論文 96-21-4-11

중첩된 프로세스 그룹 환경에서의 멀티캐스트 알고리즘

正會員 권 봉 경*, 정 광 수*, 현 동 환**, 함 진 호**

A Multicast Algorithm in Overlapped Process Group Environments

Bong-Kyoung Kwon*, Kwang-Sue Chung*, Don-Whan Hyun**,
Jin-Ho Hahm** *Regular Members*

要 約

본 논문에서는 하나의 프로세스가 여러 그룹에 속하는 중첩된 프로세스 그룹 환경에 효과적으로 적용할 수 있는 멀티캐스트 알고리즘을 제시하였다. 기존의 알고리즘과 달리 중첩된 프로세스 그룹 환경에서 그룹통신을 효율적으로 수행 할 수 있도록, 프로세스 그룹의 정보를 이단 트리로 작성하는 방식을 제안하였다. 이 방식을 이용하면 순서화를 위하여 불필요한 프로세스를 거치는 기존 알고리즘의 단점을 개선할 수 있다. 그룹통신에 전체적인 순서화 뿐만 아니라 인과관계 순서화의 기능을 부여함으로써 적은 지연을 요하는 메세지의 전달이 가능하도록 하였으며, 순서화에 요구되는 메세지 오버헤드의 크기도 줄였다. 또한, 제안된 인과관계 순서화 방식을 논리적으로 증명하였고, 시뮬레이션을 통해 기존 알고리즘과의 성능을 비교하였다.

ABSTRACT

In this paper, we proposed a new multicast algorithm which is efficiently applicable to overlapped process group environments where one process may be involved in several process groups. Unlike the existing algorithms, the proposed one provides an efficient group communication mechanism by generating the process group information in two-level tree. Using this algorithm, we improved the shortcoming of the existing algorithms by reducing the overhead in passing through unnecessay processes for message ordering. We have provided the causal ordering method as well as the total ordering method in group communication environments. As a result, we allow one process to deliver message to other processes with a short delay time, and reduced the overhead required for the message ordering. Also, we logically proved the proposed causal ordering method, and compared the performance of the proposed algorithm with ones of other existing algorithms by computer simulation.

*광운대학교 전자통신공학과 신기술연구소
**한국전자통신연구소 멀티미디어표준연구실
論文番號:95433-1218
接受日字:1995年 12月 18日

I. 서 론

최근 통신망이 고속화되고 대역폭이 증대됨에 따

라 화상회의, CSCW(Computer Supported Cooperative Work), 분산 데이타베이스 시스템, 고장허용 시스템(fault tolerant system) 등 다양한 분산 응용들이 등장하고 있다. 이들은 기존의 응용들과는 다른 새로운 형태의 통신기능을 요구하는데, 이러한 기능 중에 하나가 멀티캐스트(multicast)이다. 멀티캐스트란 동일한 정보를 하나의 송신지로부터 다수의 목적지로 전달하는 통신 방식으로, 유니캐스트(unicast)및 브로드캐스트(broadcast)와는 구분되어진다.

유니캐스트란 각 목적지로의 일대일 통신을 수행하는 것으로서, 다수의 목적지로 메세지를 전송하려면 이를 복사하여 목적지에 반복적으로 전송해야 하므로 전체 네트워크의 트래픽을 가중시킨다. 브로드캐스트는 네트워크에 연결되어 있는 모든 곳으로의 통신을 수행하는 것으로서, 현재 많은 네트워크 시스템이 물리적인 네트워크에서 브로드캐스트를 지원하고 있다. 그러나 임의의 프로세스(process)가 불필요한 메세지를 수신할 수도 있어 특수한 경우를 제외하고는 많이 사용되지 않는다.

이러한 유니캐스트와 브로드캐스트의 단점을 해결할 수 있는 것이 멀티캐스트이다. 멀티캐스트 환경에서는 프로세스 그룹간에 통신이 일어나게 되는데, 프로세스 그룹이란 단일 개체로서 여길 수 있는 통신 종단, 또는 프로세스의 집합이다. 프로세스 그룹에 관한 연구로는 미리 설정된 프로세스 그룹들 간의 그룹 통신에 관한 연구[8][9][10][11]와 하나의 프로세스가 여러 그룹에 속한 경우에 있어 메세지 송수신에 관한 연구[9][12], 프로세스 그룹의 동적인 변화에 관한 연구[19] 등이 있다.

이외에도 멀티캐스트에 관한 연구로는 주고받는 메세지의 순서화 문제[1][9][12][13][15][18], 효율성 문제[2][3][5], 멀티캐스트 도중에 발생하는 오류에 대처하는 연구[6][7], IP를 이용한 멀티캐스트 방법[20], 부분적인 순서화를 이용하여 그래픽하게 전체적인 순서화를 만드는 방법[13], 전송계층(transport layer)을 이용하여 멀티캐스트하는 방법[16][17]등이 활발히 진행되고 있다.

본 논문에서는 프로세스들이 여러 그룹에 속하는 경우에 있어 기존의 알고리즘보다 효율적으로 원하는 프로세스 그룹에게 메세지를 전달하는 메카니즘에 관하여 다루고 있다. 본 논문의 구성은 II장에서 중첩된 프로세스 그룹 환경을 위한 기존의 멀티캐스트 알고리즘과 그에 따른 문제점을 설명하였고, III장에서는 기존 알고리즘의 문제점을 대처할 수 있는 새로운 알고리즘을 기술하였으며, IV장에서는 제안된 알고리즘을 논리적으로 증명하였다. V장에서는 제안된 알고리즘을 기존의 알고리즘과 시뮬레이션을 통해 비교하였으며, VI장에서 결론을 맺는다.

II. 기존의 멀티캐스트 알고리즘

다양한 멀티캐스트 알고리즘이 연구, 제안되었으나, 임의의 프로세스가 동시에 여러 그룹에 속하는 경우, 즉 프로세스 그룹의 중첩 시에 효율적으로 적용할 수 있는 알고리즘은 매우 드물다. 현재까지 제안된 대표적인 알고리즘으로는 Molina and Spauster 알고리즘과 확장된 CBCAST 알고리즘이 있다. 각각을 살펴보면 다음과 같다.

Molina and Spauster 알고리즘은 한 프로세스가 여러 그룹에 속하는 중첩된 그룹 환경에서 멀티캐스트를 수행하는 방식이다[9]. 그림 1에서와 같이 8개의 프로세스 그룹($g1=\{c, d\}$, $g2=\{a, b, c\}$, $g3=\{b, c, d, e\}$, $g4=\{d, e, f\}$, $g5=\{e, f\}$, $g6=\{b, g\}$, $g7=\{c, h\}$, $g8=\{d, j\}$)이 존재할 시 모든 프로세스를 포함하는 멀티캐스트 트리를 만들어 메세지를 전송하므로서 수신측에 같은 순서화를 제공한다. 이 알고리즘은 중첩이 많은 환경에서 유리하다. 그러나 그룹내 각 멤버가 멀티캐스트 전송을 하고자 할 때 불필요한 여러 프로세스를 거쳐야 하므로 그에 따른 서비스 시간이 길어지는 단점이 있다.

