논문 99-24-7A-13 한국통신학회논문지 '99-7 Vol.24 No.7A

중첩된 그룹 환경에서의 효율적인 인과관계 순서화 알고리즘

정회원 권 봉 경*, 정 광 수**

An Efficient Causal Ordering Algorithm in Overlapping Groups

Bong-kyoung Kwon*, Kwang-sue Chung** *Regular Members*

요 약

본 논문에서는 임의의 프로세스가 여러 그룹에 속하는 중첩된 프로세스 그룹 환경에 효과적으로 적용할 수 있는 인과관계 순서화 알고리즘을 제시하였다. 본 알고리즘은 네트워크 구성 형태에 따라 선택할 수 있도록 제안하였다. 즉, 브로드캐스트 네크워크에서는 수신자 선택 알고리즘을 제안하였으며, 점대점 네트워크에서는 송신자 선택 알고리즘을 제안하였다. 각 알고리즘은 순서화에 요구되는 메시지 오버헤드를 줄이기 위해 그룹별로 불필요한 벡터 타임스탬프를 제거하였으며, 메시지 오버헤드를 최소화하기 위해 국부적으로 유지하고 있는 다른 프로세스와 다른 그룹의 정보를 이용하여 압축하도록 하였다. 각각의 새로운 인과관계 순서화 알고리즘을 논리적으로 증명하였고, 시뮬레이션을 통해 기존의 인과관계 순서화 알고리즘과의 성능을 비교하였다.

ABSTRACT

In this paper, we proposed a causal ordering algorithms which is efficiently applicable to overlapped process group environments where one process may belong to several process groups. The ones is proposed to choose with topology of the network. We proposed receiver select algorithm in broadcast network, sender select algorithm in point-to-point network. Each algorithms removes unnecessary vector timestamps to reduce the message overhead required for the causal ordering. And, compressed vector timestamps using the locally maintained vector timestamp information of other processes and other groups to minimize the message overhead. 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.

I. 서 론

통신망의 대역폭이 증대됨에 따라 분산 멀티미디어 데이터베이스 시스템, 화상회의, CSCW (Computer Supported Cooperative Work) 고장허용 시스템(fault tolerant system), 등 다양한 분산 응용 서비스들이 등장하고 있다. 이러한 응용 서비스들은 기존과는 다른 새로운 형태의 통신 기능을 요구하는데, 이러한 기능 중에 하나가 멀티캐스트(multicast)이다. 멀티캐스트란 동일한 정보를 하나의 송신지로부터 다수의 목적지로 전달하는 통신 방식으로, 유니캐스트(unicast)와 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 곳으로

* 대우통신 교환연구단(kwonbk@adam.dwt.daewoo.co.kr)
** 광운대학교 전자공학부(kchung@daisy.kwangwoon.ac.kr)
논문번호 : 98443-1009, 접수일자 : 1998년 10월 9일
※ 본 연구는 한국과학재단 장기기초 연구(97-01-00-12-01-5)와 광운대학교 교내학술연구비의 지원에 의해 수행되었음.

의 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 그룹들간의 그룹 통신에 관한 연구[3][5][6][13][14]와 하나의 프로세스가 여러 그룹에 속한 경우에 있어 메시지 송수신에 관한 연구[3][5][7], 프로세스 그룹의 동적인 변화에 관한 연구[3][21] 등이 있다. 이와 같은 프로세스 그룹의 통신에서는 메세지 순서화가 매우 중요하며, 이에 관한 연구도 매우 활발하다[1][3][5][7][15][17][20].

이외에도 멀티캐스트에 관한 연구로는 송수신하는 메시지의 효율성 문제[2][8][9], 멀티캐스트 도중에 발생하는 오류에 대처하는 연구[11][12], IP를 이용한 멀티캐스트 방법[22], 부분적인 순서화를 이용하여 그래픽하게 전체적인 순서화를 만드는 방법[15], 전송계층을 이용하여 멀티캐스트하는 방법[18][19] 등이 활발히 진행되고 있다.

본 논문에서는 임의의 프로세스가 여러 그룹에 속하는 중첩된 프로세스 그룹 환경에서 단일 그룹에만 속한 프로세스 뿐만 아니라 여러 그룹에 속한 프로세스에게도 효과적으로 적용할 수 있는 인과관계 순서화 알고리즘을 제시하였다. 또한, 제안된 알고리즘은 네트워크 구성 형태에 따라 선택할 수 있도록 제안하였다. 즉, 브로드캐스트 네트워크에서는 수신자 선택 알고리즘을 제안하였으며, 점대점 네트워크에서는 송신자 선택 알고리즘을 제안하였다. 본 논문의 구성은 제2장에서 인과관계 순서화를 기술하기 위한 프로세스 그룹 및 통신 모델을 설명하였고, 제3장에서는 중첩된 그룹 환경에서 네트워크 구성 형태에 관계없이 모든 프로세스들에게 효율적으로 적용가능한 수신자 선택 알고리즘 및 송신자 선택 알고리즘을 기술하였다. 제4장에서는 제안된 알고리즘을 논리적으로 증명을 하였으며, 제5장에서는 제안된 알고리즘과 기존의 알고리즘을 시뮬레이션을 통해 성능 평가를 하였다. 그리고 제6장에서 결론을 맺는다.

Ⅱ. 인과관계 순서화 기술

1. 프로세스 그룹 및 통신 모델

본 논문의 프로세스간 통신에서는 공통의 공유 메모리, 클락은 존재하지 않으며, 프로세스는 각기 다른 속도로 메세지를 송수신한다고 가정한다. 또한 응용 프로세스로의 메세지 전달 지연은 유한적이며 예측 불가능하다고 가정한다.

프로세스간에 협동(cooperation)은 그룹 개념으로서 이루어지며, 각 그룹은 고유한 하나의 이름을 갖는 프로세스의 집합으로 구성된다. 멀티캐스트 그룹 환경은 정적이며, 하나의 프로세스는 단일 그룹뿐만 아니라, 여러개의 그룹에도 속할 수 있다. 멀티캐스트 통신에서 메시지의 send, receive, deliver는 프로세스의 이벤트라 가정한다.

단일 그룹 g에 속한 프로세스 P_i에 의한 메세지 m의 전송은 $send_i(m)$으로 나타낸다. 해당 그룹의 모든 프로세스 P_j는 메세지 m을 수신하게 되며, $receiv_j(m)$로써 표시한다. 또한, P_j에 의한 메세지 m에 대한 상위 응용 프로세스에게로의 전달은 $deliv_j(m)$로 표시한다.

