : 시작 노드는 채널 설정 시에 각 노드의 자원 예약을 요구하기 위해 다음과 같은 정보를 GRM에게 전송한다.
- 전송될 패킷을 저장할 수 있는 버퍼와 성능을 산출하기 위한 시작 노드로부터 목적 노드까지의 delay bound, jitter bound
- 해당 그룹에게 패킷을 전송하기 위해 그룹명

(15)~(18) : 마지막으로 시작 노드는 Multicast Server로부터 채널 설정 완료에 대한 메시지를 받은 후 패킷을 전송하게 된다.

● GRM에서 하는 일

(2)~(3) : GRM은 시작 노드가 보낸 그룹명을 Group Manager에게 전송하고 Group Manager는 이 그룹에 해당하는 목적 노드의 리스트를 GRM에게 전송한다.

(4)~(6) : 마찬가지로 Router Server에게 그룹명을 알려준 후 Router Server로부터 이 그룹에 해당되는 목적 노드로의 경로에 관한 정보를 받는다.

(7) : GRM은 Group Manager와 Router Server한테서 받은 정보를 각 LRM에게 전송한다.

(13) : GRM은 LCS(Local Channel State)의 정보에 따라 GCS(Global Channel State)를 변경하는데, 이 GCS는 각 LAN에 있는 채널의 전체 상태를 관리한다.

(14) : GCS는 시작 노드가 보낸 그룹과 일치하면서 채널이 설정된 목적 노드로의 경로를 Multicast Server에게 전송한다.

● 목적 노드가 하는 일

(8)~(9) : 이 그룹에 해당하는 목적 노드만 시작 노드가 보낸 버퍼 크기를 테스트한다. 그리고 시작 노드가 보낸 패킷 크기와 같은 크기의 패킷을 경로에 따라 각 목적 노드에게 전송하여 ping time을 구한 후, 이 ping time을 반으로 나눈 값과 delay bound 값이 delay jitter 값의 범위를 만족하는 지를 테스트한다. 만약 목적 노드가 이 요구사항들을 만족한다면 LRM에게 'channel establishment accept' 메시지를 전송하고, 그렇지 않으면 'channel establishment reject' 메시지를 전송한다.

● LRM에서 하는 일

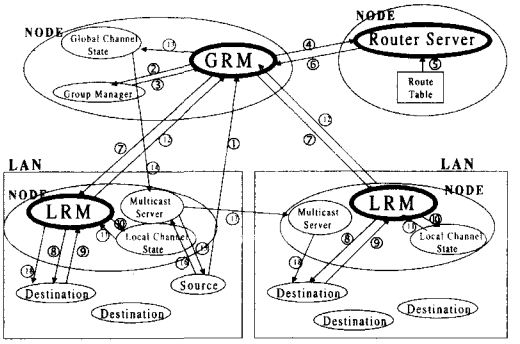
(10)~(11) : LCS는 처음에 각 목적 노드로의 경로에 대한 채널 값이 FALSE로 설정되어 있다가, LRM이 목적 노드로부터 'channel establishment accept' 메시지를 받게 되면 LCS는 이 목적 노드로의 경로에 대해 채널 값을 TRUE로 변경한다.

(12) : 각 LRM은 자신에게 속해 있는 목적 노드들의 채널 설정 상태가 완료되면 GRM에게 LCS의 채널 상태 정보를 전송한다.

4.2 목적 노드의 참가/삭제 동작

목적 노드가 참가 또는 삭제를 할 경우 LAN ID, GROUP NAME, IP ADDRESS를 Group Manager에게 전달하고 LAN ID, GROUP NAME, PATH를 Router

Server에게 전달한다. 그러면 Group Manager와 Router Server는 자신이 관리하고 있는 테이블을 변경하게 된다. 여기에서 Route Table의 경로는 정적으로 할당되어있다.



(그림 12) 채널 설정 절차와 데이터 전송
(Fig. 12) Channel establishment procedure and data transmission

Group Manager와 Router Server가 관리하는 테이블의 구조는 <표 2>와 <표 3>과 같다.

〈표 2〉 그룹 테이블
〈Table 2〉 Group Table

LAN ID	LRM IP ADDRESS	GROUP NAME	GROUP IP ADDRESS
1	127.1.2.1	A	128.1.1.1
			128.1.1.1
		B	129.1.1.2
		⋮	⋮
2	127.1.2.1	A	128.1.1.1
			128.1.1.1
		⋮	⋮

〈표 3〉 라우트 테이블
〈Table 3〉 Route Table

LAN ID	GROUP NAME	PATH
1	A	Dest.1 → Multicast Server → Neighbor MSDest.1
		Dest.1 → Multicast Server → Neighbor MSDest.2
	B	Dest.1 → Multicast Server → Neighbor MSDest.3
	⋮	⋮
2	A	Dest.1 → Multicast Server → Neighbor MSDest.1
		Dest.1 → Multicast Server → Neighbor MSDest.2
	⋮	⋮