확장된 CBCAST 알고리즘의 멀티캐스트 방식은 중첩된 그룹의 수만큼 벡터 타임스탬프를 생성하여 이용한다[12]. 그림 2에서 프로세스 P1은 g1과 g2에 대한 벡터 타임스탬프를 가지고 있으며, g1에 메세지 전송시 g1의 타임스탬프를 갱신한 후 g1, g2에 대한 타임스탬프를 같이 전송한다. 프로세스 P2는 m1을 응용에게 전달한 후 자신의 타임스탬프를 갱신한다. 만약 P2가 그룹 g2에게 메세지를 전송하고자 한다면, P2는 g2에 대한 벡터 타임스탬프를 갱신한 후 그룹 g2의 프로세스들에게 g1, g2에 대한 벡터 타임스탬프를 메세지와 함께 전송한다. 따라서 중첩된 프로세스 P3는 g2에 대한 메세지 m_2를 먼저 받더라도 벡터 타임스탬프 내용에 따라 전달조건이 만족되지 않으므로 m_1이 전달될 때까지 지연시킨다. 이와 같은 방식

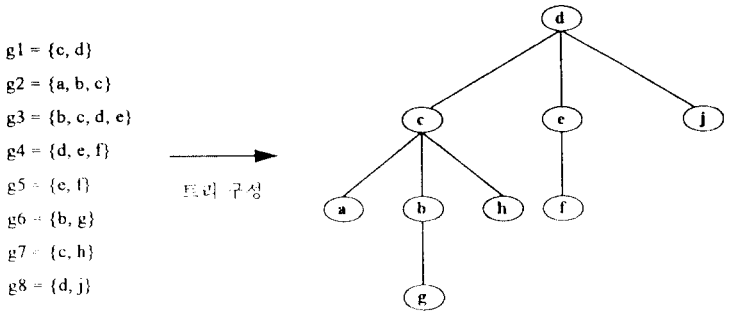


그림 1. Molina and Spauster 알고리즘에 의한 트리 생성
Fig. 1. Tree generation based on Molina and Spauster algorithm

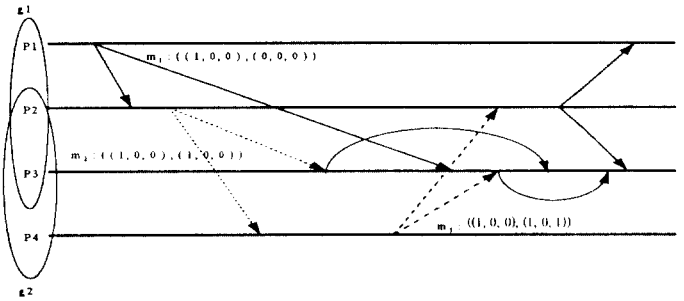


그림 2. 확장된 CBCAST의 예
Fig. 2. Example of the extended CBCAST

으로 중첩된 프로세스 환경에서도 인과관계 순서화를 만족시키며 멀티캐스트를 수행한다.

CBCAST 알고리즘의 문제점을 지적해보면, 프로세스 P1이 메세지 m_1을 그룹 g1에게만 전송함에도 불구하고 P2, P3를 위해 자신에게 불필요한 g2의 벡터 타임스탬프도 같이 전송한다는 것이다. P4 역시 불필요한 그룹 g1에 대한 벡터 타임스탬프를 유지해야 한다. 즉 자신이 속한 그룹에서 여러 그룹에 중첩되어 있는 프로세스가 존재할 시 중첩된 프로세스 때문에 중첩되지 않은 프로세스들이 불필요한 벡터 타임스탬프를 지녀야 하는 단점이 있다. 프로세스 그룹이 커지고, 중첩되는 그룹이 많아질수록 이러한 오버헤드는 더욱 늘어나게 된다.

Ⅲ. 새로운 멀티캐스트 알고리즘

본 논문에서는 II장에서 지적한 기존 알고리즘의 문제점을 해결하므로서, 프로세스 그룹의 중첩 시에 효율적으로 멀티캐스트를 수행하기 위한 방식을 제시하고자 한다.

3.1 중첩된 프로세스 그룹 환경을 위한 새로운 알고리즘

본 알고리즘은 Molina and Spauster 알고리즘과 같이 다단 트리를 생성하는 것이 아니고 독립된 이단 트리를 생성시킨다. 여기서 각 이단 트리의 루트들은 자신이 속한 프로세스 그룹의 중심 프로세스가 된다. 제안된 알고리즘의 수행과정은 다음과 같다.

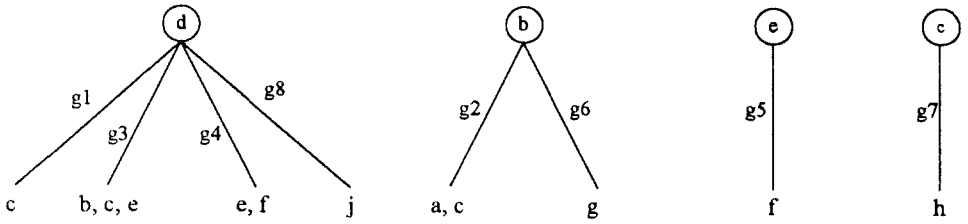


그림 3. 제안된 알고리즘에 의한 트리 생성
Fig. 3. Tree generation based on the proposed algorithm

(1) 존재하는 프로세스 그룹들 중 가장 많이 중첩되는 프로세스를 이단 트리의 루트로서 설정한다. 중첩 횟수가 같을 경우 그룹 멤버가 많은 프로세스를 선택하고, 두가지 모두 같을 경우 임의로 선택한다.
(2) 루트 아래 그룹 식별자로서 링크를 만든 후 해당 프로세스들을 위치시킨다.
(3) 전체 그룹중에서 링크가 생성된 그룹을 제외시킨다.
(4) 링크가 생성되지 않은 그룹들 중에서 (1)-(3) 과정을 반복한다.
(5) 잔여 그룹에서의 프로세스가 두개 이상의 그룹에 속하지 않을때 까지 반복하여 수행한다.
(6) 마지막 단계로서, 중첩 현상이 없는 그룹별로 임의의 루트를 설정하여 개별적인 이단 트리를 만든다.

Molina and Spauster 알고리즘과 같은 예제를 적용해 보면 8개의 그룹중 가장 많이 중첩되는 d 프로세스를 이단 트리의 루트로 선택한다. 이때 그룹 1, 3, 4, 8은 제외된다. 나머지 그룹에 대해서는 b와 c가 중첩된 그룹 수와 자신이 속한 멤버의 수가 모두 같으므로 임의로 b를 선택한다. 중첩이 일어나지 않은 경우에는 각각의 그룹에 대해 임의로 루트를 선택하므로 e와 c가 선택된다. 그러므로 d, b, e, c는 중심 프로세스가 된다. 결과적으로 그림 3과 같은 그래프가 생성된다.