다중 그룹에 속한 프로세스 P_i에 의한 특정 그룹 g^a으로 메시지 m의 전송은 $send^a_i(m)$으로 나타낸다. 해당 그룹의 모든 프로세스 P_j는 메시지의 수신하며, $receiv^a_j(m)$로써 표시한다. 단일 그룹의 경우와 마찬가지로 상위 응용 프로세스에게로의 전달은 $deliv^a_j(m)$로 표시한다.

2. 인과관계 순서화

모든 프로세스에서 발생하는 send 이벤트와 deliv 이벤트를 고려한다. 이 이벤트들은 다음의 인과관계 순서화 관계에 의해 정형화된다. 먼저, 단일 그룹에서 비동기적으로 발생하는 이벤트에 대해 인과관계 순서화를 정의하며, 다중 그룹에 속한 프로세스들간의 인과관계 순서화도 정의한다.

정의 1 : 두 개의 이벤트 e_1 과 e_2에 대해, 다음의 조건중 하나라도 성립되면, $e_1 \rightarrow e_2$ 이다.

- $e_1 = send_i(m)$가 $e_2 = send_i(m')$보다 먼저 발생한 경우
- $e_1 = send_i(m)$, $e_2 = receiv_j(m)$인 경우
- $e_1 \rightarrow e'$, $e' \rightarrow e_2$인 이벤트 e'가 존재할 경우

정의에서는 이벤트 e에 관해 $e \not\rightarrow e$라고 가정한다(이벤트는 그 자신이 happened before 관계가 될 수 없다). 또한, 두 개의 이벤트 e_1, e_2가 $e_1 \not\rightarrow e_2$, 그리고 $e_2 \not\rightarrow e_1$ 일 경우 두 이벤트는 인과관계가 없는 동시적인 메시지이다.

정의 2 : $send^a_i(m) \rightarrow send^b_i(m')$이면, 그룹 g^a, g^b에 속한 모든 프로세스 P_k에서 $deliv^a_k(m) \rightarrow deliv^b_k(m')$이다.

Ⅲ. 새로운 인과관계 순서화 알고리즘

본 장에서는 단일 그룹 및 중첩된 그룹 환경에서의 멀티캐스트 수행시 메시지간 인과관계 순서화를 유지하기 위한 새로운 알고리즘을 제안하였다. 제안된 알고리즘은 기본적으로 논리적 시간이며 여러 스칼라 값(각 프로세스들의 논리적인 시간)들이 모인 벡터 타임스탬프를 이용하여 멀티캐스트를 수행한다. 제안된 알고리즘은 기술함에 있어서 다음의 사항에 대해 가정하였다. 첫째, 멀티캐스트 목적지는 단일 그룹으로 가정한다. 둘째, 여러 그룹에 속한 프로세스는 동시에 여러 그룹에 멀티캐스트를 수행할 수 없다고 가정한다. 셋째, 하부의 통신환경은 무손실(Lossless), 활성(Liveness) 상태로 가정하였다.

제안된 알고리즘은 프로세스간 송수신되는 메시지의 인과관계 순서화를 위해 필요한 메세지 오버헤드를 줄이기 위해 단계별로 제안하였다. 각 알고리즘의 첫 번째 레벨에서는 단일 그룹에만 속한 프로세스가 자신이 속한 그룹에 다중 그룹에 속한 프로세스가 존재할 경우, 기존의 알고리즘에 비해 불필요한 다른 그룹의 벡터 타임스탬프를 제거하였다. 두 번째 레벨에서는 메시지 오버헤드를 최소화하기 위해 국부적으로 다른 프로세스의 타임스탬프를 유지함으로써 송신 메시지를 압축할 수 있도록 제안하였다. 또한, 각 레벨의 알고리즘에 대해 네크워크 구성 형태와 메모리 크기에 따라 최적의 알고리즘을 선택할 수 있도록 제안하였다. 즉, 브로드캐스트 네트워크에서는 수신자 선택 알고리즘을 제안하였으며, 점대점 네트워크에서는 송신자 선택 알고리즘을 제안하였다.

1. 불필요한 타임스탬프 제거(레벨 1)

기존의 확장된 CBCAST 알고리즘[7]은 CBCAST 알고리즘에서 한 프로세스 그룹내에서 여러 그룹에 속한 프로세스가 존재하는 경우에도 인과관계 순서화를 만족시키기 위한 알고리즘이다. 즉, 한 프로세스 그룹내에서 각 프로세스들은 존재하는 그룹의 수와 같은 타임스탬프로서 멀티캐스트를 수행한다. 그러나, 단일 그룹에만 속한 프로세스는 자신의 그룹에 중첩된 프로세스가 존재할 경우 메시지간의 인과관계 순서화를 만족시키기 위해 불필요한 벡터 타임스탬프를 유지하게 된다. 그러나 제안된 알고리즘은 확장된 CBCAST 알고리즘과 달리 모든 프로세스는 자신이 속한 그룹의 수만큼의 벡터 타임스탬프를 이용하여 멀티캐스트를 수행한다. 여기서 중첩된 프로세스에서는 단일 그룹에 속한 프로세스와 달리 정의 2의 인과관계 순서화를 만족하기 위해 CS(Causal State)를 유지하며, 메시지를 전송할 경우 CS를 기반으로 한 CI(Causal Information)를 piggybacking한다. 제안된 알고리즘은 네트워크 형태에 따라 수신자 선택 알고리즘과 송신자 선택 알고리즘으로 나눈다. 각 알고리즘에 대해 단일 그룹에 속한 프로세스와 다중 그룹에 속한 프로세스로 나누어 기술한다.

1.1. 수신자 선택 알고리즘(Receiver-Select Algorithm)

다음은 수신자 선택 알고리즘을 기술하였으며, 그림 1은 제안된 알고리즘의 예를 보여준다.

