메타 그룹을 이용한 신뢰성 있는 멀티캐스트 프로토콜의 설계 및 분석

정회원 이 동 춘*, 김 배 현*, 송 주 석**

Design and Analysis of a Reliable Multicast Protocol using Meta-Groups

DongChun Lee*, BaeHyun Kim*, Jooeok Song** *Regular Members*

요 약

본 연구는 전파트리 프로토콜을 기반으로 하여 중첩된 그룹의 멤버를 고려한 메타-그룹(MG)을 노드로 하여 멀티캐스트 트리를 구성하여 트리의 깊이가 작아지고, 순서화를 이루기가 쉬워 지게 하며, 메타-그룹에 대표 수신자의 역할을 하는 Designated Manager(DM)을 두어 송신자를 대신하여 DM이 속한 메타-그룹의 멤버들에 대한 ACK 처리와 재전송을 담당하는 멀티캐스트 프로토콜을 설계한다. 시뮬레이션 분석을 통해서 송신자의 ACK 처리를 분산시키고, commit delay를 줄이고 송신자의 overhead를 줄일 수 있어서, 기존의 프로토콜보다 우수한 성능을 보였다.

ABSTRACT

In this paper, we propose a protocol that makes use of a concept of a meta-group based on propagation trees to deal with duplicated members of the same multicast group. It is shown that, if multicast tree is composed of these meta-groups, the depth of the tree can be shortened and the ordering of the muticast that communicates between multiple senders and receivers can be easier. In the protocol, we assign a Designated Manager(DM) to each meta-group and make each DM do the role of the representative receiver of the meta group. In this Paper, the DM's are supposed to handle ACK and retransmission for the members in the same meta group. Hence, the DM's distribute the ACK from senders, and they can reduce the burden of senders by shortening commit delay time. We also show, through a simulation analysis, that the new multicast protocol outperforms the existing ones not only in message costs but also in commit delay times.

I. 서 론

최근의 통신환경을 지원하기 위해서는 기존의 한 수신자와 일대일로 통신하는 것뿐만 아니라, 오디오나 비디오 분배 서비스, 화상 회의, 분산 데이터베이스 응용 등을 지원할 수 있는 멀티캐스트 통신이 급격히 요구된다. 멀티캐스트는 프로세스의 그룹으로 메시지가 신뢰성을 갖고 빠르게 전송될 수 있어야 한다. 멀티캐스트 프로토콜은 분산과 효율성이 중요한 문제가 된다. 신뢰성은 송신자가 보낸 메시지를 모든 수신자들이 정확히 수신하는 것이다. 신뢰성에 대한 정의는 구현되는 환경에 따라 다르게 정의되지만, 일반적으로 다음과 같다. 원자성

* 호원대학교 컴퓨터학부
** 연세대학교 컴퓨터과학과
논문번호 : 99215-0528, 접수일자 : 1999년 5월 28일

(atomic)은 멀티캐스트 메시지가 멀티캐스트 그룹으로 전송될 때, 멀티캐스트 그룹의 멤버 중 어느 하나라도 전송된 메시지를 받지 못하는 경우가 없이 모든 멤버들이 멀티캐스트 메시지를 수신하는 것이다. 순서화는 같은 멀티캐스트 그룹내의 모든 멤버들이 수신한 메시지들의 순서가 동일하도록 하는 것이다[2,4]. 멀티캐스트 통신에서 메시지의 순서화는 SSSG, MSSG, SSMG, MSMG로 분류할 수 있다. 또한 신뢰성을 보장하기 위해서는 송신자가 메시지를 멀티캐스트한 후, 수신자들이 메시지를 손실 없이 받았는지 확인해야 한다. 이러한 신뢰성을 보장하기 위해 주로 사용되는 기법은 ACK에 의한 선택적 재전송 메카니즘이다. 이 메카니즘은 송신자들이 송신한 메시지를 재전송 버퍼에 복사해 두고 수신자가 재전송을 요청하는 경우에 해당 메시지를 재전송한다. 따라서, 수신자가 송신자에게 해당 메시지를 받았다는 확인을 하므로써 송신자는 재전송 버퍼의 메시지를 삭제할 수 있다. 그러나, 다수의 수신자에게 일시에 ACK를 받아야 하는 것은 통신망에 너무 많은 부담을 주고 링크가 경쟁 링크인 경우에는 수신자들이 한꺼번에 ACK를 함으로써 많은 충돌이 발생하여 망 전체의 효율을 떨어뜨릴 여지가 있다[2]. 본 논문은 멀티캐스트에서의 신뢰성과 효율성을 위하여 메타그룹을 사용하며, 각 메타그룹의 DM만이 주 수신자에게 ACK를 보내어 ACK의 수를 줄이고 순서번호를 이용하여 선택적인 재전송을 하도록 함으로써 중첩된 재전송을 최소화시키는 효율적이고 신뢰성을 갖는 멀티캐스트 프로토콜을 제안한다.

Ⅱ. 관련연구

1. 기존의 멀티캐스트 프로토콜

그 동안 많은 멀티캐스트 메커니즘이 제안되고 구현되었다. Chang and Maxemchuck(CM) 프로토콜은 먼저 토큰을 사용하여 토큰 사이트라는 중앙 사이트를 정한다. 모든 송신자들은 중앙 사이트에게로 메시지를 전송하면, 중앙 사이트는 순서번호를 정하여 다른 여러 수신자들에게 메시지를 브로드캐스트하여, 수신자들이 동일한 순서로 메시지를 전달받는 프로토콜이다[1,5,6]. 이 프로토콜은 긍정 응답과 부정 응답을 사용한다. 송신자가 메시지를 멀티캐스트하면 중앙 사이트는 데이터를 손실 없이 받았을 경우 다른 여러 수신자에게 메시지를 전달하고, 데이터의 손실이 발생하였을 경우 중앙 사이트는 송신자에게 긍정 응답을 사용하여 Stop-and-wait로 에러 복구를 한다. 중앙 사이트는 메시지를 받는 각 수신자로부터 부정 응답을 받고, 손실된 메시지에 대해서는 책임지고 재전송을 요구한다. 중앙 사이트는 고정된 것이 아니라, 멀티캐스트 그룹내의 사이트간에 토큰이 순환하면서 중앙 사이트를 정한다. 중앙 사이트 이외의 사이트들이 재전송 요구를 하지 않는 경우는 모든 사이트가 메시지를 에러 없이 수신한 경우뿐만 아니라, 계속해서 메시지를 수신하지 못해 재전송 요구를 하지 않는 경우도 있기 때문에 중앙 사이트가 무한정 메시지를 저장하고 있어야 하는 단점이 있다. 이런 문제를 해결하기 위해 사이트들 사이에 토큰을 교환함으로써 중앙 사이트를 순환시켜서 모든 수신자들이 언젠가는 ACK를 할수 있도록 해서 무한정 메시지를 저장하는 단점을 해결하도록 하였다. 일반적으로 다수의 송신자가 존재하고 연결된 모든 사이트들에게 메시지를 전송하는 방송형 전달 방식에서 유용하다. 또한, 재전송하기 위해서 중앙 사이트가 확보해야 하는 버퍼의 크기가 수신자에 영향을 받기 때문에 수신자의 수가 적을 때 효율적이다. 그러나, 토큰을 주고받기 때문에 추가의 제어 메시지가 필요하고, 송신자와 중앙 사이트가 stop-and-wait 프로토콜로 동작하기 때문에 링크를 효율적으로 사용하지 못한다.[8]