초기에 지정된 하나의 프로세스는 그림 3의 트리를 생성하여 각 그룹 멤버에게 전송하게 된다. 이때 그래프 정보를 수신한 프로세스들은 어느 프로세스가 그룹의 중심 프로세스인지를 알수 있게 된다. 제안된 알고리즘은 이와 같은 트리 정보를 이용하여 전체 순서화와 인과관계 순서화를 제공하도록 설계되었다.

3.2 순서화 제공

본 절에서는 중첩된 프로세스 그룹 환경에서 제안된 멀티캐스트 알고리즘을 이용하여 제공되는 전체 순서화와 인과관계 순서화를 기술하고 있다.

3.2.1 전체 순서화

제안된 알고리즘은 Molina and Spauster 알고리즘과 달리 불필요한 프로세스를 거치지 않고도 전체 순서화를 만족시킨다. 전체 순서화는 단일 송신측 순서화(single source ordering), 다중 송신측 순서화(multiple source ordering), 다중 그룹 순서화(multiple group ordering)라는 세 가지 형태의 순서화로서 구성된다. 단일 송신측 순서화란 한 송신측이 수신 그룹에게 임의의 메세지 m_1과 m_2을 전송하면 모든 수신 그룹은 같은 순서로 메세지를 응용 프로세스에게 전달하는 것이다. 다중 송신측 순서화란 서로 다른 송신측이 메세지 m_1과 m_2를 같은 수신 그룹에게 전송하면 수신 그룹의 모든 프로세스는 같은 순서화로 메세지를 응용 서비스에게 전달하는 것이다. 다중 그룹 순서화란 서로 다른 그룹의 프로세스로부터 송신된 두개의 메세지 m_1, m_2가 다른 수신 그룹에게 전달될 경우, 중첩된 프로세스 그룹에서 같은 순서화로 메세지를 전달하는 것이다.

다중그룹 순서화는 특별한 응용에서만 제한적으로 요구되므로, 본 논문에서는 응용에 따라 다중그룹 순서화를 선택적으로 제공하는 멀티캐스트 알고리즘을

제안하였다.

⊙ 다중 그룹 순서화를 고려하지 않는 방식

본 방식에서는 단일 송신측 순서화와 다중 송신측 순서화를 유지하기 위해 GI(Group Information)라는 정보 구조를 정의하였으며, 이는 '그룹 id', '프로세스 id', 'sequence number'로 구성된다. GI는 모든 프로세스에서 멀티캐스트를 수행할 시 항상 일정 크기의 메시지 오버헤드를 갖는다. 즉 그룹의 크기나 수와는 관계가 없다. 이 방식은 다음과 같은 방법으로서 단일 송신측 순서화와 다중 송신측 순서화를 만족시킨다.

(1) 프로세스가 메세지를 멀티캐스트 할 경우 메세지와 함께 GI를 해당 그룹의 중심 프로세스에게 전송한다. 만약 자신이 그룹의 중심 프로세스인 경우 직접 멀티캐스트를 수행한다.
(2) GI와 함께 메세지를 받은 중심 프로세스는 메세지 수신 순서에 따라 멀티캐스트를 수행한다.

⊙ 다중 그룹 순서화를 고려한 방식

본 방식에서는 먼저 다중 그룹 순서화를 필요로 하는 프로세스를 지정하며, 그중의 한 프로세스에게 다중 그룹 순서화를 전담시킨다. 여기서 다중 그룹 순서화를 필요로 하는 프로세스란 두개 이상의 프로세스가 서로 다른 트리에 공통으로 존재하는 프로세스이다. 예를 들면, 그림 3에서는 그룹 g2, g3에 공통으로 존재하는 b, c와 그룹 g4, g5에 공통으로 존재하는 e, f가 해당된다. 전담 프로세스를 선출하는 방법은 다중 그룹 순서화를 필요로 하는 프로세스중 중심 프로세스가 있는 경우, 이를 전담 프로세스로 지정한다. 만약, 그렇지 않은 경우에 대해서는 중첩 횟수가 적은 프로세스를 선출한다. 그러므로 그림 3에서의 전담 프로세스는 b, c의 경우 b가 선출되며, e, f에 대해서는 e가 선출된다. 다중 그룹 순서화를 제공하는 멀티캐스트 방식을 구체적으로 설명하면 다음과 같다.

(1) 보통의 프로세스는 메세지와 함께 GI를 해당 그룹의 중심 프로세스에게 전송한다. 또한 다중 그룹 순서화를 위한 전담 프로세스는 자신이 속한 그룹에 보통의 프로세스가 존재할 경우 중심 프로세스에게 GI를 보내며, 다중 그룹 순서화를 필요로 하는 프로세스에게는 직접 멀티캐스트를 수행한다.
(2) GI와 함께 메세지를 수신한 중심 프로세스는 보통의 프로세스에게 멀티캐스트를 수행한다. 만약 전송한 그룹에, 다중 그룹 순서화를 필요로 하는 프로세스와 보통의 프로세스가 존재 한다면 전담 프로세스에게는 GI와 함께 메세지를 전송하며, 보통의 프로세스에게는 중심 프로세스가 메세지를 전송한다.
(3) 중심 프로세스로 부터 메세지를 수신한 전담 프로세스는 다중 그룹 순서화가 필요한 중첩 프로세스에게 메세지를 전송한다.

예를 들어 그림 3에서 a 프로세스는 그룹 g2에게 메세지를 전송하고, e 프로세스는 그룹 g3에게 메세지를 전송한다고 가정하면, 그룹 g2의 a 프로세스는 먼저 중심 프로세스인 b에게 멀티캐스트를 수행하고, 그룹 g3의 e 프로세스는 중심 프로세스 d에게 멀티캐스트를 수행한다. 그리고 중심 프로세스 d는 알고리즘 (2)에 따라 전담 프로세스인 b에게 메세지를 전송하며, 전담 프로세스 b는 a와 d 프로세스에서 전송한 메세지의 도착 순서에 따라 c에게 메세지를 전송한다. 그러므로 프로세스 b와 c에서 같은 순서화가 유지된다. 프로세스 e와 f에 대해서 같은 절차를 적용해보면 상기와 마찬가지로 동작한다.

3.2.2 인과관계 순서화

Molina and Spauster 알고리즘은 전체 순서화만을 제공하지만, 본 논문에서는 인과관계 순서화를 제공하므로써 적은 지연을 필요로 하는 메세지에 대해 빠른 전달을 가능하게 한다. 본 절에서는 인과관계 순서화만을 요구하는 메세지에 대해서, 이를 중심 프로세스에게 보내는 것이 아니라 각자의 그룹 멤버들이 국부적인 벡터 타임스탬프를 이용하여 멀티캐스트를 수행하는 방식을 제안하고자 한다. 그러나 인과관계 순서화는 전체 순서화를 제공할 때와 달리 순서화를 수행하는 프로세스가 없으므로, 여러 그룹에 중첩된 프로세스에서는 인과관계 순서화가 성립되지 않을 수가 있다. 그러므로 인과관계 순서화를 우선 정의하고, 중첩된 프로세스의 인과관계 순서화를 유지할 수 있는 새로운 멀티캐스트 방식을 제시한다.

인과관계 순서화는 'happened before' 관계이며→로 나타낸다[1]. 여기서 메세지의 송신과 수신은 이벤트의 발생이라고 가정한다.