단일 그룹에 속한 프로세스

송신절차(Process P_i)

1. $VT(P_i)[i] = VT(P_i)[i] + 1$
2. $send_i(m, VT(P_i))$

수신절차(Process P_j)

1. $receive_j(m, VT(m))$
2. delay m until

   $\forall k : 1, \ldots n, VT(m)[k] = VT(P_j)[k] + 1$ if $k = i$

   $VT(m)[k] \le VT(P_j)[k]$ otherwise
3. after delivering m

   $VT(P_j)[i] = \max(VT(P_j)[i], VT(m)[i])$

다중 그룹에 속한 프로세스

송신절차(Process P_i)

1. $VT^a(P_i)[i] = VT^a(P_i)[i] + 1$
2. $CI \leftarrow CS(VT^{a \cap b \neq \emptyset}(P_i))$
3. $send^b_i(m, VT^a(P_i), CI)$ to processes belong to g^b

수신절차(Process P_j)

1. $receive^a_j(m, VT(m), CI)$ or $(m, VT(m))$
2. if P_j receives (m, VT, CI)

   each $VT^{a \cap b \neq \emptyset}$ in CI

   wait until $\forall k : 1, \ldots n, VT^a(m)[k] \le VT^a(P_j)[k]$

   delay m until

   $\forall k : 1, \ldots n, VT^a(m)[k] = VT^a(P_j)[k] + 1$ if $k = i$

   $VT^a(m)[k] \le VT^a(P_j)[k]$ otherwise
3. after delivering m

   $VT^a(P_j)[i] = \max(VT^a(P_j)[i], VT^a(m)[i])$

$VT(P_i)$: 프로세스 P_i의 벡터 타임스탬프

$VT(P_i)[i]$: 프로세스 P_i의 i번째 타임스탬프 값

$VT(m)[k]$: 메세지 m의 k번째 타임스탬프 값

$VT^a(P_i)[i]$: 그룹 a에 대한 프로세스 P_i의 i번째 타임스탬프 값

$VT^{a \cap b \neq \emptyset}(P_i)$: 그룹 a를 제외한 그룹에 대한 프로세스 P_i의 벡터 타임스탬프

그림 1. 제안된 알고리즘의 예

브로드캐스트 네트워크 환경에서 수신자 선택 알고리즘을 살펴보면 다음과 같다. 단일 그룹에 속한 프로세스는 자신이 속한 그룹에 대한 타임스탬프로서 멀티캐스트를 수행하므로 확장된 CBCAST 알고리즘에 비해 인과관계 순서화에 필요한 메시지 오버헤드가 줄어들게 된다. 즉, 자신의 그룹에 중첩된 프로세스가 존재할 시 불필요하게 다른 그룹의 타임스탬프를 유지해야 하는 단점을 제거하였다. 그림 1을 살펴보면 P_1은 m_1을 전송하며, P_2는 m_1을 상위 응용 프로세스에게 전달한 후 그룹 g^2에게 m_2를 전송하게 된다. 이때 자신이 속한 g^1에 대한 벡터 타임스탬프를 CI에 저장한 후 m_2를 전송한다. 여기서 m_2에 대

한 타임스탬프를 살펴보면, P_2가 그룹 g^2에서는 첫 번째 프로세스로 인식되므로(1,0,0)이된다. 그러므로, P_3는 m_2를 먼저 수신하게 되더라도, CI의 정보에 따라 m_1의 도착을 기다린다. 그러므로, P_3는 m_1 다음에 m_2를 상위 응용 프로세스에게 전달하게 된다. P_3는 m_3, m_4를 전달한 후 m_5를 전송할 때, 자신이 속한 g^1, g^2에 대한 타임스탬프들을 CI에 저장하여 전송한다. 마찬가지로 P_2는 CI의 정보에 의해 m_4, m_5의 순서로 상위 응용 프로세스에게 전달한다.

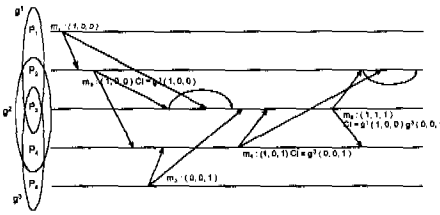


그림 1. 수신자 선택 알고리즘의 예

1.2. 송신자 선택 알고리즘(Sender Select Algorithm)

다음은 송신자 선택 알고리즘을 기술하였으며, 그림 2는 제안된 알고리즘의 예를 보여준다.

단일 그룹에 속한 프로세스

송신절차(Process P_i)
1. $VT(P_i)[i] = VT(P_i)[i] + 1$
2. send$_i$(m, $VT(P_i)$)

수신절차(Process P_j)
1. receive(m, $VT(m)$)
2. delay m until
$\forall k : 1, \ldots, n$, $VT(m)[k] = VT(P_j)[k] + 1$ if $k = i$
$VT(m)[k] \le VT(P_j)[k]$ otherwise
3. after delivering m
$VT(P_j)[i] = \max(VT(P_j)[i], VT(m)[i])$

다중 그룹에 속한 프로세스

송신절차(Process P_i)
1. $VT^x(P_i)[i] = VT^x(P_i)[i] + 1$
2. send$_i^x$(m, $VT^x(P_i)$) to processes belong to g^x
3. for destination processes in g^x except for processes belong to only g^x
destination process $\in g^x$
$CI \leftarrow CS(VT^{x\cap m}(P_i))$
send$_i^x$(m, $VT^x(P_i)$, CI) to destination process

수신절차(Process P_j)
1. receive$_j^x$(m, $VT(m)$, CI) or (m, $VT(m)$)
2. If P_j receives (m, VT, CI)
each $VT^{x\cap m}$ in CI
wait until $\forall k : 1, \ldots, n$, $VT^x(m)[k] \le VT^x(P_j)[k]$
delay m until
$\forall k : 1, \ldots, n$, $VT^x(m)[k] = VT^x(P_j)[k] + 1$ if $k = i$
$VT^x(m)[k] \le VT^x(P_j)[k]$ otherwise
3. after delivering m
$VT^x(P_j)[i] = \max(VT^x(P_j)[i], VT^x(m)[i])$

$VT(P_i)$: 프로세스 P_i의 벡터 타임스탬프
$VT(P_i)[i]$: 프로세스 P_i의 i번째 타임스탬프 값
$VT(m)[k]$: 메세지 m의 k번째 타임스탬프 값
$VT^x(P_i)[i]$: 그룹 a에 대한 프로세스 P_i의 i번째 타임스탬프 값
$VT^{x\cap m}(P_j)$: 그룹 a를 제외한 그룹에 대한 프로세스 P_j의 벡터 타임스탬프

점대점 네트워크 환경에서 송신자 선택 알고리즘을 살펴보면 다중 그룹에 속한 프로세스가 메시지를 전송할 시 그룹 정보를 이용하여 다른 타임스탬프를 piggybacking하여 전송한다. 그룹 정보는 초기에 멀티캐스트 그룹이 설정될 때 각 프로세스가 알고 있다. 그림 2를 살펴보면, 단일 그룹에 속한 P_1은 m_1을 g^1에 속한 프로세스에게 전송하며, m_2를 수신한 P_2는 g^2에게 멀티캐스트를 수행할 시 수신자 선택 알고리즘과 달리 그룹 정보를 이용하여 g^1에 속한 P_3에게는 CI로서 g^1에 대한 정보를 같이 전송한다. 그러나, g^1에 속하지 않은 P_4에게는 m_2에 대한 정보만 전송한다. 왜냐하면, P_4는 m_1을 수신하지 않기 때문이다. 그러므로 P_3는 CI의 정보를 이용하여 m_2보다 m_1을 먼저 응용 프로세스에게 전송하게 되며, P_4는 m_2만 응용 프로세스에게 전송한다. 나머지의 경우도 마찬가지로 동작한다. m_5에 대해서는 P_3가 P_2에게는 CI(g^1에 대한 타임스탬프)와 g^2에 대한 타임스탬프를 piggybacking하여 전송하며, P_4에게는 CI(g^2에 대한 타임스탬프)와 g^3에 대한 타임스탬프를 piggybacking하여 전송한다. 이와 같이 타임스태프 정보가 압축되므로 기존의 확장된 CBCAST 알고리즘보다 인과관계 순서화에 필요한 메시지 오버헤드가 줄어들게 된다.

