論文 97-22-6-1

새로운 압축 방식을 이용한 인과관계 순서화 알고리즘

正會員 권 봉 경*, 정 광 수**

A Causal Ordering Algorithm Using a New Compression Method

Bong-Kyoung Kwon*, Kwang-Sue Chung** *Regular Members*

※본 연구는 광운대학교 교내연구비 지원에 의해 수행되었음.

요 약

그룹 통신에서 메세지 순서화를 만족시키기 위해 벡터 타임스탬프를 사용한다. 본 논문에서는 하나의 프로세스가 단일 그룹에 속하는 환경에서 이용되는 벡터 타임스탬프의 효율적인 압축 방식을 제안하였다. 기존의 압축 방식은 이전에 전송한 메세지의 벡터 타임스탬프와 현재의 벡터 타임스탬프를 비교하여 변화된 부분만 전송하는 방식이다. 본 논문에서는 기존의 압축 방식과 달리 국부적으로 유지하고 있는 다른 프로세스의 벡터 타임스탬프 정보를 이용하여 개별적인 압축을 수행하는 방식을 제안하였다. 또한 새로운 압축 방식을 이용한 인과관계 순서화 알고리즘을 논리적으로 증명하였고, 시뮬레이션을 통해 기존의 압축 알고리즘과의 성능을 비교하였다. 제안된 압축 방식을 이용하면 기존의 압축 방식에 비해 인과관계 순서화에 요구되는 메세지 오버헤드를 줄일 수 있다.

ABSTRACT

A vector timestamp is used to satisfy message ordering in a group communications. In this paper, we propose a new vector timestamp compression method which is applicable to a single process group environment where one process belongs to only one precess group. An existing compression method compares the fields of the previously sent vector timestamp with those of the currently updated vector timestamp, then sends only the modified fields of the vector timestamp. Unlike the previous one, a proposed compression method performs individual compression for each process using the locally maintained vector timestamp information on other processes. Also, we logically proved the causal ordering algorithm using the new compression method and compared the performance of the proposed method with one of the previous compression method by computer simulation. Using the proposed compression method, the message overhead required for causal ordering can be reduced.

*대우통신 교환연구단 교환연구5실
**광운대학교 전자공학부
論文番號:96324-1011
接受日字:1996年 10月 11日

Ⅰ. 서 론

대역폭이 증대되고 통신망이 고속화됨에 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 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. 멀티캐스트를 이용하는 응용에서는 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 메세지 송수신에 관한 연구[3][5][7], 프로세스 그룹의 동적인 변화에 관한 연구[3][21] 등이 있다. 이와 같은 프로세스 그룹의 통신에서는 메세지 순서화에 관한 연구[1][3][5][7][15][17][20] 및 메세지의 오버헤드를 줄이는 연구[3][7]가 매우 중요하다.

이외에도 멀티캐스트에 관한 연구로는 멀티캐스트 도중에 발생하는 오류에 대처하는 연구[11][12], 송수신하는 메세지의 효율성 문제[2][8][9], 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 공동 에디팅, 화상회의와 같이 프로세스들이 하나의 그룹에 속하는 단일 프로세스 그룹 환경에서 프로세스간에 송수신되는 메세지의 인과관계 순서화를 위한 벡터 타임스탬프에 대해 기술한다. 또한 효율적인 멀티캐스트를 위해 기존의 벡터 타임스탬프의 압축 방식을 기술하며, 메세지 오버헤드 측면에서 기존 방식의 단점을 극복하기 위한 새로운 압축 방식을 제안한다. 본 논문의 구성은 제2장에서 인과관계 순서화를 기술하기 위한 프로세스 그룹 및 통신 모델을 설명하였고, 제3장에서는 하나의 프로세스가 단일 그룹에 속하는 환경에서의 벡터 타임스탬프 및 기존의 압축 기법에 관해 기술하였다. 제4장에서는 기존의 압축 방식의 단점을 극복한 새로운 압축 방식을 제안하였으며, 제안된 알고리즘의 논리적인 증명을 하였다. 제5장에서는 단일 프로세스 그룹 환경에서 기존의 압축 방식과 제안된 압축 방식을 컴퓨터 시뮬레이션을 통해 비교하였으며, 제6장에서 결론을 맺는다.