[정의 1] 만약 다음의 조건중 하나라도 성립되면 두개의 이벤트 a, b의 관계는 a→b이다.

(1) 임의의 그룹중 한 프로세스에서 b보다 a가 먼저 발생한 경우

(2) 어느 한 프로세스의 a는 송신 이벤트이며, b는 다른 프로세스에서의 같은 메세지의 수신 이벤트인 경우

(3) a→c, c→b인 이벤트 c가 존재할 경우

정의에서는 이벤트 a에 관해 a→a라고 가정한다(이벤트는 그 자신이 happened before 관계가 될 수 없다). 또한, 두개의 이벤트 a, b가 a→b, b→a 일 경우 두 이벤트는 인과관계가 없는 동시적인 메세지이며 a‖b로 나타낸다.

[정의 2] 임의의 중첩된 프로세스가 g1으로부터 m을 수신하고 g2에게 m′를 송신할 경우 인과관계 순서화는 m→m1′이다.

제안된 메카니즘은 확장된 형태의 CBCAST와 달리 각 그룹의 멤버가 자신이 속한 그룹에 대한 벡터 타임스탬프만을 유지하며 통신하는 것이다. 여기서 벡터 타임스탬프는 프로세스의 수에 따라 차원(dimension)을 결정하고 각 차원은 개별적인 프로세스를 나타낸다. 또한 중첩된 프로세스에서는 인과관계 순서화를 위하여 CS(Causal State)를 유지한다. CS란 중첩된 프로세스에서 자신이 속한 그룹중 가장 최근에 전달된 메세지의 벡터 타임스탬프이다. 중첩된 프로세스에서 메세지를 전송할 때 CS의 정보 내용에 근거하여 CI(Causal Information)를 같이 전송한다. 제안된 메카니즘은 그림 4와 같이 각기 국부적인 벡터 타임스탬프로서 멀티캐스트를 수행하고 중첩된 프로세스에서만 CS를 유지한다. 프로세스 P2는 m_1을 수신하여 응용 프로세스로 전달하면 CS에 m_1에 대한 타임스탬프를 저장한다. 이때 응용 서비스에 m_2를 전달하면 CS는 m_2에 대한 타임스탬프로 갱신된다. 또한 m_3를 전송하고자 할시에는 가장 최근에 저장된 CS의 내용, 즉 m_2에 대한 벡터 타임스탬프를 CI에 입력한 후 m_3와 함께 전송한다. 프로세스 P3는 먼저 CI를 확인하고 전송된 메세지의 지연 여부를 결정한 후 m1, m2가 전달되면 m3에 대한 벡터 타임스탬프를 이용하여 응용 프로세스에게 전달한다.

구체적인 알고리즘을 살펴보면 다음과 같다.

⊙ 중첩되지 않은 프로세스의 경우

송신측:

(1) VT(P_i)[i] = VT(P_i)[i] + 1 /* 송신자 P_i에 대한 벡터 타임스탬프를 갱신한다. */

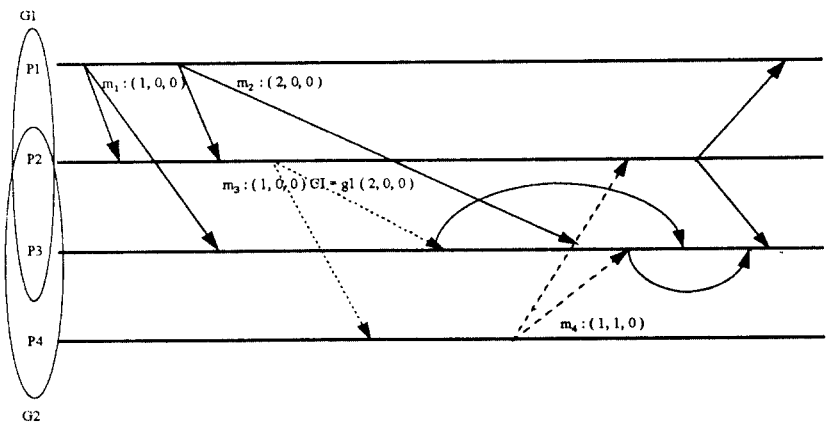


그림 4. 제안된 알고리즘의 예
Fig. 4. Example of the proposed algorithm

(2) 전송 메세지에 타임스탬프를 piggybacking하여 전송한다.

수신측:

(1) 프로세스 P_i로부터 VT(m)와 함께 메세지 m을 받는다.

(2) 프로세스 P_i가 아닌 프로세스 P_j는 다음의 조건이 만족할 때까지 메세지의 전송을 지연시킨다.

$$\forall k: 1...n \begin{cases} VT(m)[k] = VT(p_j)[k] + 1 \text{ if } k = i \\ VT(m)[k] \le VT(p_j)[k] \quad \text{otherwise} \end{cases}$$

VT(m)[k]: 메세지 m에 대한 프로세스 k의 벡터 타임스탬프

$VT(P_j)[k]$: 프로세스 P_j가 지닌 프로세스 k의 벡터 타임스탬프

(3) 프로세스 P_j는 메세지를 전달 후에 $VT(P_j)$를 다음의 절차에 따라 갱신한다.

$$\forall k \in 1...n: VT(p_j)[k] = \max(VT(p_j)[k], VT(m)[k])$$

⊙ 중첩된 프로세스의 경우

송신측:

(1) $VT(P_i)[i] = VT(P_i)[i] + 1$ /* 로컬 그룹에 대한 벡터 타임스탬프를 증가시킨다. */

(2) CI ← CS /* 자신의 causal 상태를 CI에 입력한다. */

(3) 메세지 m과 함께 VT, CI를 piggybacking 하여 전송한다.

수신측:

(1) 프로세스 P_i로부터 메세지와 함께 CI, VT를 받는다.

(2) 수신된 메세지의 CI에 해당하는 메세지가 전달되었는 가를 확인한 후 지연 여부를 결정한다. 만약 이전의 메세지가 모두 전달되었으면, 다음의 전송 조건이 만족할때까지 지연시킨다.

$$\forall k: 1...n \begin{cases} VT(m)[k] = VT(p_j)[k] + 1 \text{ if } k = i \\ VT(m)[k] \le VT(p_j)[k] \quad \text{otherwise} \end{cases}$$

(3) CS에 그룹별로 최근에 전송된 메세지의 타임스탬프를 저장한다.

(4) 메세지를 전달 후 다음과 같이 그룹별로 벡터 타임스탬프 값을 갱신한다.

$$\forall k \in 1...n: VT(p_j)[k] = \max(VT(p_j)[k], VT(m)[k])$$

Ⅳ. 인과관계에 따른 순서화 증명

본 절에서는 제안된 알고리즘이 중첩이 허용된 그룹 환경에서 인과관계 순서화를 만족하는 가에 대해 두단계를 이용하여 증명하고자 한다. 즉, 안전성(safety)으로서 인과관계 순서화가 만족되는 가를 확인하고 활성(liveness)으로서 멀티캐스트 메세지는 무한히 지연되지 않는다는 것을 보이고자 한다. 증명을 위하여 통신환경이 무손실(lossless), 활성화(live) 상태인 것으로 가정하였다.