Kaaxhoek and Tanenbaum(KT) 프로토콜은 순차기(sequencer)를 사용하는 프로토콜로서, 송신자들은 먼저 순차기라고 하는 중앙 사이트에게 일대일로 메시지를 보내고, 순차기는 메시지에 순서번호를 부여하여 모든 멤버들에게 브로드캐스트 하도록 하여 메시지의 순서가 일정하게 되도록 하는 방법이다[5,7]. 만일, 두 송신자가 동시에 메시지를 멀티캐스트하고자 할 때, 두 송신자가 모두 순차기에게 메시지를 보내면 순차기는 두 송신자가 보낸 메시지에 대해 순서번호를 부여하여 멤버들에게 브로드캐스트한다. 또한, 이 메시지 순서번호를 사용하여 수신자가 오류가 있는 메시지를 받을 경우, 수신자는 순차기에게 재전송을 요구하는 부정 응답 메시지를 보낸다. 순차기는 재전송 요구에 응답하기 위해서 재전송 버퍼에 자신이 멀티캐스트한 메시지를 저장하고 있다. 순차기 사이트는 멀티캐스트 그룹내의 모든 멤버들에 대한 ACK를 관리함으로써 모든 멤버들이 에러 없이 수신한 메시지 자신의 재전송 버퍼로부터 삭제한다. 이 방법은 토큰을 사용하지 않으므로 토큰 제어 메시지를 사용하지 않아도 되는 잇점을 가진다.

에러가 발생하는 수신 사이트만이 ACK와 함께 재전송 요구를 하고, 나머지 사이트들은 자신이 멀티캐스트할 메시지가 있을 경우, 수신에 대한 ACK를 Piggyback해서 보내므로 모든 사이트들이 번갈아 가며 ACK를 할수 있도록 해야 한다. 그렇지만, 어느 한 사이트만이 자주 ACK를 발생할 경우, 다른 사이트들에 대한 ACK가 수집되지 않아 순차기의 재전송 버퍼가 가득 찰 경우가 자주 발생한다. Reliable internet Multicasting(RM) 프로토콜은 단일 송신자를 기준으로 한 프로토콜로서 송신자가 연결된 게이트웨이를 루트로 하여 트리를 구성한다[5,8].

게이트웨이로 연결된 네트워크망 모델인 인터넷 같은 구조의 망에서 메시지들로 인한 망의 부하를 줄이기 위해서 제한된 프로토콜이다. 송신자가 수신자에게 일대일로 전송하는 것이 아니라, 가상 트리 구조를 따라 메시지를 전송하므로써 전체 수신자 노드에게 전송하기 위한 시간이 줄어든다. 또한, 수신자들이 보낸 모든 응답 메시지를 그대로 송신 받는 것이 아니라 트리의 게이트웨이에서 연결된 자식 노드들로부터 응답 메시지를 받은 후 부모 노드로 하나의 응답 메시지만을 보냄으로써 송신자에게로 전달되는 응답 메시지의 수를 줄이는 방법을 사용한다. 인터넷과 같은 상황에서 수신자가 많을 때 망전체의 부하가 감소하지만, 일반적으로 수신자 노드들의 수가 많아질 경우 트리의 깊이가 커지므로 수신자 노드에 메시지가 전달되는 시점의 편차가 커지고, 트리를 구성하기 위한 비용이 크다. Propagation algorithm(PT) 프로토콜은 전파 알고리즘과 메시지 전송 프로토콜로 구성된다. PG는 전파 트리를 생성하고, MP는 트리를 따라 메시지를 전송하는 프로토콜이다. 이 프로토콜은 중앙 집중형이 가지는 병목현상을 줄이기 위한 효율적이고 분산된 방법으로 멀티캐스트 메시지에 대해 다중 그룹 순서화 속성을 보증한다[5,6]. 여러 그룹에 중첩된 멤버를 중간 노드로 사용하여 트리를 구성함으로써 모든 수신자들에 대해 같은 메시지의 순서를 보증한다. 단점은 트리를 구성하는 비용이 많이 든다. 또한, 멀티캐스트 멤버의 변화에 대한 유연성이 적어서 그룹멤버의 변화는 트리가 재구성되도록 하므로 이것은 많은 비용의 부담을 준다. 따라서, 트리가 구성되면 많은 양의 데이터가 전달되는 경우에 적합하다. 전파 트리에서는 멀티캐스트 메시지가 목적지 노드에 도달하기 까지 불필요한 추가 노드의 존재와 관련한 오버헤드가 있다[5,8].