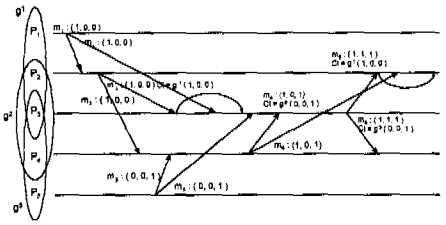


그림 2. 송신자 선택 알고리즘의 예

2. 개별적인 정보를 이용한 압축 알고리즘(레벨2)

레벨 1 알고리즘보다 더욱 메세지 오버헤드를 줄이기 위해, 그룹 정보뿐만 아니라 자신이 속한 그룹의 프로세스 정보를 이용한 알고리즘을 제안하였다. 단계 2 알고리즘은 다중 그룹에 속한 프로세스뿐만 아니라, 단일 그룹에 속한 프로세스도 통신 오버헤드를 줄일 수 있다. 레벨 1 알고리즘과 마찬가지로, 네트워크 환경에따라 수신자 선택 알고리즘과 송신자 선택 알고리즘을 제안하였다. 또한, 각 알고리즘에 대해 단일 그룹에 속한 프로세스와 다중 그룹에 속한 프로세스로 나누어 기술한다.

2.1. 수신자 선택 알고리즘

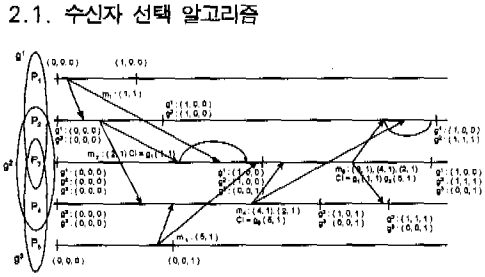


그림 3. 레벨 2 수신자 선택 알고리즘의 예

다음은 수신자 선택 알고리즘을 기술하였으며, 그림 3은 제안된 알고리즘의 예를 보여준다.

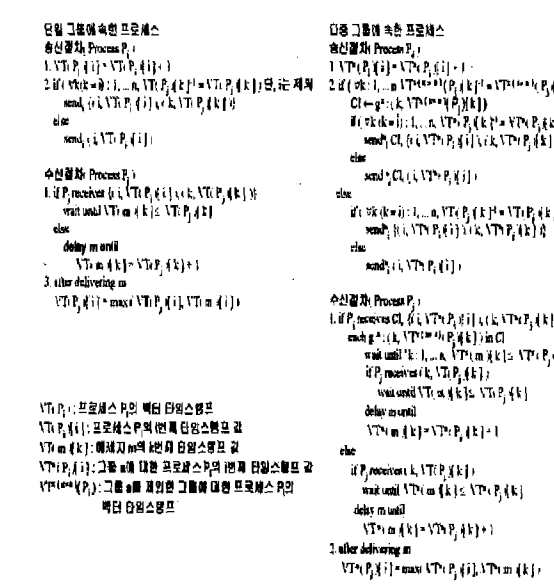
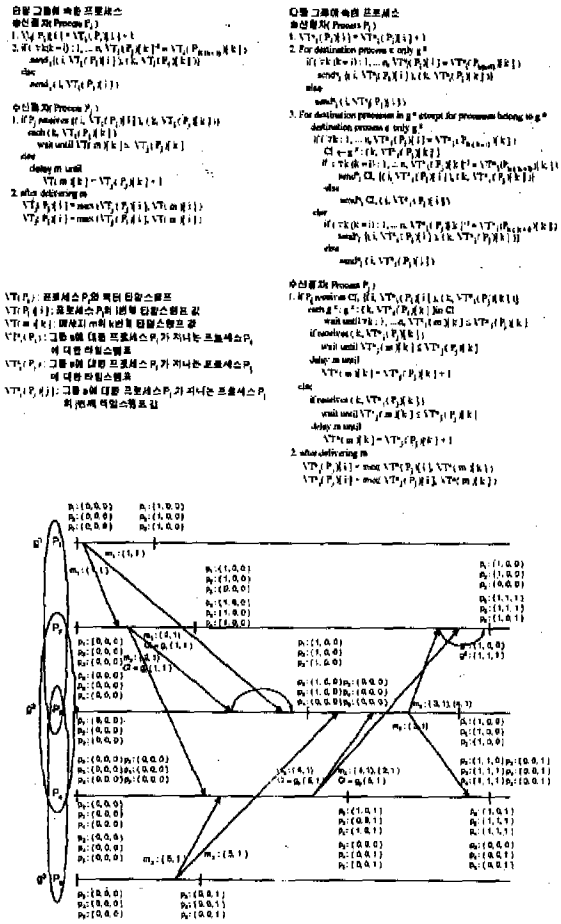


그림 3을 살펴보면, P_1은 m_1을 전송할 시, 이전의 타임스탬프와 비교하여 변한 field만 전송하므로 (1, 1)을 piggybacking하여 전송한다. 여기서 P_2는 g^2에게 전송할 시 level 1과 달리 프로세스 P_2는 그룹 g^2에게 메세지를 전송할 시, 그룹 g^2에 대한 타임스탬프를 이전의 타임스탬프와 비교하여 변한 field (2, 1) 뿐만 아니라, g^1에 대한 타임스탬프도 이전과 비교하여 변한 field, (1, 1)을 CI에 저장한 후 piggybacking하여 전송한다. 즉, 레벨 1 수신자 선택 알고리즘과 달리 현재 송신하고자하는 그룹의 타임스탬프뿐만아니라 CI의 정보도 압축하여 더욱 메세지 오버헤드를 줄였다. 그러므로 프로세스 P_3는 CI의 정보를 이용하여 m_2를 먼저 수신하더라고 m_1을 전달한 후 m_2를 전달하게 된다. 마찬가지로 프로세스 P_5도 m_3를 전송하며, 프로세스 P_4는 g^2에게 메세지를 전송할때 이전의 타임스탬프와 비교하여 변한 field (4, 1), (2, 1)와 함께 g^3에 대한 (5, 1)을 CI에 저장한 후 piggybacking하여 전송한다. 마찬가지로 P_5가 메세지를 g^2에게 전송하면 CI의 정보를 이용하여 P_2는 m_4, m_5의 순서로 메세지를 응용 프로세스에게 전달한다.