4.1 안전성(safety)

[Theorem 1] 정의 2의 인과관계 순서화 m→m′는 모든 중첩된 프로세스에서 만족된다.

증명: 그림 4에서 m_2, m_3를 m과 m′라 하자. 그런 후 그림 4에서 프로세스 P3가 m과 m′을 받았다면 논리적인 유도로 인해 m보다 m′이 먼저 전달될 수 없음을 보이고자 한다. m_1과 m은 국부적인 타임스탬프로서 m이 m_1보다 먼저 응용 프로세스로 전달될 수 없다. 먼저 m→m′ 사이에 아무런 메세지가 없을 때를 살펴보면 프로세스 P2는 정의 1의 (1)에 따라 m의 수신 이벤트가 먼저 발생하므로 m→m′ 이다. 여기서 m′은 알고리즘에 따라 CS의 내용을 입력한 CI와 함께 전송된다. 무손실, 활성화 통신환경을 가정하였으므로 프로세스 P3는 결국 m, m′을 받는다. 알고리즘에 따라 m′에는 CI가 포함되어 있으며 여기에는 m에 대한 벡터 타임스탬프가 존재한다. 그러므로 m′은 m보다 먼저 전달될 수 없다. 또한 m→m′ 사이에 k개의 메세지(m_1, m_2,...,m_k)가 있는 경우 m→m′ 사이의 인과관계 순서화는 만족된다고 가정하고 k + 1인 메세지의 경우를 살펴보겠다. 여기서는 그림 5와 같은 두 가지 형태가 있다.

그림 5-(a)의 경우 m_k는 m이 전달된 후에 바로소 전달 가능하다. 또한 mk→m′이며 m_k는 m′의 CI에 속하므로 m′은 결코 m_k이전에 전달될 수 없다. 그림

5-(b)의 경우에는 중간에 각기 동시적인 메세지(m_1... m_k)가 있으며 이는 m이후에 전달 가능하다. 여기서 어느 한 프로세스에 동시적인 메세지가 모두 전달되었다면, CS에 이들 모두의 타임스탬프를 저장한 후 m′을 전송할때 CI에 입력한 후 piggybacking하여 전송한다. 이때 m′을 받은 프로세스는 CI가 가리키는 모든 메세지가 전달될 때까지 기다린다. 그러므로 메세지 m_{k+1}의 경우에도 인과관계 순서화는 만족된다.

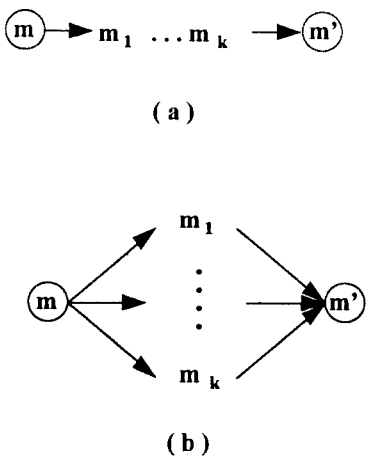


그림 5. 메세지 m과 m′사이의 인과 관계
(a) 유도적인 관계 (b) 동시적인 관계
Fig. 5. Causal relation between message m and m′
(a) Inductive relation (b) concurrent relation

4.2 활성(liveness)

[Theorem 2] 중첩된 프로세스에서 모든 메세지는 결국 전달된다.

증명 : 먼저 메세지 m이 중첩된 프로세스에게 전달될 수가 없다고 가정한다. 이때 중첩된 프로세스에게 메세지 m이 도착하지 않았거나, 인과관계 때문에 전달이 지연이 된 인과관계 m′→m인 메세지 m′가 반드시 존재한다. 모든 메세지는 무손실, 활성화 통신 환경에서 중첩된 프로세스에게 도착된다는 가정아래, m′은 중첩된 프로세스에게 도착되고 m″→m′인 m″도 도착된다. 마찬가지로 m도 도착하게 되며, 이때 m 이전에 도착해야 할 메세지는 유한개이며→은 비순환이므로 CI에 의해 모두 인과관계 순서화를 만족하며 전달된다. 이는 가정에 위배되므로 중첩된 프로세스에게 모든 메세지는 전달된다.

V. 성능 평가

본 장에서는 제안된 알고리즘에 대해 전체 순서화 제공시 Molina and Spauster 알고리즘과 비교하였으며, 또한 인과 관계 순서화 제공시 확장된 CBCAST 알고리즘과 비교하였다. 전체 순서화에 대해서 점대점(point-to-point) 네트워크 환경과 브로드캐스트 네트워크 환경에 대해 멀티캐스트에 필요한 메세지의 수와 그에 따른 지연시간을 비교하였다. 여기서는 acknowledgement, 네트워크 구성, 폭주, 라우팅 등은 고려하지 않았다. 시뮬레이션을 통해 중심 프로세스의 생성을 관찰하였으며, Molina and Spauster 알고리즘과는 멀티캐스트에 필요한 메세지의 수에 따른 load 면에서 비교하였다. 인과관계 순서화에 대해서는 중첩된 그룹 환경에서 한 그룹에 두개 이상의 프로세스가 동일한 그룹에 중첩되었을 시 필요한 메세지 오버헤드의 크기를 비교하였으며, 또한 시뮬레이션을 통해 이를 확인하였다.

5.1 전체 순서화에 대한 성능 평가

전체 순서화 제공시 다중 그룹 순서화를 위해 멀티캐스트 통신에 필요한 메세지의 수를 N이라 하고, 중첩된 그룹 환경에서 멀티캐스트 시작으로부터 모든 프로세스가 메세지를 수신할 때까지의 시간을 T라고 한다. 여기서 송신측이 멀티캐스트를 위해 네트워크에 송신할 때까지의 내부 처리 시간을 P, 메세지가 네트워크를 통해 목적 프로세스에게 도달할 때까지의 지연 시간을 L이라 한다.

먼저 점대점 네트워크에 대해 살펴보면, 점대점 네트워크에서는 한 송신자가 n개의 프로세스에게 멀티캐스트를 수행할 경우 각각의 수신자에게 전송해야 하므로 n개의 메세지가 필요하다. 여기서 간단한 수식 비교를 위해 P와 L은 상수라고 가정한다. 표 1은 메세지의 수와 총 지연 시간에 대해 Molina and Spauster 알고리즘과 제안된 알고리즘을 비교한 것이다. 한 프로세스가 n 개의 프로세스에게 멀티캐스트를 수행할 때 필요한 메세지의 수를 살펴보면, Molina and Spauster 알고리즘은 만약 불필요한 프로세스를

거칠 경우 $n+\varepsilon$이고, 그렇지 않을 경우는 n이다. 여기서 ε은 메세지 전달시 거치는 불필요한 프로세스의 갯수이므로 $n \le N \le n+\varepsilon$이다. 그러나 제안된 알고리즘에서는 불필요한 프로세스를 거치지 않으므로 $N=n$이다.