2. 응답(Acknowledgement) 방식

비 신뢰적인 통신망에서 신뢰성을 보장하기 위한 방법으로 송신자들은 재전송 버퍼에 전송한 메시지를 복사해 두고 수신자가 재전송을 요청할 경우에 해당 메시지를 재전송하고, 수신자가 메시지를 안전하게 받은 경우는 그 메시지를 재전송 버퍼에서 삭제한다. 송신자는 수신자들이 해당 메시지를 안전하게 받았다는 확인을 해야 한다. 확인 시스템은 기본적으로 긍정 응답(Pos. Ack.) 시스템과 부정 응답(Neg. Ack.) 시스템으로 분류할 수 있다. 긍정 응답 시스템은 각각의 모든 수신자들이 메시지를 받을 때마다 확인 메시지를 송신자에게 보내는 시스템이다. 이 방법은 메시지 전달을 명확히 확인할 수 있지만, 다수의 송신자로 구성된 시스템에서는 각 수신자들이 올바른 순서로 메시지를 받았는지 확인할 수가 없다. 부정 응답 시스템은 단일 송신자다수 수신자(Single Souce Multiple Receiver)의 경우 수신자들이 메시지를 수신할 때마다, 매번 응답 메시지를 보낼 필요 없이 메시지의 번호에 이상이 있을 경우에만 재전송을 요구한다. 그러나 다수 송신자일 경우 메시지의 순서 보장이 이루어지지 않으며, 송신자의 버퍼는 무한대라는 가정 하에서 수행된다. 신뢰성 있는 전송을 위한 일반적인 응답 방식은 수신측이 데이터를 받을 때마다 ACK를 송신자에게 보내고, 송신측에서는 다음 패킷을 보내기 전에 전송된 사본을 유지하며 ACK를 기다리게 된다. 전송된 패킷에 대한 ACK가 도착되면 다음 패킷을 전송한다. ACK를 이용한 재전송 방식은 3가지 방법 이 있다[3]. Stop-and-wait 방식은 단순 정지-대기 ACK 기법으로서, 송신자는 하나의 패킷을 전송한 다음 수신자로부터 응답을 받은후 다음 패킷을 전송하는 방식이다. 즉, 송신자는 한 개의 패킷을 전송하고 나서 응답이 올 때까지 기다려야 하고, 응답이 오지 않으면 더 이상의 패킷을 전송할 수 없으며, 수신자는 수신된 패킷에 대한 오류가 없으며 응답 신호로서 ACK를 전송하고, 오루가 발생되었으면 NAK를 전송한다. Stop-and-wait 방식은 전송 패킷이 잡음 등으로 인해 수신자에게 수신되지 않으면 수신자는 응답 신호를 전송하지 못하므로 통신의 중단이 발생할 수 있다. 그러므로 이를 고려하여 송신자에 타이머를 부착하여 한 개의 패킷을 전송한 후 타이머를 가동시켜 일정한 시간 동안 응답 신호가 오지 않으면 같은 패킷을 재전송 한다. Go-back-N 방식은 송신자가 패킷을 연속적으로 전송하고 재전송

요구 응답이 있는 시점에서 오류가 있었던 패킷까지 돌아가 그곳으로부터 N개의 블록을 순서가 바뀌지 않게 재전송하는 방식이다. 즉 송신자는 윈도우 크기만큼 연속된 패킷을 전송하며, 수신 측에서는 패킷을 수신하다 오류가 발견되면 그 패킷에 대한 NAK 응답 신호를 전송하고 오류가 발견된 패킷이 고쳐질 때까지 이후의 패킷은 수신하지 않는다. 따라서 송신측은 NAK 응답 신호가 수신되면, 오류가 발생된 패킷과 그 이 후의 모든 패킷들을 재전송해야 한다. Selective-repeat ACK 방식에서는 송신자가 NAK 응답 신호를 보낸 패킷만 재전송하므로, go-back-N 방식보다 효율적인 방식을 제공한다. 수신자는 오류가 발생된 패킷이 재 전송될 때까지 그 다음의 패킷을 저장할 기억장소와 재전송되는 패킷을 삽입할 수 있는 논리회로를 가지고 있어야 하며, 송신자도 재전송하기 위하여 전송 순서와는 다른 패킷을 전송할 수 있는 논리회로를 가져야 하므로 복잡하다.

Ⅲ. 제안된 멀티캐스트 프로토콜

본 논문에서 제안하는 프로토콜은 중앙 집중형이 가지는 병목현상을 줄이기 위한 효율적이고 분산된 방법으로써 멀티캐스트 메시지에 대해 다중 그룹 순서화를 가질수 있다. 이것은 전파 트리를 기반으로 한다. 또한, 멀티캐스트 프로토콜은 기존의 전파 트리 메카니즘이 갖는 문제점을 해결하고, 완전한 신뢰성을 보증하는 것을 목적으로 한다. 메타-그룹을 이용한 트리는 멀티캐스트 그룹들에게 메시지의 종합적 순서화를 보증하는데. 이것이 가지는 장점은 병렬성을 높이고, 트리의 깊이를 줄임으로서 메시지 비용성이 좋고, 지연시간이 짧아진다는 것이다. 그리고, 멤버 변화에 더 큰 유연성과 메시지 전달에서 높은 신뢰성을 제공 한다. 또한, DM을 각 메타-그룹에 두어 각 메타-그룹의 멤버들에게 메시지를 전달하고, ACK를 처리하며 선택적인 재전송을 담당하도록 한다. 그리고, 각 메타-그룹 멤버들에 대한 ACK를 처리하는 것과는 독립적으로 목적지 메타-그룹의 DM은 자신이 수신한 메시지에 대한 ACK를 곧바로 트리상에서 가장 가까운 PM의 DM에게 보내도록 하여 송신자의 ACK 처리와 재전송에 대한 부담을 감소시킨다.

3.1 메타-그룹을 이용한 트리

멀티캐스트 환경에서 프로세스들은 보통 다른 서비스를 제공하는 다른 프로세스들과 협력한다. 따라서 하나의 프로세스는 여러 멀티캐스트 그룹의 멤버로서 존재할수 있다. 그림 1.에서는 4개의 멀티캐스트 그룹이 서로 중첩된 모습을 보여준다. 이러한 중첩된 멤버들은 멀티캐스트에서 메시지의 순서화를 이루기 어렵게 한다. 이러한 어려움을 극복하기 위해 메타-그룹의 개념을 도입하여 트리를 구성한다. 메타-그룹을 이용한 트리를 구성하기 위해 다음과 같이 정의한다.

정의 1. 멀티캐스트 그룹:메세지를 전달하기 위한 목적지 프로세스들로 구성된 그룹이다. 멀티캐스트 그룹은 대문자 A,B,C,… 로 표시한다. 또한, 멀티캐스트 그룹 G의 멤버인 프로세스 I는 G(Pi)로 나타낸다.