2.2. 송신자 선택 알고리즘

다음은 송신자 선택 알고리즘을 기술하였으며, 그림 4는 제안된 알고리즘의 예를 보여준다.

그림 4를 살펴보면, P_1은 m_1을 전송할 시 $VT_1(P_1)[i]$을 갱신한 후, $VT_1(P_1)$와 $VT_1(P_{k(k\neq1)})$를 비교한다. 그러므로 (1, 1)을 전송하게 된다. 여기서 P_2는 g^2에게 전송할 시 레벨 1 송신자 선택 알고리즘과 달리 그룹 g^2에 대한 타임스탬프를 갱신한 후 $VT^2_2(P_1)$와 $VT^2_2(P_{k(k\neq2)})$를 비교할뿐만 아니라, g^1에 대한 $VT^1_2(P_2)$와 $VT^1_2(P_{k(k\neq2)})$를 비교한다. 그러므로 (2, 1)과 함께 g^2 : (1, 1)을 CI에 저장한 후 같이 piggybacking하여 전송하게 된다. 즉, 레벨 1 송신자 선택 알고리즘과 달리 다른 프로세스의 정보를 이용하여 전송하므로 메시지 오버헤드를 더욱 줄일 수 있다. 그러므로 프로세스 P_3는 CI의 정보를 이용하여 m_2를 먼저 수신하더라고 m_1을 전달한 후 m_2를 전달하게 된다. 마찬가지로 다른 프로세스들도 단일 그룹에 속한 프로세스에게는 해당 그룹에 속한 프로세스의 정보를 이요하여 타임스탬프를 압축하여 전송한다. 또한 중첩된 그룹에 속한 프로세스에게는 전송하고자 하는 그룹의 정보뿐만아니라 다른 그룹에 대한 타임스탬프도 그 그룹에 속한 프로세스의 타임스탬프 정보를 이용하여 전송한다. 그러므로 각 메시지는 인과관계 순서화에 어긋남이 없이 응용 프로세스에게 전달하게 된다.



Ⅳ. 인과관계 순서화 증명

본 장에서는 제안된 알고리즘이 중첩된 그룹 환경에서 인과관계 순서화를 만족하는 가에 대해 두 단계를 이용하여 증명하고자 한다. 즉, 안정성(Safety)으로서 인과관계 순서화가 만족되는 가를 확인하고 활성(Liveness)으로서 멀티캐스트 메시지는 무한히 지연되지 않는다는 것을 보이고자 한다. 통신환경은 무손실(Lossless), 활성(Liveness) 상태로 가정하였다.

1. 안정성(Safety)

Theorem 1. 송신자 선택 및 수신자 선택 알고리즘은 정의 1과 정의 2의 인과관계 순서화를 만족하며 메시지를 전달한다.

i) 수신자 선택 및 송신자 선택 알고리즘(레벨 1)

증명을 위해 프로세스 P_j는 $m_1 \rightarrow m_2$ 관계인 두 개의 메시지 m_1, m_2를 수신하였다고 가정하고, 그룹 g^a의 P_i가 전송한 메시지 m_1을 P_j가 응용 프로세스에게 전달한 후, 그룹 g^b에 m_2를 전송할 시 g^a와 g^b에 속한 프로세스 P_k가 수신한 경우에 대해 증명한다.

프로세스 P_k에서 m_2가 m_1,보다 먼저 전달될 수가 없음을 증명하면 다음과 같다. 먼저 m_1은 전달되지 않았고 P_k는 k개의 메시지를 수신하였다고 가정한다.

기본 단계 : P_k에 의해 전달되는 첫 번째 메시지는 m_2가 될 수 없다. 만약 P_k에게 어떤 메시지도 전달되지 않았으면 $VT^a(P_k)[i] = 0$이다. 그러나, $VT^a(m_1)[i] > 0$이므로 $CI(m_2) > 0$이다. 알고리즘의 송신 절차에 의해 $VT(m_1) \in CI(m_2)$이므로 m_2는 P_k에게 첫 번째로 전달될 수 없다.

유도 단계 : 프로세스 P_k는 k개의 메시지를 수신하였고, 그중 어떤 것도 $send(m_1) \rightarrow send(m_2)$인 메시지가 아니라고 가정한다. 만약 m_1이 아직 전달되지 않았다면, $VT^a(P_k)[i] < VT^a(m_1)[i]$이다. $VT^a(P_k)[i]$가 $VT^a(m_1)[i]$보다 큰 값을 가질 수 있는 유일한 방법은 P_i로부터 m_1에 결과적인 메시지를 전달하는 것이고, 그러한 메시지는 m_1에 인과관계를 가진다. 그러므로 $VT(m_1) \in CI(m_2)$과 $VT^a(P_k)[i] < VT^a(m_1)[i]$에 의해 프로세스 P_k에 k + 1번째 전달되는 메시지는 m_2가 될 수 없다.

ii) 수신자 선택 알고리즘(레벨 2)

압축된 타임스탬프를 이용하여 전체 정보를 복원할 수 있으므로 증명은 레벨 1 송신자 선택 및 수신자 선택 알고리즘의 경우와 같다.

iii) 송신자 선택 알고리즘(레벨 2)

프로세스 P_k에서 m_2가 m_1보다 먼저 전달될 수가 없음을 증명하면 다음과 같다. 먼저 m_1은 전달되지 않았고 프로세스 P_k는 k개의 메세지를 수신하였다고 가정한다. $send(m_1) \rightarrow send(m_2)$이므로 $CVT(m_2)$에 $CVT(m_1)$을 포함하게 된다. 특별히, m_1의 송신자인 프로세스 P_i에 대한 field만 고려한다면, (i, sequence number(m_1)) $\leq$ (i, sequece number(m_2))이다.

기본 단계 : P_j가 응용 프로세스에게 전달한 첫 번째 메세지는 m_2가 될 수 없다. 만약 P_j에게 어떤 메세지도 전달되지 않았다면, $VT^a_k(P_k)[i] = 0$이다. 그러나 (i, sequence number(m_1))에서 sequence number(m_1) > 0(왜냐하면 m_1이 P_i에 의해 전송되었으므로)이므로 알고리즘의 수신 절차에 의해 m_2는 P_k에게 첫 번째로 전달될 수 없다.