총 지연시간의 경우에 대해 살펴보면 다음과 같다. Molina and Spauster 알고리즘의 경우 목적 그룹의 프로세스중 멀티캐스트 트리의 최상위에 위치한 프로세스로부터 모든 목적 그룹 멤버까지 가장 긴 경로를 d로 한다. 이 경우, 최대 지연은 송신측으로부터 목적 그룹중 최상위에 위치한 프로세스까지의 지연과 최상위의 프로세스로부터 가장 긴 경로를 통해 전송하는 지연의 합이다. 여기서 송신자로부터 최상위 프로세스까지의 지연은 $L+P$이며, 최상위 프로세스로부터 목적 그룹까지 가장 긴 경로 d를 가진 프로세스까지의 지연은 $d \times L + (n-1+\varepsilon) \times P$이다. 그러므로 최대 지연은 $(d+1) \times L + (n+\varepsilon) \times P$이다. 최소 지연은 최상위 프로세스가 멀티캐스트를 수행할 경우이며, 이 값은 $d \times L + (n+\varepsilon) \times P$이다. 제안된 알고리즘의 분석을 위해 전체 순서화가 필요한 프로세스의 수를 x라 한다. 여기서 최대 지연은 보통의 프로세스로부터 중심프로세스까지의 지연 $L+P$, 중심프로세스에서 전담 프로세스와 나머지 목적 그룹까지의 지연 $L+(n-x)$, 전담 프로세스로부터 전체 순서화가 필요한 프로세스까지의 지연 $L+(x-1) \times P$을 합한 것이다. 그러므로 총 지연은 $3 \times L + n \times P$이다. 최소 지연으로는 중심 프로세스가 송신측이고 전체 순서화가 필요한 프로세스가 없을 경우이며, 이 경우 총 지연은 $L + n \times P$이다. 즉 제안된 알고리즘에서는 d와 ε의 영향을 받지 않는다.

두번째로 Ethernet과 같은 브로드캐스트 네트워크에 대해 두가지의 알고리즘을 비교해 보겠다. 여기서 브로드캐스트 네트워크의 특성을 살펴보면 다음과 같다. 즉, 한 송신측이 메세지를 전송할 경우 네트워크에 연결된 모든 수신측은 한번에 수신하게 된다. 그러므로 멀티캐스트 통신을 위해서 각 노드는 메세지의 목적지인가를 확인해야 한다.

브로드캐스트 네트워크에서의 두 알고리즘의 비교는 표 2와 같다. 점대점 네트워크에서와 마찬가지로 먼저 N에 대해 Molina and Spauster 알고리즘의 경우를 살펴보면, 최대 메세지 수는 송신측으로부터 그룹의 멤버중 멀티캐스트 트리의 최상위 프로세스까지 전송되는 메세지의 수와 최상위 프로세스로부터 가장 긴 path를 통해 목적 프로세스에게 전송되는 메세지의 수를 더한 값이다. 즉, 멀티캐스트에 필요한 최대 메세지의 수은 $d+1$이다. 또한 최소 메세지 수는 점대점 네트워크와 같은 방법으로 하면, d가 된다. 이에 반해 제안된 알고리즘의 경우 멀티캐스트에 필요한 최소 메세지 수로서 중심 프로세스가 그룹의 멤버에게 멀티캐스트를 수행할 때 필요한 메세지 수이며, 최대 메세지 수는 보통의 프로세스에서 중심 프로세스까지의 전달, 중심 프로세스에서 전담 프로세스까지의 전달, 전담 프로세스에서 전체 순서화를 요하는 프로세스까지의 전달에 필요한 메세지 수의 합이다. 즉, 최대 메세지 수는 3이 된다.

두번째의 총지연 시간에 대해 살펴보면, 브로드캐스트 네트워크 특성상 메세지의 목적지인가를 확인하는 시간이 필요하므로 이를 C라 한다. Molina and Spauster 알고리즘의 경우 최대 지연은 송신측으로부터 최상위 프로세스까지의 지연 $(P+L+C)$과 가장 긴 경로를 통한 지연 $d \times (P+L+C)$의 합이다. 최소 지연으로는 최상위 프로세스에서 직접 전송하는 지연값 $d \times (P+L+C)$이다. 제안된 알고리즘의 경우에 대해서도 마찬가지의 방법을 적용하면, 최소 지연은 $(P+L+C)$이고 최대 지연은 $3 \times (P+L+C)$이다.

시뮬레이션을 통해 고정된 총 프로세스의 수, 그룹의 크기, 그룹의 수에 대해 각 요소를 변화시키면서 발생되는 트리의 수(중심 프로세스의 수)와 멀티캐스

표 1. 점대점 네트워크에서 두 알고리즘의 비교
Table 1. Comparison of two algorithms in a point-to-point network

	Molina and Spauster 알고리즘	제안된 알고리즘
메세지의 수(N)	$n \le N \le n+\varepsilon$	$N=n$
총 지연 시간(T)	$d \times L + (n+\varepsilon) \times P \le T \le (d+1) \times L + (n+\varepsilon) \times P$	$L + n \times P \le T \le 3 \times L + n \times P$

표 2. 브로드캐스트 네트워크에서 두 알고리즘의 비교
Table 2. Comparison of two algorithms in a broadcast network

	Molina and Spauster 알고리즘	제안된 알고리즘
메세지의 수(N)	$d \le N \le d+1$	$1 \le N \le 3$
총 지연 시간(T)	$d \times (P+L+C) \le N \le (d+1) \times (P+L+C)$	$(P+L+C) \le N \le 3 \times (P+L+C)$

트 수행 완료를 위해 처리해야 할 메세지의 load를 측정하였다. 여기서 각 요소의 범위를 살펴보면, 고정된 프로세스의 갯수는 20개에서 500개, 그룹의 크기는 5에서 40, 그룹의 수는 10에서 40이다.

그림 6은 고정된 그룹의 크기를 설정한 후 각 그룹의 수에 대해 총 프로세스의 수를 변화시키고 이 결과로서 발생되는 트리의 수를 평균한 값이다. 그림 7은 고정된 총 프로세스의 수를 설정한 후, 각 그룹의 수에 대해 그룹의 크기를 변화시키면서 발생되는 트리의 수를 평균한 값이다. 그림 6은 그룹의 크기를 5로, 그림 7은 프로세스의 수를 200개로 가정하여 시뮬레이션한 결과이다. 그림 6에서 알 수 있듯이 존재하는 총 프로세스의 수와 그룹의 수가 많아질수록 중심 프로세스의 수가 많이 발생함을 볼 수 있다. 그림 7에서는 그룹의 수가 증가할수록, 그룹의 크기가 적을수록 중심 프로세스의 수는 적어지는 것을 볼 수 있다.