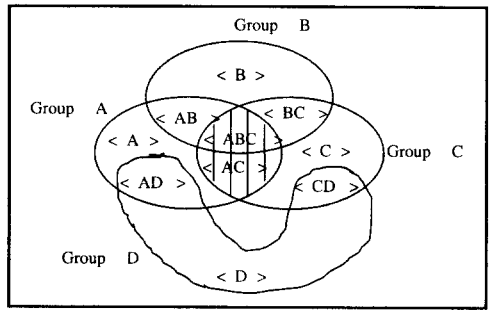

그림 1. 멀티캐스트 그룹의 중첩

정의 2. 메타-그룹:멀티캐스트그룹 Ga(Pi)=Gb(Pj) 일 때, 프로세스 Pi와 Pj의 집합이다. 메타-그룹은 다른 메타-그룹과 중첩할 수 없다. 메타-그룹은 "<>"으로 나타낸다. 그림 1. 의 예를 보면 멀티캐스트 그룹 A와 B에 중첩된 <AB>로 표시된 부분이 하나의 메타-그룹이고, 멀티캐스트 그룹 A,B 그리고 C가 중첩된 <ABC>로 표시된 부분이 하나의 메타-그룹이다. 이러한 경우 두 개의 송신자로부터 메시지 m1과 메시지 m2를 받는 두 메타-그룹<AC>와 <ABC>가 있을 때, <AC>는 m1을 먼저 받고 m2를 나중에 받았지만,<ABC>는 m1이 지연되어 m2를 먼저 받고 m1을 나중에 받을 경우 메타-그룹 <AC>와 <ABC>는 순서화를 보증할 수 없다. 따라

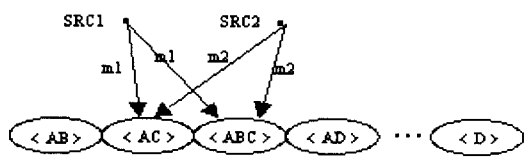

그림 2. 중첩된 그룹이 둘 이상인 경우의 순서화

서 이런 경우 두 메타-그룹에 대한 순서화를 보증하기 위해서 <AC>와 <ABC>는 메시지를 받기 이전에 메시지가 순서화 되어야한다. 만일 그림 3. 에서와 같이 <ABC>에서 <AC>로 경로를 만들면 <ABC>에서 먼저 메시지 M1과 M2을 수신하여 순서화를 시키고, 같은 순서로 메시지를 <AC>에게 전달하면, <AC>역시 같은 순서로 메시지를 전달받을 수 있다.

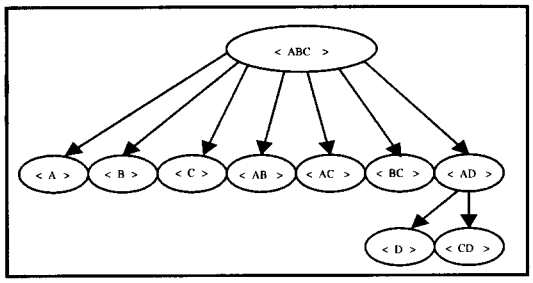

그림 3. 그림 2 의 트리

정의 3. Primary 메타-그룹(PM):멀티캐스트 그룹의 메타-그룹 중에서 다른 모든 메타-그룹의 조상인 메타-그룹이다. 각 멀티캐스트 그룹은 오직 하나의 PM을 갖는다. 즉, 하나의 PM은 몇 개 메타-그룹의 PM일수 있다. 그룹 Gi의 PM을 PM(Gi)로 표시한다. 멀티캐스트 그룹에 대한 메시지는 첫 번째로 PM에게 보내진다. PM은 다른 메타-그룹으로 메시지를 전파한다. 각각의 메타-그룹이 중첩되어 있지 않기 때문에 한 그룹으로 메시지를 멀티캐스트하는 것은 목적지 그룹의 Subset인 메타-그룹으로 메시지를 분리하여 보내는 것으로 이루어질수 있다. 다른 메타-그룹과 중첩되지 않기 때문에 메타-그룹으로 메시지를 순서화하는 것은 SSSG또는 MSSG의 경우로 볼수 있다. 그러므로 메타-그룹을 사용하는 것은 중첩된 그룹의 멤버들에 대한 메시지 순서화를 보증하기가 더욱 쉬워진다.

규칙 1. 각 멀티캐스트 그룹은 오직 하나의 PM을 갖는다.

규칙 2. 각 메타-그룹은 PM으로부터의 경로가 유일하다.

이런 경우 규칙2에 따라 [그림 4]와 같은 경로를 구성하면 PM(Gj)는 <aGiGjβ >와 <a'GiGjβ '>는 같은 순서로 메시지가 전달된다.

3.2 메타-그룹을 이용한 트리 생성 알고리즘

메타-그룹을 이용한 트리 생성 알고리즘은 트리를 구성하기 위해 멀티 캐스트 그룹으로부터 메타-그룹

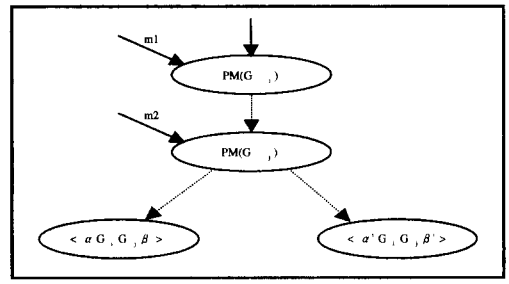

그림 4. <α GiGjβ >와 <α GiGjβ >에서 메시지의 순서

을 생성하고, 생성된 메타-그룹으로 트리를 구성하는 두단계로 이루어진다. 트리 생성 알고리즘의 목적은 앞에서 정의한 규칙1과 규칙2를 만족하는 트리를 구성하는 것이다. 트리를 구성하기 위해 추가적으로 다음을 정의한다.

정의 4. Cardinality:한 메타-그룹이 속해있는 멀티캐스트 그룹의 수이다.

정의 5. Subsume:한메타-그룹에 속해있는 그룹들로만 구성된 메타-그룹을 포함한다.

정의 6. Joint: 두 메타-그룹이 Subset이고, 연관된 두 메타-그룹이 포함 관계를 갖지 않는 하나 이상의 멀티캐스트 그룹에 존재할 때 두 메타 그룹은 Joint할수 있다. 트리 생성 알고리즘은 먼저 가장 큰 Cardinality를 갖는 메타-그룹을 루트로 표시한다. 만약 가장 큰 Cardinality가 하나 이상이면 임의로 선택한다. 그리고 루트로 표시된 메타-그룹에 포함되는 (정의 5) 모든 메타-그룹을 자식으로 만든다. 루트와 Subsume이 적용되지 않는 멀티캐스트 그룹을 고려 하여 루트의 멀티캐스트 그룹에 속하지 않는 메타-그룹에 대해 Cardinality가 가장 큰 메타-그룹을 루트와 joint(정의 6)하여 subroot로 만들어 subtree를 구성한다.