여기에서 제안한 시스템 설계에 대한 장점은 다음과 같다. 첫번째, GRM과 LRM을 따로 두어, GRM이 모든 목적 노드에 대한 채널 설정을 관리하는 것보다 각 LAN에 있는 LRM이 자신에게 속해 있는 목적 노드에 대해서만 채널 설정을 관리하기 때문에 그만큼 GRM의 과부하를 줄일 수 있다. 두번째, 일반적으로 채널 설정을 할 경우, 목적 노드 또는 중간 노드에서 channel establishment reject가 발생하면 시작 노드까지 'channel establishment reject' 메시지가 역방향으로 전달되고, 뿐만 아니라 'channel establishment accept' 메시지도 역방향으로 전달된다. 그러나 여기에서는 이미 LRM에서 channel establishment accept된 목적 노드의 경로만 시작 노드에게 보내지기 때문에 그만큼 채널 설정시간과 자원 예약시간을 줄일 수 있다. 세번째, 일반적으로 다중전송 방식은 모든 목적 노드에게 패킷을 보내면 목적 노드는 자신이 속한 그룹과 일치하는 그룹이 아니면 패킷을 받지 않는다. 그러나 여기에서는 모든 목적 노드에게 패킷을 전송하는 것이 아니라 채널이 설정된 목적 노드에게만 패킷을 전송하기 때문에 네트워크 트래픽의 부하를 줄일 수 있다. 네번째, 패킷을 전송하기 전에 이미 자원 예약을 위한 채널 설정을 함으로써 QoS를 보장할 수 있으며, 또 충분한 버퍼가 이미 설정 되어 있기 때문에 패킷 블록킹(blocking)의 발생으로 인한 데드락(deadlock)을 줄일 수 있다.

5. 결 론

통신망이 급진적으로 고속화하고 다자간 통신 응용분야들은 특정 다수의 통신 스테이션이 참여하는 다중전송 통신의 특성을 가지고 있으며 이러한 응용들은 실시간성 및 신뢰성의 엄격한 제약을 받는다. 그러므로 이러한 요구사항에 맞는 다중전송 실시간 통신 기능의 특성을 갖는 통신망이 갖추어져야 한다.

본 논문에서는 다자간 통신을 위한 다중전송 서버를 구축하였다. 첫번째 방법에서는 하나의 서버가 모든 목적 노드의 정보를 관리하고 메시지를 전송하는 중앙집중 다중전송 서버 방식을 구현하였으며, 두번째 방법에서는 시스템의 부하를 줄이기 위해 분산 환경하에서 메시지 전송을 더욱 효율적으로 할 수 있는 분산 다중전송 서버를 구현하였다. 그리고 이 두 방식을 복잡성, 확장성, 효율성, 실시간성 측면에서 비교하였

다. 그 결과로서 노드를 추가할 경우 시간적으로 볼 때 복잡성과 확장성은 중앙집중 방식이 분산 방식보다 효과적이었다. 그러나 메시지 평균 전송시간에 대한 것은 중앙집중 방식으로 보낼 때보다 분산 방식으로 전송하는 것이 평균 전송시간이 작음을 알 수 있었다. 그리고 마지막으로 다중전송 시 종단점 간의 지연 또는 손실률에 대한 QoS를 보장하기 위해 자원 예약을 위한 채널 설정(channel establishment)을 함으로써 QoS를 보장하는 다자간 실시간 통신 방식을 설계하였다.

향후 계획으로는 이 논문에서 설계한 QoS를 보장하는 다자간 실시간 통신 방식을 구현하고자 한다. 그리고 각 목적 노드들의 경로가 정적으로 할당되어 있는데, 이것은 목적 노드들의 참가와 삭제가 빈번한 네트워크 환경에서는 비효율적이다. 그러므로 네트워크 환경에 따라 동적으로 목적 노드를 할당하여 라우트 테이블을 관리할 수 있는 방안에 관한 연구와 QoS를 보장하기 위한 메시지 지연시간과 메시지들 간의 지연시간 편차에 대한 방법들에 관한 연구를 계속 하고자 한다.

참 고 문 헌

[1] D. Ferrari et al., "Network Support for Multimedia," Tech Report TR-92-072, ICSI, November 1992.

[2] D. Dandlur et al., "Real-Time Communication in Multihop Networks," IEEE Trans. on PADS, Vol.5, No.10, October 1994.

[3] Chen Chen, "Selective Multicast Communication in Distributed Systems," Tech. report, Univ. of Maryland, October 1992.

[4] M. Schollmeyer, "Multicast Routing in Unreliable Networks," Tech. report, Univ. of Missouri at Rolla, July 1991.

[5] H.Kanakia et al., "An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport," Proc. ACM SIGCOMM'93, pp.20-31, 1993.

[6] "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, INTERNET DRAFT, March 1995.

[7] "RTP Profile for Audio and Video Conference with Minimal Control," Internet Engineering Task Force, INTERNET DRAFT, March 1995.

[8] K. Nahrstedt and Ralf Steinmetz, "Resource Management in Networked Multimedia Systems," IEEE Computer, Vol.28, No.5, pp.52-63, May 1995.

[9] W.R. Stevens, "Unix Network Programming," Prentice Hall, Englewood Cliffs, 1991.

[10] W.R. Stevens, "Advanced Programming in the UNIX Environment," Addison-Wesley, Reading, 1992.

[11] Victor Firoiu and Don Towsley, "Call Admission and Resource Reservation for Multicaast Sessions," Proc. IEEE INFOCOM '96, pp.94-101, 1996.

임 선 화

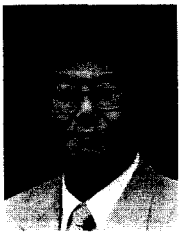
shlim@pluto.konkuk.ac.kr

1994년 2월 부산여자대학교 전자계산학과(학사)

1996년 8월 건국대학교 전자계산학과(석사)

1996년 9월~현재 건국대학교 컴퓨터·정보통신공학과 박사과정

관심분야 : 분산 실시간 통신, 통신망 관리, 소프트웨어 공학



김 문 회

mhkim@pluto.konkuk.ac.kr

1979년 2월 서울대학교 전기공학과(학사)

1985년 5월 University of South Florida, MSCS

1991년 5월 University of California, Berkeley, Ph.D

1991년 3월~현재 건국대학교 컴퓨터공학과 부교수

관심분야 : 실시간 분산처리 시스템, 통신망 관리, 운영체제, 소프트웨어공학