Ⅱ. 인과관계 순서화 기술

2.1 프로세스 그룹 및 통신 모델

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 그룹에만 속한다고 가정한다. 멀티캐스트 통신에서 메세지의 send, receive, deliver는 프로세스의 이벤트라 가정한다.

프로세스 그룹 g에 속한 프로세스 P_i에 의한 메세지 m의 멀티캐스트는 $send_i(m)$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.

2.2 인과관계 순서화

인과관계 순서화는 "happened before" 관계이며 → 로 나타낸다[1].

정의 : 만약 다음의 조건중 하나라도 성립되면, 두개의 이벤트 a, b의 관계는 a→b이다.

- 프로세스 P_i에서, $a = send_i(m)$가 $b = send_i(m)$보다 먼저 발생한 경우
- 프로세스 P_i에서 $a = send_i(m)$이며, b는 프로세스 P_j에서의 $receiv_j(m)$인 경우
- 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, 두 개의 이벤트 a, b가 a→b, b→a일 경우 두 이벤트는 인과관계가 없는 동시적인 메세지이다.

Ⅲ. 기존의 알고리즘 및 압축 방식

3.1 VT(Vector Timestamp) 알고리즘[3][7]

Lamport 타임스탬프[1]와 비교하여 벡터 타임스탬프는 → 관계를 정확하게 나타낼 수 있는 장점이 있다. 프로세스 P_i에 대한 벡터 타임스탬프는 n 차원(dimension) 벡터인 $VT(P_i)$로 표시되며(n은 그룹내 프로세스의 수), 프로세스가 전송한 메세지 m에 piggybacking 되어 있는 벡터 타임스탬프는 VT(m)로 표시한다. 각각의 차원은 각 프로세스가 전송한 메세지의 순차번호를 나타낸다. 구체적인 알고리즘은 다음과 같으며, 그림 1에서 예를 보여준다.

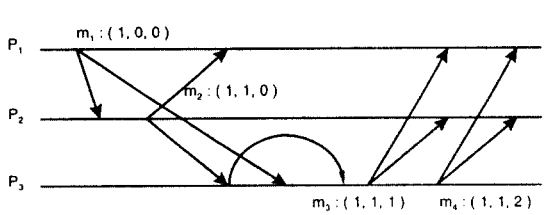

그림 1. VT 알고리즘의 예
Fig. 1 Example of the VT algorithm

(1) 프로세스 P_i는 메세지 m을 그룹 멤버에게 멀티캐스트하기 전에, $VT(P_i)[i]$을 1만큼 증가시킨 후 멀티캐스트를 수행한다.

(2) P_i가 송신한 VT(m)을 포함하는 메세지를 프로세스 P_j가 수신하면, P_j는 다음의 조건이 만족될 때까지 전달을 지연시킨다.

$$\forall k : 1 \ldots n \left\{ \begin{array}{ll} VT(m)[k] = VT(p_j)[k] + 1 & \text{if } k = i \\ VT(m)[k] \le VT(p_j)[k] + 1 & \text{otherwise} \end{array} \right\}$$

(3) 프로세스 P_j는 메세지 m이 응용 프로세스에게 전달되면, $VT(P_j)[i]$을 다음과 같이 갱신한다.

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

VT 알고리즘을 이용한 멀티캐스트 방식은 단일 프로세스 그룹내에 많은 프로세스가 존재하는 경우, 벡터 타임스탬프로 인한 메세지 오버헤드의 크기가 매우 커지게 되는 단점이 있다. 이와 같은 단점을 극복하기 위하여 벡터 타임스탬프 압축 기법이 사용되고 있다.