3.3 효율적인 재전송 메카니즘과 그룹 멤버 관리

본 논문에서 제시한 프로토콜은 효율적인 데이터 전송을 위해 윈도우 흐름 제어를 사용한다. 윈도우는 0부터 어떤 최대 값까지의 순서 번호를 포함하는데, 현재 버퍼에 저장하고 잇는 패킷의 순서번호를 표시한다. 따라서 ,수신 윈도우는 수신자가 유지하는 수신 버퍼에서 비어있는 공간의 크기를 나타낼 수 있다. 윈도우 흐름제어는 수신자가 유지하는 수신 윈도우에서 수신자의 가용 버퍼의 양을 송신자에게 알려줌으로써 송신자는 수신자의 윈도우 크기만큼 연속해서 패킷을 전송하는 것이다. 따라서 송신자가 데이터 패킷을 한법 전송하고 ACK를 확

```
main()
   {
     for each  group  Gi  whose pm(Gi)= 0  do;
      make Gis highest cardinal meta-group MG root;
      make MG;
       build_tree(MG);
     endfor
   }
built_tree(MG)
{ Let Root G be the set of multicast-groups of which
      MG is a subset;
    for each Gi in RootG do;
     if prn(Gi) = 0  then  prn(Gi) = MG;
    endfor
   for each unmarked meta-group subsumed by MG do;
      make it the child of MG and mark it;
   endfor
Let JointL be a list of unmarked meta-groups which is
Joint with MG;
for each unmarked meta-group MGi  in JointL do;
     Make MGi , the child of  MG and mark MG;
     build_tree(MGi );     /* recursive call */
endfor
}
```

알고리즘 1. 메타-그룹을 이용한 트리생성 알고리즘

인한 후,다음 패킷을 수신자에게 보내는 방법보다 효율적이다. 송신자는 Ws의 송신 윈도우을 갖고 수신자는 Wr의 수신 윈도우를 갖는다. Ws는 송신자가 전송할수 있는 패킷의 최대 수를 나타낸다. Wr은 수신자의 수신 버퍼 크기를 나타낸다. 수신 윈도우가 공간이 부족하여 패킷을 놓치는 경우가 발생하지 않도록 항상 Ws<Wr이 되도록 해야 한다. 수신자들은 정확히 수신되어 버퍼에 저장된 패킷드를 기록하는 비트맵과 1을 사용한다. 그림 5. 에서는 수신 버퍼의 크기가 8이고, 순서번호 15에서 22까지의 수신 상태를 나타내는 수신 윈도우의 구조이다. 이때, 순서번호 L은 L보다 작은 모든 순서번호가 정확히 수신됨을 나타내고, 윈도우의 수신 범위 내에 잇는 비트맵의 0은 손실된 패킷을, 1은 정확히 수신된 패킷을 나타낸다. 따라서 15번 이하의 패킷과 16,17 ,그리고 19번 패킷은 정확히 수신된 것이고, 15,18,20,21,22번은 정확히 수신되지 않은 것이다. 만약 15번 패킷이 정확히 수신되면 L은 18번으로 증가된다.

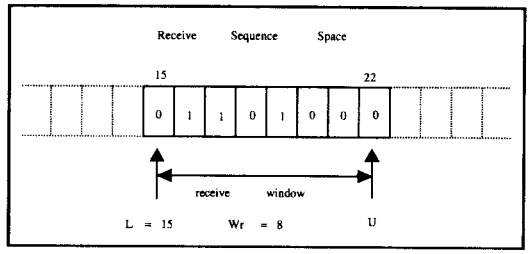

그림 5. 수신 윈도우 구조

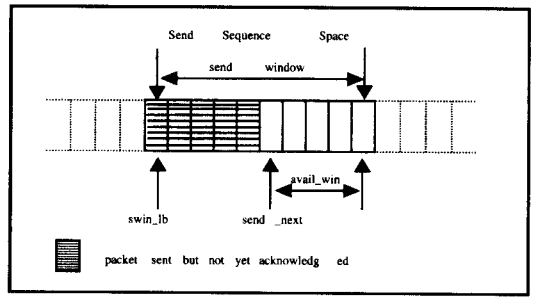

그림 6. 송신 윈도우 구조

그림 6. 은 송신 버퍼의 크기가 10인 송신 윈도우의 구조이다. Swin_Ib는 송신 윈도우의 하한 값이며, Send-next는 데이터 패킷을 송신할 때 사용하는 다음 순서번호이다. Avail-win을 감소시킨다. 송신 윈도우는 순서번호 Swin-Ib에 해당하는 메시지가 수신된 패킷임을 ACK로 응답 받을 때 Swin-Ib과 avail-win는 증가한다. 이러한 선택적인 재전송의 사용은 불필요한 재전송을 줄인다. 멀티캐스트 프로토콜이 효과적으로 동적인 그룹 멤버의 변화를 지원하는 것은 중요하다. 전통적인 트리 메카니즘에서 그룹 멤버의 변화는 트리의 재구성을 초래하기 때문에 멤버를 교환하기가 어려웠다. 그러나, 메타-그룹을 이용한 트리에서는 멤버의 변화가 메타-그룹의 생성과 삭제를 발생시키지 않는다. 즉, 멤버의 변화는 트리의 구조에 변화를 초래하지 않으므로, 트리를 재구성할 필요가 없다. 그러나 메타-그룹의 생성과 삭제가 일어나면, 멀티캐스트 트리의 구성에 영향을 끼치므로 트리를 재구성해야 한다. 본 논문의 프로토콜은 그룹 멤버의 동적인 변화를 위해서 두 연산자 add와 delete를 제공한다.

3.4 DM에 의한 효율적인 ACK 처리

멀티캐스트와 같은 환경에서 다수의 수신자가 하나의 송신자에게 일시에 ACK를 보내는 것은 망에게 너무 많은 부하를 주고 링크 경쟁으로 인한 폭증현상을 발생시킨다[2]. 그러므로, 본 논문에서는 이