유도 단계 : 프로세스 P_j는 k개의 메세지를 수신하였고, 그중 어떤 것도 $send(m_1) \rightarrow send(m)$인 메세지 m이 아니라고 가정한다. 만약, m_1이 아직 전달되지 않았다면, $VT^a_k(P_k)[i] <$ sequence number(m_1)이다. $VT^a_k(P_k)[i]$가 sequence number(m_1)보다 큰 값을 가질 수 있는 유일한 방법은 P_i로부터 m_1에 subsequent한 메세지를 전달하는 것이고, 그러한 메세지는 m_1에 인과관계를 가진다. 그러므로 식 (i, sequence number(m_1)) $\leq$ (i, sequece number(m_2))과 $VT^a_k(P_k)[i] <$ sequence number(m_1)에 따라 $VT^a_k(P_k)[i] <$ sequece number(m_2)가 유도된다. 그러므로 수신 알고리즘에 따라 프로세스 P_j에 k+1번째 전달되는 메세지는 m_2가 될 수 없다.

2. 활성(Liveness)

Theorem 2. 송신자 선택 및 수신자 선택 알고리즘은 결국 모든 메시지를 전달한다.

증명 : 프로세스 P_k에 결코 전달될 수 없는 메시지 m이 존재한다고 가정한다.

i) 수신자 선택 및 송신자 선택 알고리즘(레벨 1)

<u>단일 그룹에 속할 경우</u>

증명은 이미 기술되어 있다[23],[24].

<u>다중 그룹에 속할 경우</u>

가정에 의해 다음의 둘중 하나가 성립한다.

경우 1. $VT^b(m)[j] \neq VT^b(P_k)[j] + 1$, 혹은 $\exists k \neq j : VT^b(m)[k] > VT^b(P_k)[k]$

증명은 단일 그룹의 경우와 동일하다.

경우 2. CI(m) = VT(m')일 때, VT(m') ∋ CI(m) 인 경우

중첩된 프로세스에게 메시지 m이 도착하지 않았거나, 인과관계 때문에 전달이 지연이 된 m' → m 인 메시지 m'가 반드시 존재한다. 모든 메시지는 무손실 통신 환경에서 중첩된 프로세스에게 도착된다는 가정아래, m'는 중첩된 프로세스에게 도착되고 m'' → m'인 메시지 m''도 도착된다. 그러므로 m도 도착하게 되며, 이때 메시지 m 이전에 도착해야 할 메시지는 유한개이며 → 는 비순환이므로 제안된 알고리즘은 CI에 의해 모두 인과관계 순서화를 만족하며 전달된다. 이는 가정에 위배되므로 다중 그룹에 속한 프로세스에게 모든 메시지는 전달된다.

ii) 수신자 선택 및 송신자 선택 알고리즘(레벨 2)

단일그룹에 속할 경우

- (i, $VT_i(m)[i]$)에서 $VT_i(m)[i] \neq VT_j(P_j)[i] + 1$ 또는
- ($\exists k \neq i$, $VT_k(m)[k]$)에서 $VT_k(m)[k] > VT_j(P_j)[k]$

경우 1. $VT_i(m)[i] \neq VT_j(P_j)[i] + 1$

m은 프로세스 P_i로부터 전송된 다음 메세지가 아닌 경우이다. 모든 메세지는 모든 프로세스에게 멀티캐스트되며, channel은 무손실이므로, 프로세스 P_i로부터 P_j에게 전송된 메세지(P_j가 아직 전달하지 않은) m'가 반드시 존재한다. 그리고 $VT_i(m)[i] \neq VT_j(P_j)[i] + 1$이다. 만약 m'가 지연되었다면, 경우 2이다.

경우 2. $\exists k \neq i : VT_k(m)[k] > VT_j(P_j)[k]$

$VT_k(m)[k]$에서 $n = VT_k(m)[k]$이라고 하자. 프로세스 P_k의 n번째 전송에서 프로세스 P_j가 아직 수신하지 못했거나, 혹은 수신했어도 지연된 어떤 메세지 m'->m 가 반드시 존재한다. 모든 메세지는 모든 프로세스에게 송신된다는 가정아래, m'는 이 미 P_j에게 멀티캐스트되었다. 그러므로 m'는 통신 채널에 존재한다. 통신 시스템은 결국 모든 메시지 를 전달하므로 m'는 P_j에게 수신되었다고 할 수 있다. 또한, m'에 적용한 같은 이유를 m에게도 적 용가능하다. m이전에 전달되어야 할 메세지의 수 는 유한적이고, -> 는 비순환이므로 이는 가정에 위배된다.

다중 그룹에 속할 경우

CI에 해당하는 그룹에 대한 타임스탬프와 해당 그룹에 대한 타임스탬프는 서로 독립적인 관계이다. 그러므로 CI 역시 단일 그룹에 대한 타임스탬프이므로 CI에 해당하는 메시지에 대한 증명도 단일 그룹에 대한 증명과 같다.

V. 성능평가

본 장에서는 제안된 인과관계 순서화 알고리즘에 대해 중첩된 그룹 환경에서 각 단계별 알고리즘을 기존의 확장된 CBCAST 알고리즘과 비교하였다. 모의 실험 환경은 acknowledgement, 네트워크 구성, 폭주, 라우팅 등은 고려하지 않았으며, 시뮬레이션은 Solaris 2.5의 운영 체계를 가지는 SUN Sparc 20에서 수행하였다. 프로세스간 통신을 위한 IPC (InterProcess Communication)로는 메세지 큐를 사용하였으며, 각 프로세스의 메세지는 랜덤하게 발생시켜 실험하였다. 모의 실험은 브로드캐스트 네트워크 환경과 점대점 네트워크 환경의 두 경우에서 실험하였다. 즉, 브로드캐스트 네트워크 환경에서는 기존의 확장된 CBCAST 알고리즘과 레벨 1의 수신자 선택 알고리즘과 레벨 2 수신자 선택 알고리즘을 비교하였으며, 점대점 네트워크 환경에서는 기존의 알고리즘과 레벨 1 송신자 선택 알고리즘과 레벨 2 송신자 선택 알고리즘을 메시지의 인과관계 순서화에 필요한 메세지 오버헤드 측면에서 비교하였다.

중첩된 프로세스 환경에서의 모의 실험은 각 프로세스들이 하나의 그룹에서 멀티캐스트를 수행할 시 각 프로세스들은 다른 지연 타이머(delay timer)를 이용하여 메세지를 생성시켰다. 모의 실험에서는 고정된 총 프로세스의 수, 그룹의 크기, 그룹의 수에 대해 각 요소를 변화시키면서 인과관계 순서화에 필요한 메세지 오버헤드를 측정하였다. 메세지 오버헤드란, 인과관계 순서화를 위해 필요한 각 타임스탬프 field의 개수이다.