그림 8과 9의 시뮬레이션 결과는 모든 그룹의 멤버가 한번씩 메세지를 멀티캐스트한다고 가정하였을 때, 각 멀티캐스트 메세지가 모든 목적 그룹에 도착

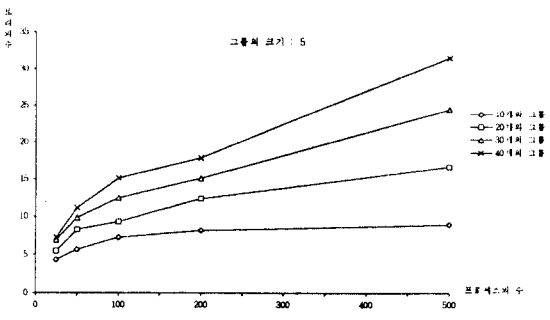


그림 6. 프로세스 수의 변화에 따른 트리 수
Fig. 6. Number of trees with changing the process number

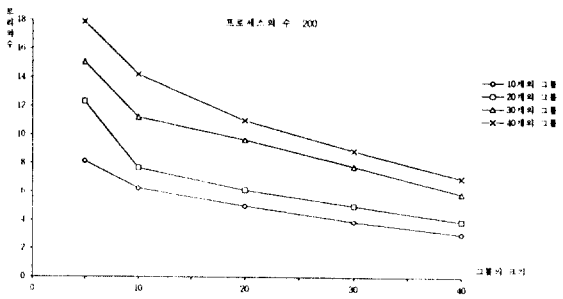


그림 7. 그룹 크기의 변화에 따른 트리 수
Fig. 7. Number of trees with changing the group size

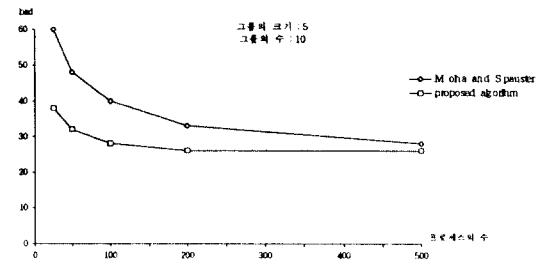


그림 8. 프로세스 수의 변화에 따른 두 알고리즘의 load 비교
Fig. 8. Comparison of the loads of two algorithms with changing the process number

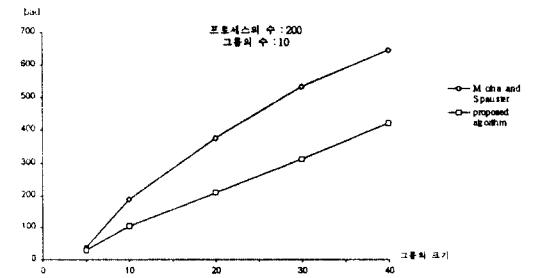


그림 9. 그룹 크기의 변화에 따른 두 알고리즘의 load 비교
Fig. 9. Comparison of the loads of two algorithms with changing the group size

하기 위해 얼마나 많은 메세지가 처리되어야 하는지를 비교한 값이다. 여기서의 load는 멀티캐스트 통신에서 송신 및 수신에 포함되는 메세지의 갯수이다. 그림 8은 그림 6과 같은 조건에서의 시뮬레이션 결과이며, 그림 9는 그림 7와 같은 조건에서의 시뮬레이션 결과이다. 그림 8과 그림 9에서 볼 수 있듯이 제안된 알고리즘이 멀티캐스트를 수행할 때 적은 메세지의 처리를 필요로 한다. 그러나 그림 8에서는 프로세스의 수가 많아질수록, 그림 9에서는 그룹의 크기가 적을수록, 두 알고리즘의 load는 비슷함을 볼 수 있다. 그러므로 제안된 알고리즘은 그룹의 크기가 결정되었을 때 총 프로세스의 수가 적을수록, 총 프로세스의 수가 결정되었을 때 그룹의 크기가 커질수록 Molina and Spauster 알고리즘보다 유리함을 볼 수 있다.

5.2 인과관계 순서화에 대한 성능 평가

본 절에서는 인과관계 순서화를 제공할 때, 메세지의 오버헤드 측면에서 제안된 알고리즘과 확장된 CBCAST 알고리즘을 비교하였고 시뮬레이션을 통해 이를 확인하였다. 표 3은 두 알고리즘에 대해 중첩된 그룹 환경에서 인과관계 순서화를 위해 필요한 메세지 오버헤드의 크기를 나타내었다. 여기서 n은 그룹의 크기($n \geq 2$), m은 한 그룹에서 중첩이 일어난 프로세스의 수($m \geq 2$), k는 중첩 횟수($k \geq 2$)이다. 확장된 CBCAST 알고리즘의 경우, 그룹에서 중첩된 프로세스 때문에 중첩되지 않은 프로세스가 메세지를 전송할 시에도 인과관계 순서화를 위해 다른 그룹에 대한 제어 정보를 같이 전송해야 한다. 그러므로 그룹의 크기가 n일때 부가적인 제어 정보의 총합은 $n(n-1)$이며, 이 값은 중첩 횟수가 증가함에 따라 $(k-1)$에 비례하며 증가한다. 그러나 중첩된 프로세스의 수가 두개 이상이면 어느 프로세스라도 그룹의 모든 멤버에게 부가적인 제어 정보를 전송하므로, 중첩된 프로세스의 수와는 관계없다. 제안된 알고리즘의 경우, CI는 중첩된 프로세스에서만 메세지의 전송시에 부가적인 제어 정보를 필요로 한다. 그러므로 메세지 오버헤드는 중첩 횟수에는 상관없으며, 중첩된 프로세스의 수에 비례적으로 증가한다. 또한 CI는 그룹의 크기가 클수록 다른 프로세스의 정보가 많아지게 된다. 즉 그룹의 크기에 비례한다. 그러므로 중첩된 프로세스에서 전송할 메세지의 오버헤드는 $m(n-1)$이다.

표 3. 두 알고리즘의 메세지 오버헤드 비교

Table 3. Comparison of the message overhead of two algorithms

	확장된 CBCAST 알고리즘	제안된 알고리즘
필요한 오버헤드	$(k-1)(n^2-n)$	$m(n-1)$

시뮬레이션을 통해 총 프로세스의 수를 변화시키면서 인과관계 순서화를 위한 메세지의 오버헤드를 비교하였으며, 그 결과는 그림 10과 같다. 시뮬레이션 결과에서 알 수 있듯이 제안된 알고리즘이 확장된 CBCAST 알고리즘에 비해 인과관계 순서화를 위한 부가적인 메세지가 적으나, 총 프로세스의 수가 많아질수록 두 방식의 성능이 큰 차이가 없음을 볼 수 있다. 그러므로 전체 순서화와 마찬가지로 제안된 알고리즘은 그룹의 크기가 결정되었을 때 총 프로세스의 수가 적을수록 CBCAST 알고리즘보다 더 유리함을 알 수 있다.