러한 현상을 줄이기 위해 각각의 메타그룹에 DM이라는 새로운 관리자를 두어 메타-그룹의 멤버들에 대한 ACK처리와 데이터 재전송을 담당하도록 하여 송신자의 부담을 분산시켜 주도록 하였다. 기존의 메타-그룹 관리자는 멀티캐스트되는 메시지를 각각의 메타 그룹을 대표하여 수신하고, 그 메타-그룹의 모든 멤버들에게 메시지를 브로드캐스트하여 메시지의 순서화를 유지하였다. 그러나, 본 논문에서 제시한 DM은 송신자로부터 메시지를 수신하여 재전송 버퍼에 저장하고, 자신이 속한 메타-그룹내의 모든 멤버들에게 메시지를 브로드캐스트한다. 그리고, 자신이 속한 메타-그룹의 모든 멤버들로부터 ACK들을 송신자를 대신하여 받아 처리하고, 만약 재전송 요청이 있으면 자신의 재전송 버퍼에서 선택적 재전송을 한다. 또한, DM은 메타-그룹의 멤버들에 대한 ACK 처리와는 무관하게, 자신이 수신한 메시지에 대한 ACK를 자신이 속한 가장 가까운 PM에게 보내도록 하여 송신자가 빠른 시간내에 ACK를 처리할수 있도록 한다. 여기서 각각의 DM이 ACK를 자신이 속한 PM중 가장 가까운 PM에 ACK를 보내는 것에 주의하라, 메타-그룹은 하나 이상의 멀티캐스트 그룹에 속할수 있다. 따라서, 하나의 메타-그룹은 하나 이상의 PM을 가질 수 있다(정의 3). 만일, 어느 메타-그룹의 DM이 두 개의PM을 가질 때, 트리 상에서 DM에 가까운 PM2가 메시지 m2를 송신자로부터 받고, 더욱 상위에 존재하는 PM1이 메시지 m1을 송신자로부터 받아서 PM2에서 병합(merge)되었다고 하자. 이때, DM이 메시지 m2를 재전송 받기 위PM1 에게 ACK를 하였다면, PM1은 m2를 가지고 있지 않기 때문에 재전송을 할 수 없다. 따라서, DM은 ACK를 트리상에서 자신과 가장 가까운 PM에게 보내야 한다. DM을 사용하는 것은 멀티캐스트에서 수신자들이 모두 직접 ACK를 송신자에게 보내는 경우에 비해 ACK의 수를 감소시킨다. 또한, 기존의 트리를 이용한 멀티캐스트 프로토콜에서 트리의 경로를 따라 역순으로 ACK를 보내는 방법에 비해 DM이 직접 PM에게 ACK를 보내도록 하여 ACK 전달에 필요한 시간을 줄였다. DM은 송신자로부터 보내진 메시지를 수신하기 위한 수신 버퍼와 자기 메타-그룹의 멤버들에게 재전송를 하기 위한 재전송 버퍼를 가지고 있다. DM의 수신 버퍼는 재전송 버퍼에 공간이 비면 수신 버퍼의 패킷을 재전송 버퍼로 이동시키고 공간을 확보한다. DM은 재전송을 위해 재전송 큐에 다음의 네 가지 요소를 유지시킨다. 재전송된 패킷의 순서 번호, 패킷을 요구하는 수신자의 수, 수신자의 주소, 다음 큐를 가리키는 포인터이다. DM은 재전송시 재전송을 요구한 멤버가 하나이면 유니캐스트 그리고 재전송을 하나 이상의 멤버가 요구하였을 경우 멀티캐스트를 사용하여 재전송을 한다. ACK에는 순서번호 L과 비트맵 V 포함하도록 하여 재전송시 선택적인 재전송이 가능하도록 하였다. 메타-그룹에서 DM을 이용한 멀티캐스트 트리는 그림8.과 같으며, 본 연구에서 제안한 DM의 프로토콜은 알고리즘2.와 같다.

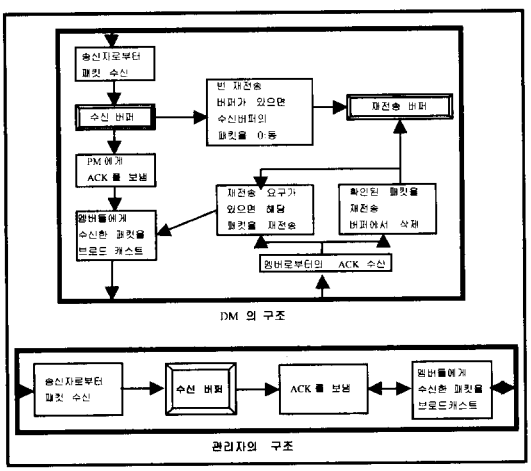

그림 7. DM과 관리자의 구조

Ⅳ. 성능분석

4.1 기존 프로토콜과의 비교

일반적으로 멀티캐스트 프로토콜들을 비교하기 위한 기준으로 먼저 모든 송신자가 그룹 멤버들에게 메시지를 보내는데 필요한 메시지 비용과 모든 멤버들이 메시지를 전달받을 때까지의 시간인 지연시간(latency time), 그리고 메시지를 대표로 수신한 멤버가 다른 멤버들에게 메시지를 전달한 후 확인 메시지를 받아 재전송 qjj에서 삭제 할수 있는 시간인 commit delay를 사용한다[1]. 먼저 메시지 비용에 대해 비교해 보면, 중앙 집중형은 첫 번째로 송신자에서 중앙 사이트로 메시지를 전달하므로 이때 메시지 비용은 1이고, 두 번째로 중앙 사이트에서 멥버로 브로드캐스트를 이용하여 메시지를 전달할 때 비용은 1이다. 따라서, 총 메시지 비용는 2이다. RM프로토콜은 송신자가 트리의 루트에게로 메시지를 보낼 때 메시지 비용이 1이고, 루트에서 목적지 최하위 노드까지 트리의 깊이가 Dr이라고 할 때,

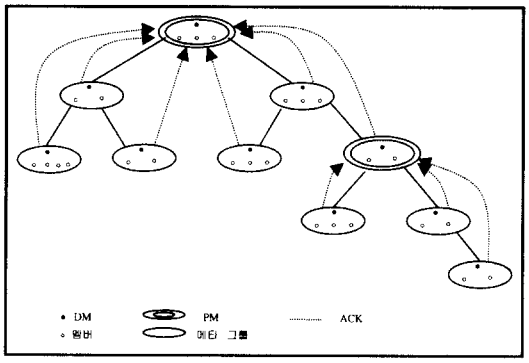

그림 8. 메타 그룹에서 DM을 이용한 새로운 멀티캐스트 트리

```
DM() {
for Receive Packet(seq_no) do;
   if (seq_no < L) then Discard Packet;
   Put Packet to Receive_Buffer;
   Send ACK(L, bitmap) to the nearst PM;
   Broadcast Packet to all Members;
   If (Rtans_Buffer full) then
   Move Packet from Receive_Buffer to Rtrans_Buffer;
endfor
for Receive ACK from Members do;
   if (bitmap=0) then Retrans(bitmap_no, trans_type);
   if (bitmap=1) then Del(bitmap_no);
endfor }
Retrans() {
if (C>1) then trans_type = Multicast
else trans_type = Unicast; for Retransmit Packet which
is bitmap_no = Retrans_seqno do;
endfor }
Del() {
for Delete Packet which is bitmap_no = Retrans_seqno
do;
endfor }
```