그림 5와 그림 6은 점대점 네트워크 환경에서의 실험 결과이다. 실험은 고정된 범위의 총 프로세스의 수에서 고정된 개수의 그룹 크기와 그룹의 수만큼 랜덤하게 선택하도록 하였다. 그림 5는 그룹의 크기를 5개의 프로세스로, 그룹의 수를 10개의 그룹으로 고정시킨 후 총 프로세스의 수를 변화시키면서 생성되는 그룹 환경에서의 실험 결과이다. 실험 결과, 프로세스의 수가 적을 수록 중첩된 그룹이 많이 생성됨을 확인하였다. 또한, 중첩 그룹의 수가 많을 수로 제안된 인과관계 순서화 알고리즘의 성능은 레벨 1 수신자 선택 알고리즘의 경우 최고 28%의 메세지 오버헤드의 단축을 보였으며, 레벨 2

의 경우에는 최고 37%의 압축 효과를 보였다. 반면에 프로세스의 수가 많아질수록 압축 효과는 적어졌으며 레벨 1의 경우 최고 15%, 레벨 2의 경우는 20%였다. 그림 6에서는 총 프로세스의 수를 200개로, 그룹의 수를 10으로 고정시킨 후, 그룹의 크기(그룹내 프로세스의 수)를 변화시키면서 생성되는 중첩된 그룹 환경에서의 실험 결과이다. 실험 결과, 그룹내 프로세스의 수가 많아질수록 중첩 그룹이 많이 생성됨을 확인하였다. 또한, 그룹의 크기가 클수록 제안된 레벨 1 수신자 선택 알고리즘의 경우 최고 25%의 메세지 오버헤드의 단축을 보였으며, 레벨 2의 경우에는 최고 36%의 압축 효과를 보였다. 반면에 프로세스의 수가 많아질수록 압축 효과는 적어졌으며 레벨 1의 경우 최고 15%, 레벨 2의 경우는 18%였다.

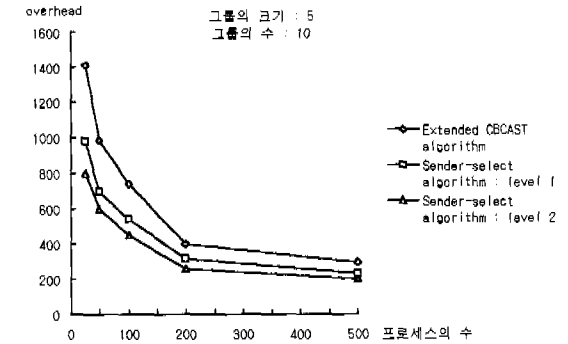


그림 5. 프로세스 수의 변화에 따른 각 알고리즘의 오버헤드

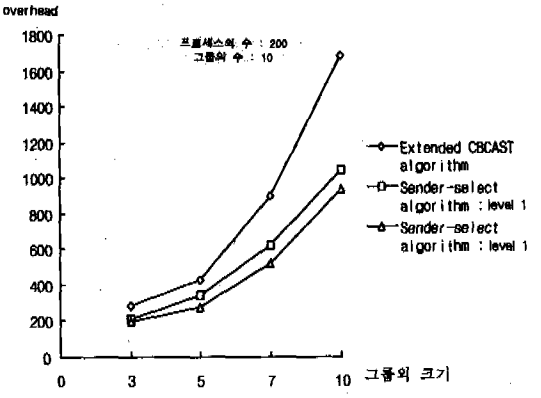


그림 6. 그룹 크기의 변화에 따른 각 알고리즘의 오버헤드

즉, 제안된 수신자 선택 알고리즘의 경우 그룹의 크기와 그룹의 수가 결정되었을 때, 총 프로세스의 수가 적을수록 유리함을 볼 수 있으며, 총 프로세스의 수와 그룹의 수가 결정되었을 때, 그룹의 크기가 커질수록 기존의 알고리즘보다 유리함을 알 수 있다.

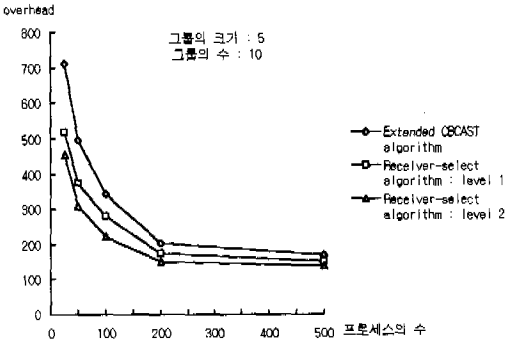


그림 7. 프로세스 수의 변화에 따른 각 알고리즘의 오버헤드

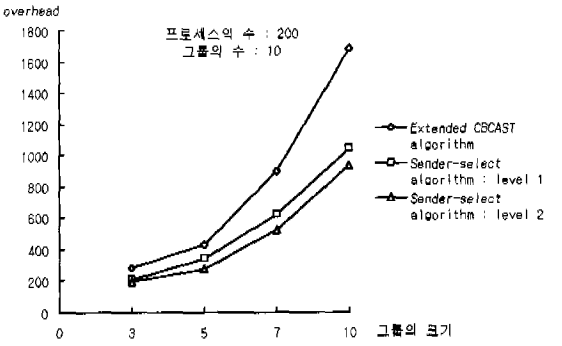


그림 8. 그룹 크기의 변화에 따른 각 알고리즘의 오버헤드

그림 7은 그림 5와 같은 조건에서 생성된 중첩 그룹 환경에서 제안된 송신자 선택 알고리즘을 기존의 확장된 CBCAST 알고리즘과 비교하였다. 중첩 그룹의 수가 많을 수록 제안된 송신자 선택 알고리즘의 경우 최고 34%의 메세지 오버헤드의 단축을 보였으며, 레벨 2의 경우에는 최고 43%의 압축 효과를 보였다. 반면에 프로세스의 수가 많아질수록 압축 효과는 적어졌으며 레벨 1의 경우 최고 20%, 레벨 2의 경우는 27%였다. 그림 8은 그림 6과 같은 조건에서 생성된 중첩 그룹 환경에서 제안된 송신자 선택 알고리즘을 기존의 확장된 CBCAST 알고리즘과 비교하였다. 또한, 그룹의 크기가 클수록 제안된 레벨 1 수신자 선택 알고리즘의 경우 최고 33%의 메세지 오버헤드의 단축을 보였으며, 레벨 2의 경우에는 최고 41%의 압축 효과를 보였다. 반면에 프로세스의 수가 많아질수록 압축 효과는 적어졌으며 레벨 1의 경우 최고 20%, 레벨 2의 경우는 27%였다.