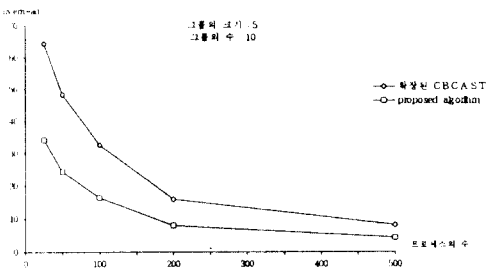


그림 10. 두 알고리즘의 메세지 오버헤드 비교

Fig. 10. Comparison of the message overhead of two algorithms

Ⅵ. 결 론

본 논문에서는 하나의 프로세스가 여러 그룹에 속하는 즉, 중첩된 프로세스 그룹 환경에서 기존의 멀티캐스트 알고리즘을 개선한 새로운 알고리즘을 제시하였다. 기존의 알고리즘과 달리 중첩된 프로세스 그룹 환경에서 그룹통신을 효율적으로 수행 할 수 있도록 프로세스 그룹의 정보를 이단 트리로 작성하는 방식을 제안하였다. 이를 통해 기존의 알고리즘에서

불필요한 프로세스를 거치는 단점을 해결하므로써 메세지의 전달시간을 단축하였다. 또한, 그룹통신에 전체 순서화뿐만 아니라 인과관계 순서화의 기능을 부여함으로써 적은 지연을 요하는 메세지의 전달이 가능하도록 제안하였으며, 순서화에 요구되는 오버헤드의 크기도 줄어들게 하였다.

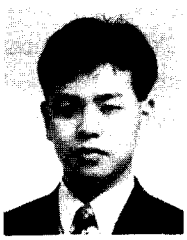
분산응용 서비스들이 복잡해지고 고도화함에 따라 멀티캐스트 환경에서 프로세스 그룹의 중첩이 빈번히 일어나게 되므로, 본 논문에서 제시한 새로운 알고리즘 방식을 적용하면 멀티캐스트를 효율적으로 수행할 수 있을 것이다. 향후 연구과제로는 그룹통신에서의 효과적인 ACK 방식, 오류 발생시 효율적인 대처방안 등이 연구되어야 하겠다.

참 고 문 헌

1. L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System", Communications of The ACM, Vol. 21, No. 7, pp 558-565, July 1978.
2. M. Frans Kaashoek, Andrew S. Tanenbaum, Susan Flynn Hummel, and Henri E. Bal, "An Efficient Reliable Broadcast Protocol", Operating Systems Review, Vol. 23, pp 5-19, Oct. 1989.
3. Bala Rajagopalan, "Reliablility and Scaling Issues in Multicast Communicaton", ACM SIGCOMM'92, pp 188-198, Aug. 1992.
4. Erwin Mayer, "An Evaluation Framework for Multicast Ordering Protocols", ACM SIGCOMM'92, pp 177-187, Aug. 1992.
5. J. Chang and N. Maxemchuck, "Reliable Broadcast Protocols", ACM Transaction on Computer System, Vol. 2, No. 2, pp 251-273, Aug. 1984.
6. Kenneth P. Birman and Thomas A. Joseph, "Reliable Communication in the Presence of Failures", ACM Transaction on Computer Systems Vol. 5, No. 1, pp 47-76, Feb. 1987.
7. M. Frans Kaashoek, Andrew S. Tanenbaum, "Fault Tolerance Using Group Communication", Operating Systems Review, Vol. 30, pp 71-74, Aug. 1989.
8. Kenneth P. Birman and Thomas A. Joseph, "Exploiting Replication in Distributed Systems". In Sape Mullender, editor, Distributed Systems, ACM Press, Addison-Wesley. pp. 319-368, New York, 1989.
9. Hector Garcia-Molina and Annemaria Spauster, "Message Ordering in Multicast Environment", IEEE Communications Magazine, pp 354-361, June 1988.
10. Kenneth P. Birman and Thomas A. Joseph, "Exoting Virtual Synchrony in Distributed Systems", Proceedings of The 11th ACM Symposium on Operating Systems Principles, pp. 123-138, Nov. 1987.
11. Kenneth P. Birman, Robert Cooper, and Barry Gleeson, "Programming with Process Groups: Group and Multicast Semantics", Techinical Report, Cornell University Computer Science Department, Sept. 1991.
12. Kenneth P. Birman, Andre Schiper and Pat Stephenson, "Lightweight Causal and Atomic Group Multicast", ACM Transaction on Computer Systems, Vol. 9, No 3, pp 242-271, 1991.
13. P. M. Melliar-Smith, Louise E. Moser, and Vivek Agrawara "Broadcast Protocols for Distributed Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 1, No. 1, pp 17-25, Jan. 1990.
14. Adrian Segall, and Baruch Awerbuch, "A Reliable Broadcast Protocol", IEEE Transactions on Communications, Vol. Com-31, No. 7, pp 896-901, July 1983.
15. Rosario Aiello, Elena Pagani, and Gian Paolo Rossi "Causal Ordering in Reliable Group Communications", ACM SIGCOMM'93, pp 106-115, Sept. 1993.
16. J. Crowcroft and K. Paliwoda, "A Multicast Transport Protocol", ACM SIGCOMM, Vol. 18. No. 4, pp. 247-256, Aug. 1988.
17. Alan O. Freier, Keith Marzullo, "MTP:An Atomic Multicast Transport Protocol", Technical

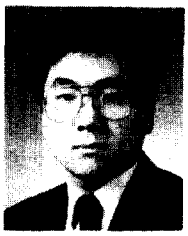
Report, Cornell University Computer Science Department, July 1990.

18. Terunao Soneoka, and Toshihide Ibaraki "Logically Instantaneous Message Passing in Asynchronous Distributed Systems", IEEE Transactions on Computers, Vol. 43, No. 5, pp 513-527, May 1994.
19. Nasr E. Belkeir and Mustaque Ahamad, "Low Cost Algorithms for Message Delivery in Dynamic Multicast Groups", Proceedings of the 9th ICDCS, pp. 110-117, 1989.
20. S. Deering, "Host Extension for IP Multicasting", RFC 1112, Standford University, May 1989.



권 봉 경(Bong-Kyoung Kwon) 정회원
1995년 : 광운대학교 전자통신공학과 학사
1995년~현재 : 광운대학교 전자통신공학과 석사과정
※주관심 분야 : 멀티미디어 통신, 분산처리

정 광 수(Kwang-Sue Chung) 정회원
1981년 : 한양대학교 전자공학과 학사
1983년 : 한국과학기술원 전기 및 전자공학과 석사
1991년 : 미국 University of Florida 전기공학과 박사(컴퓨터공학 전공)
1983년~1993년 : 한국전자통신연구소 선임연구원
1991년~1992년 : 한국과학기술원 전산학과 대우교수
1993년~현재 : 광운대학교 전자통신공학과 조교수
※주관심 분야 : 멀티미디어 통신, 컴퓨터통신, 분산처리



현 동 환(Don-Whan Hyung) 정회원
1989년 : 광운대학교 전자계산기공학과 학사
1991년 : 광운대학교 전자계산기공학과 석사
1991년~현재 : 한국전자통신연구소 멀티미디어표준연구실 선임연구원
※주관심 분야 : 멀티미디어 통신, 멀티미디어 응용서비스

함 진 호(Jin-Ho Hahm) 정회원
1982년 : 한양대학교 전자공학과 학사
1984년 : 한양대학교 전자통신공학과 석사
1984년~현재 : 한국전자통신연구소 멀티미디어표준연구실 실장
※주관심 분야 : 멀티미디어 통신, 멀티미디어 시스템, 멀티미디어 응용서비스