알고리즘 2. DM의 알고리즘

메시지 비용은 Dr이다. 따라서, 총 메시지 비용은 Dr+1이다. 전파 프로토콜도 RM 프로토콜과 유사하다. 송신자가 전파 트리의 주수신자에게로 메시지를 보낼 때 메시지 비용은 1이고, 주 수신자에게 최하위 목적지 노드까지 트리의 깊이가 Dp라고 할 때, 메시지 비용은 Dp이다. 따라서, 총 메시지 비용은 Dp+1이다. 메타-그룹을 이용한 트리에서는 먼저 송신자가 트리의 주 수신자인 PM에게 메시지를 전달하므로, 이때 메시지 비용는 1이고,PM에서 목적지의 최하위 메타-그룹의 관리자까지의 깊이인 Dm이다. 또한 최하위 메타 그룹의 관리자가 메타-그룹의 다른 멤버들에게 메시지를 브로드캐스트하므로, 이때 메시지 비용은 1이다. 그러므로, 총 메시지 비용은 Dm+2이다. 두 번째로 지연 시간을 비교하면, 모든 노드간의 지연시간을 T라고 가정할 때, T는 한 노드의 처리시간인 P와 네트워크에서 노드간의 전송시간 1의 합이다. 중앙 집중형은 송신자에서 중앙 사이트로 메시지를 전달할 때 T이고, 중앙 사이트가 모든 멤버에게 메시지를 브로드캐스트할 때 T이므로 ,총 메세지 지연시간은 2T이다. RM 프로토콜은 송신자가 트리의 루트에게로 메시지를 보낼 때 T이고, 루트에서 최하위 목적지 노드까지 트리의 깊이가 Dr이면 Dr*T이다. 따라서 총 메시지 지연 시간은 (Dr+1)*T이다. 전파 프로토콜도 송신자가 전파트리의 주 수신자에게로 메시지를 보낼 때 T이고, 주 수신자에서최하위 목적지 노드까지 트리의 깊이가 Dp이면 Dp*T이다. 따라서 총 메시지 지연 시간은 (Dp+1)*T이다. 메타-그룹을 이용한 트리는 먼저 송신자가 트리의 주 수신자인 PM에게 메시지를 전달할 때 T이고, RM에서 목적지의 최하위 메타-그룹의 관리자까지의 깊이가 Dm이면 Dm*T이다. 그리고 최하위 메타 그룹의 관리자가 메타-그룹의 다른 멤버들에게 메시지를 브로드캐스트 할 때 T이다. 그러므로 총 메시지 비용은 (Dm+2)*T이다. 다음은 Commit delay이다. Commit delay는 주 수신자에게 멤버들까지의 지연시간에 멤버들이 다시 확인 메시지를 주 수신자에게 전달하는 시간까지 포함되는 시간이다. 중앙 집중형은 중앙 사이트에서 멤버들에게 메시지를 전달할 때, 지연시간 T에 멤버가 중앙 사이트에게 확인 메시지를 보내는 시간 T가 더해진다. 따라서 2T이다. RM 프로토콜은 루트에서 목적지 최하위 노드까지의 지연시간 Dr*T에 최하위 노드가 다시 루트에게 확인메시지를 트리를 거쳐 전달하므로 트리의 깊이인 Dr가 추가된다. 그러므로,Commit delay,는 2Dr*T이다. 전파 트리도 RM 프로토콜과 같으므로, 2Dp*T이다. 메타-그룹을 이용한 트리에서는 RM프로토콜 과 같은 방법을 사용한다면 2Dp*T이다. 메타-그룹을 이용한 트리에서는 RM프로토콜과 같은 방법을 사용한다면 (2Dm+2)*T이다. 하지만, DM을 이용한 프로토콜에서 DM은 PM으로부터 전달받은 메시지를 자신의 메타-그룹 멤버들에게 전달하는 것과 ACK를 처리

하는 책임을지기 때문에,DM 자신이 PM으로부터 받은 메시지에 대한 ACK를 멤버들의 확인없이 PM에게 보낼수 있다. 그러므로, PM에 대한 Commit delay는 최하위 메타-그룹의 DM까지만 계산하면 된다. 또한, 주 수신자와 목적지 최하위 메타-그룹 사이에 목적지 최하위 메타-그룹이 속한 또 다른 PM이 존재할 경우 이들 PM을 거쳐서 주 수신자에게 ACK가 전달되어야 한다. 따라서, ACK를 보내야하는 메시지를 수신한 PM에서 목적지 최하위 노드 사이에 존재하는 다른 PM의 수를 Dm'라 하면, DM을 이용할 때 Commit delay는 (Dm+Dm')*T이다. [표1]에서 가장 우수한 성능을 보이는 것은 중앙 집중형이다. 그러나, 중앙 집중형은 병목현상이 심하다는 단점이 있다. 그리고, 같은 멀티캐스트 환경하에서 메타-그룹을 이용한 트리는 전파 트리 프로토콜이나, RM 프로토콜보다 트리의 깊이가 작기 때문에 전체적인 성능에서 우수함을 나타낸다[9].

표 1. 멀티캐스트 프로토콜의 성능비교

	중앙 집중형	전파 트리	RM	메타-그룹
메시지 비용	2	Dp+1	Dr+1	Dm +2
지연시간 (L+P)	2T	(Dp+1)*T	(Dr+1)*T	(Dm +2)*T
Commit Delay (대표 수신자 기준)	2T	2 DpT	2 DrT	DM을 사용한 경우 : (Dm+Dm)*T 기존의 ACK 방식 : (2 Dm +2) * T

Dm : 메타-그룹을 이용한 트리에서의 깊이,
Dr : RM프로토콜에서 트리의 깊이,
Dp : 전파 트리에서 트리의 깊이.