즉, 제안된 송신자 선택 알고리즘과 마찬가지로, 그룹의 크기와 그룹의 수가 결정되었을 때; 총 프로

세스의 수가 적을수록 유리함을 볼 수 있으며, 총 프로세스의 수와 그룹의 수가 결정되었을 때, 그룹의 크기가 커질수록 기존의 알고리즘보다 유리함을 알 수 있다.

Ⅵ. 결 론

본 논문에서는 임의의 프로세스가 여러 그룹에 속하는 중첩된 프로세스 그룹 환경에 효과적으로 적용할 수 있는 인과관계 순서화 알고리즘을 위한 효율적인 벡터 타임스탬프 기법을 제시하였다. 기존의 알고리즘을 중첩된 프로세스 환경에 적용하면, 단일 그룹에 속한 프로세스는 비효율적으로 타임스탬프를 이용하게 된다. 이에 반해 제안된 알고리즘은 기존의 단점을 제거하였을뿐만 아니라 더 나아가 메시지의 인과관계 순서화에 필요한 벡터 타임스탬프의 압축을 수행하였다. 또한, 제안된 알고리즘은 네트워크 구성 형태에 따라 선택할 수 있도록 제안하였다. 즉, 브로드캐스트 네크워크에서는 수신자 선택 알고리즘을 제안하였으며, 점대점 네트워크에서는 송신자 선택 알고리즘을 제안하였다. 각 알고리즘은 순서화에 요구되는 메시지 오버헤드를 줄이기 위해 그룹별로 불필요한 벡터 타임스탬프를 제거하였으며, 메시지 오버헤드를 최소화하기 위해 국부적으로 유지하고 있는 다른 프로세스와 다른 그룹의 정보를 이용하여 벡터 타임스탬프를 압축하였다. 또한, 각각의 새로운 인과관계 순서화 알고리즘을 논리적으로 증명하였으며, 시뮬레이션을 통해 기존의 인과관계 순서화 알고리즘과의 성능을 비교하였다.

향후 연구 과제로는 제안된 인과관계 순서화 알고리즘을 동적인 그룹 환경으로 확장하여 적용하는 것과 통신망상에서 실질적으로 운용하는 연구를 수행하는 것이다.

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] J. Chang and N. Maxemchuck, "Reliable Broadcast Protocols", ACM Transaction on Computer System, Vol. 2, No. 2, pp 251-273, Aug. 1984.

[3] P. Stephenson, Fast Ordered Multicast, Ph.D. thesis, Dept. of Computer Science, Cornell University, 1991.

[4] A. Acharya and B.R. Badrinath, Recording Distributed Snapshots based on Causal Order of Message Delivery, Inform. Process. Lett. pp 317-321, 1992

[5] Hector Garcia-Molina and Annemaria Spauster, "Ordered and Reliable Multicast Communication, ACM Transaction on Computer System, Vol. 9, No. 3, pp 242-271, Aug. 1991.

[6] 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.

[7] 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.

[8] 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.

[9] Bala Rajagopalan, "Reliablility and Scaling Issues in Multicast Communicaton", ACM SIGCOMM'92, pp 188-198, Aug. 1992

[10] Erwin Mayer, "An Evaluation Framework for Multicast Ordering Protocols", ACM SIGCOMM '92, pp 177-187, Aug. 1992

[11] 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.

[12] M. Frans Kaashoek and Andrew S. Tanenbaum, "Fault Tolerance Using Group Communication", Operating Systems Review, Vol. 30, pp 71-74, Aug. 1989.

[13] 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.

[14] 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.

[15] 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.

[16] Adrian Segall, and Baruch Awerbuch, "A Reliable Broadcast Protocol" IEEE Transactions on Communications, Vol. Com-31, pp 896-901, No. 7, July 1983.

[17] Rosario Aiello, Elena Pagani, and Gian Paolo Rossi "Causal Ordering in Reliable Group Communications", ACM SIGCOMM'93, pp 106-115, Sept. 1993.

[18] J. Crowcroft and K. Paliwoda, "A Multicast Transport Protocol", ACM SIGCOMM, Vol. 18. No. 4, pp 247-256, Aug. 1988.

[19] Alan O. Freier and Keith Marzullo, "MTP : An Atomic Multicast Transport Protocol", Technical Report, Cornell University Computer Science Department, July 1990.

[20] 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.

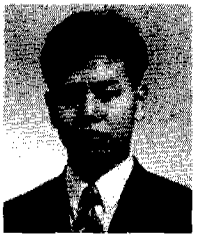
[21] Nasr E. Belkeir and Mustaque Ahamad, "Low Cost Algorithms for Message Delivery in Dynamic Multicast Groups", Proceedings of the 9th ICDCS, pp 110-117, 1989.

[22] S. Deering, "Host Extension for IP Multicasting" RFC 1112, Standford University, May 1989.

[23] 권 봉경, 정 광수, 현 동환, 함 진호, "중첩된 프로세스 그룹 환경에서의 멀티캐스트 알고리즘", 한국통신학회 논문집, 제 21권 제 4호, 1996.

[24] 권 봉경, 정 광수, "새로운 압축 방식을 이용한 인과관계 순서화 알고리즘", 한국통신학회 논문집, 제 22권 제 6호, 1997.

권 봉 경(Bong-kyoung Kwon) 정회원



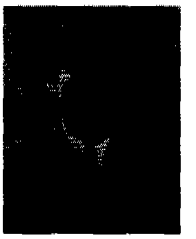
1995년 2월 : 광운대학교 전자통신공학과 학사

1997년 2월 : 광운대학교 전자통신공학과 석사

1997년~현재 : 대우통신 교환연구단 교환연구5실 연구원

<주관심 분야> 멀티미디어통신, 컴퓨터통신, 분산처리 시스템

정 광 수(Kwang-sue Chung) 정회원



1981년 2월 : 한양대학교 전자공학과 학사

1983년 2월 : 한국과학기술원 전기 및 전자공학과 석사

1991년 1월 : 미국 University of Florida 전기공학과 박사 (컴퓨터공학전공)

1983년~1993년 : 한국전자통신연구원 선임연구원

1991년~1992년 : 한국과학기술원 대우교수

1993년~현재 : 광운대학교 전자공학부 부교수(신기술연구소 연구원)

<주관심 분야> 멀티미디어통신, 컴퓨터통신, 분산처리 시스템