3.2 VT 압축 기법[3, 7]

n 차원의 벡터 타임스탬프를 모두 송신하는 VT 알고리즘과 달리 VT 압축 알고리즘은 마지막으로 전송한 멀티캐스트 메세지의 벡터 타임스탬프와 VT 알고리즘의 (1)을 수행한 벡터 타임스탬프를 비교한 후, 변한 차원의 벡터 타임스탬프만을 piggybacking하여 메세지를 멀티캐스트하는 방식이다. 벡터 타임스탬프에서 각 프로세스의 순차 번호가 생략되는 이유는 다음의 두 가지 경우이다. 첫째, 프로세스 P_i는 생략된 순차 번호에 해당하는 프로세스에게서 멀티캐스트 메세지를 수신하지 못한 경우이며, 둘째, P_i에 의한 이전 멀티캐스트 메세지에 생략된 프로세스의 순차 번호가 포함되었기 때문이다. 여기서 프로세스 P_i에 의한 현 송신 상태의 타임스탬프를 $VT(P_i)[i]$이라 할 때, 바로 이전에 프로세스 P_i가 송신했던 타임스탬프를 $VT_{-1}(P_i)[i]$로 표시한다. 압축 알고리즘은 다음과 같으며, 구체적인 예는 그림 2에 나타나있다.

(1) 프로세스 P_i는 $VT(P_i)[i]$를 증가시킨 후, $VT_{-1}(P_i)$와

비교하여 변한 부분에 대해서만 piggybacking한다.

(2) P_i는 VT(m)을 포함하는 메세지를 멀티캐스트하며, 프로세스 P_j는 이 메세지를 수신한다. 여기서 생략된 프로세스 식별자는 제외한 후 다음의 조건이 만족될 때까지 전달을 지연시킨다.

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

(3) 메세지 m이 전달되면, VT(P_j)을 갱신하며 생략된 부분은 프로세스 Pj가 이전에 수신한 타임스탬프와 동일하게 갱신한다.

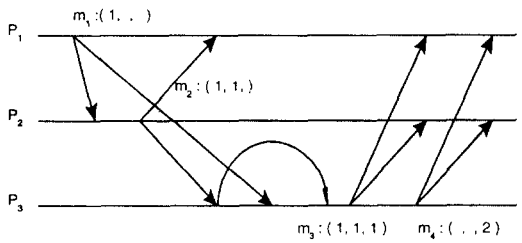

그림 2. VT 압축 기법의 예
Fig. 2 Example of the VT compression method

Ⅳ. 제안된 압축 기법 및 논리적 증명

4.1 제안된 압축 기법

제안된 압축 알고리즘의 기본적인 개념은 한 프로세스가 국부적으로 가지고 있는 다른 프로세스에 대한 벡터 타임스탬프 정보를 이용하여 각각의 프로세스에게 최소의 벡터 타임스탬프를 piggybacking함으로써 메세지 오버헤드의 양을 줄이는데 있다. 이를 위해 프로세스들은 각각의 프로세스에 대한 추가적인 벡터를 유지해야 한다. 프로세스 P_i에 대한 벡터 타임스탬프는 $VT_i(P_i)$로 표시하며, 국부적으로 유지하고 있는 다른 프로세스에 대한 타임스탬프 정보는 $VT_i(P_{k(k\neq i)})$로 표시한다. 추가적인 벡터는 처음 그룹이 생성될 때 그룹 멤버의 수에 따라 결정되어지므로 확장성이 보장된다.

구체적인 알고리즘은 다음과 같다.

송신측 :

(1) $VT_i(P_i)[i] = VT_i(P_i)[i] + 1$

(2) $VT_i(P_i)$와 $VT_i(P_{k(k\neq i)})$를 비교한다.

(3) CVT(Compressed Vector Timestamp)로서 (i, $VT_i(P_i)[i]$)와 함께 $VT_i(P_i)$와 $VT_i(P_{k(k\neq i)})$의 다른 field를 (process id, sequence number)로서 각각의 프로세스에게 piggybacking하여 전송한다.

(4) 각각의 프로세스에 대한 국부적인 VT값을 갱신한다.

수신측 :

(1) 프로세스 P_j에서 $receiv_j(m)$이 발생한 후, CVT에서 (i, $VT_i(P_i)[i]$)를 제외한다.