4.2 시뮬레이션 분석과 결과

시뮬레이션을 통해서 제시한 기법에 대해 성능 분석을 한다. 시뮬레이션을 하기 위해 우선, 한 노드당 처리 시간을 P, 노드간의 전송 지연시간을 L이라 할 때, 각각의 노드당 지연시간은 P+L이며, T로 표시한 다. 그리고 모든 노드들간의 지연시간 T는 같다고 가정한다. 또한, 네트워크의 전송 에러율은 0.01로 설정한다. 시뮬레이션에서는 각 멀티캐스트 그룹이 중첩할 수 있는 그룹의 개수를 5개 이하로 하였다. 프로세스는 무작위로 각 그룹에 할당 된다. 시뮬레이션은 각 모델별로 매번 다른 데이터를 발생시켜 100번씩 수행하는 평균값을 구하였다. 멀티 캐스트그룹에 따른 프로토콜의 전체적인 성능을

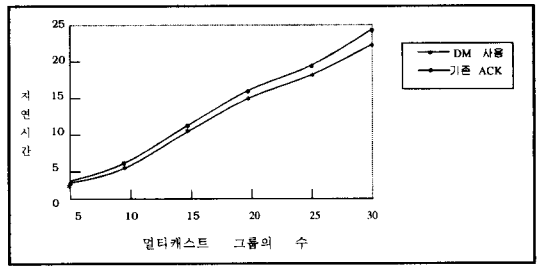

그림 9. 멀티캐스트 그룹의 수와 지연시간

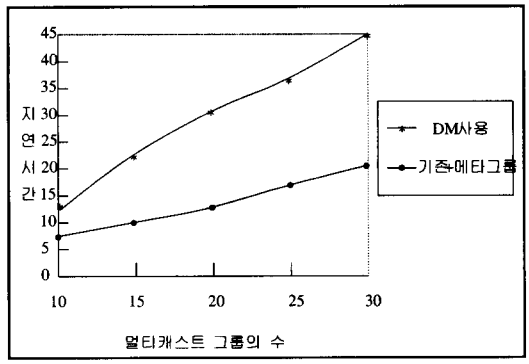

그림 10. 주 수신자에 대한 Commit delay

평가하기 위해 그림 9. 에서는 본 논문에서 제시한 메타-그룹을 이용한 트리 구조에서 DM을 사용한 경우와 기존의 트리 프로토콜에서 사용하는 ACK처리와 재전송 기법을 사용한 경우에 PM에서 목적지 최하위 메타-그룹의 관리자까지의 지연시간을 나타낸다. 이 지연시간은 송신자가 유지해야 하는 재전송 버퍼의 크기에 영향을 미친다. 트리의 깊이가 깊어질 때의 지연시간이 DM을 사용했을 경우에 기존 트리 프로토콜에서 사용하는 ACK기법과 재전송 방법을 사용한 경우보다 지연 시간이 더 크다는 것을 알 수 있다. 이것은 전송시 에러가 발생하면 기존의 방법에서는 ACK를 바로 상위 노드가 보내서 재전송을 받지만, DM을 사용하는 경우 바로 상위 노드가 아닌 가장 가까운 PM에게 ACK를 보내고 재전송 받기 때문에 재전송을 할 때, 다시 트리의 경로를 따라 전달되므로 바로 상위 노드에게 재전송을 받는 것보다 지언 시간이 길어진다. 그렇지만, PM을 기준으로 한 전체적인 Commit delay를 나타내는 그림 10. 을 보면 DM을 사용한 경우는 메타-그룹의 멤버들에 대한 ACK처리와는 독립적으로 PM에게 ACK를 한다. 하지만, 기존의 방법에서는 메타-그룹의 멤버들에게 모두 ACK를 받아서 PM에게 ACK를 보내기 때문에, PM에 대한 지연시간은 메

타 그룹의 ACK를 처리하는 지연 시간을 추가해야 한다. 또한, PM에게 ACK를 보낼때, 트리의 경로를 따라 전송하므로 DM을 이용한 것보다 더욱 지연이 된다. 따라서, DM을 사용한 경우가 기존의 방법으로 ACK를 처리하는 방법보다 PM의 부담을 감소시키므로 송신자의 부담을 감소시킨다는 것을 알 수 있다.

V. 결 론

본 논문은 트리를 이용한 멀티캐스트 프로토콜에서 메타-그룹(MG)의 개념을 도입하고, 새로운 응답(ACK)메시지의 전달 방법을 제안하였다. 기존의 멀티캐스트 트리에서는 사이트를 하나의 노드로 하여 트리를 구성하였으나, 새로운 멀티캐스트 프로토콜은 메타-그룹을 하나의 노드로 하여 트리를 구성하였다. 메타-그룹의 사용은 멀티캐스트 트리에서 다중 송신자-중첩된 다중 수신자의 순서화를 보장하고, 트리의 깊이를 작게 하였다. 또한, DM의 사용은 송신자가 처리해야 하는 ACK의 수를 줄여주고, 멀티캐스트 메시지에 대한 응답 시간을 감소시켜서 재전송 버퍼의 크기를 줄였다.

참 고 문 헌

[1] J. Chang. And N. F. Maxemchuk, Reliable Broadcast Protocols, ACM Tran. on Com. Systems. 2(3), Aug 1984

[2] John C. Lin Sanjoy Paul, RMTP : A Reliable Multicast Transport Protocol, Pro. of IEEE INFO96, Mar. 1996

[3] Fred Halsall, Data Comm., Computer Networks and OSI, Add. Wesley, 1992

[4] J. N. Gray and M. Anderton, Distributed Computer Systems, Proc. IEEE, Vol. 75, No. 5, pp. 719-726 May 1987

[5] Hactor Garcia-Molina, Annemarie Sp- auster, Message Ordering in A multicast Environment, IEEE Communicatio- ns Magazine, June 1988.

[6] Hactor Garcia-Molina, Annemarie Spauster, Ordered and Reliable Multicast Comm. ACM Tran. on Com. Systems, Vol. 9, No. 3, August 1991.

[7] M. F. Kaashoek, A. S. Tanenbaum, et al, An Efficient Reliable Broadcast Protocol, Op. Sys. Rev. Vol. 23, No. 4, Oct 1989

[8] Sanjoy Paul, Krishan K. Sabnani, Danid M. Kristol, Multicast Transport Protocols for High Speed Net. Pro. of network protocols 1994

이 동 춘(Dong-Chun Lee) 정회원

연세대학교 컴퓨터과학과 박사 수료
호원대학교 컴퓨터학부 부교수
호원대학교 전자계산소 소장

<주관심 분야> 컴퓨터 네트워크, 이동통신, 멀티미디어 무선통신, 통신의 성능 분석

김 배 현(Bae-Hyung Kim) 정회원

호원대학교 컴퓨터학부 졸업
수원대학교 대학원 전자계산학과 졸업
현재 : 호원대학교 컴퓨터학부 시간강사
<주관심 분야> 이동통신, 멀티미디어 무선 통신.

송 주 석(Joo-Seok Song) 정회원

연세대학교 컴퓨터과학과 정교수
통신학회 논문지 제23권 제 7 호 p1859 참조