(2) CVT에서 다른 field가 존재할 경우 해당 메세지 (m')의 수신 이벤트를 기다린다. 다른 field가 없는 경우는 (4)를 수행한다.

(3) $deliv_j(m')$을 수행한 후 해당 메세지에 대한 CVT를 이용하여 관련된 국부 벡터 타임스탬프를 갱신한다.

(4) $deliv_j(m)$을 수행한 후 (i, $VT_i(P_i)[i]$)에 대해 $VT_j(P_j)[i]$와 프로세스 P_j가 지닌 국부적인 프로세스 P_i에 대한 $VT_j(P_i)[i]$타임스탬프를 갱신한다.

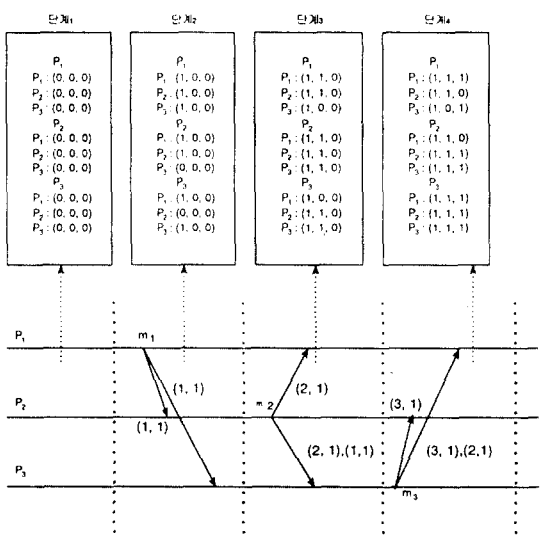

그림 3. 제안된 압축 기법의 예
Fig. 3 Example of the proposed compression method

그림 3은 제안된 압축 알고리즘의 예를 단계별로 나누어 보여주고 있다. 각 단계별로 프로세스의 동작을 설명하면 다음과 같다.

단계 1: 멀티캐스트를 수행하기 전이며, 모든 프로세스는 국부적으로 지니고 있는 벡터 타임스탬프는 0으로 초기화한다.

단계 2: 프로세스 P_1은 m_1을 송신하며 프로세스 P_2, P_3는 m_1을 수신한다.

프로세스 P_1

①송신 알고리즘 (1)에 따라 $VT_1(P_1)$을 (1, 0, 0)으로 갱신한다.

②송신 알고리즘 (2)에 따라 $VT_1(P_1)$과 $VT_1(P_2)$=(0, 0, 0), $VT_1(P_3)$=(0, 0, 0)을 비교한다.

③송신 알고리즘 (3)에 따라 (1, 1)를 메세지 m_1에 piggybacking 한다.

④송신한 타임스탬프 정보에 따라 각각의 프로세스에 대한 타임스탬프를 갱신하므로 $VT_1(P_2)$, $VT_1(P_3)$가 (1, 0, 0)로 갱신된다.

프로세스 P_2

①수신 알고리즘 (1)에 따라 (1, 1)을 제외한다.

②수신 알고리즘 (4)에 따라 $deliv_2(m_1)$을 수행한 후 $VT_2(P_2)$와 $VT_2(P_1)$을 (1, 0, 0)으로 갱신한다. $VT_2(P_3)$에 대해서는 m1의 수신여부를 확인 할 수 없으므로 (0, 0, 0)이 된다.

프로세스 P_3

①수신 알고리즘 (1)에 따라 (1, 1)을 제외한다.

②수신 알고리즘 (4)에 따라 $deliv_3(m_1)$을 수행한 후 $VT_3(P_3)$와 $VT_3(P_1)$을 (1, 0, 0)으로 갱신한다. $VT_3(P_2)$에 대해서는 m_1의 수신여부를 확인 할 수 없으므로 (0, 0, 0)이 된다.

단계 3: 프로세스 P_2는 m_2를 송신하며 P_1, P_3는 m_2를 수신한다.

프로세스 P_2

①송신 알고리즘 (1)에 따라 $VT_2(P_2)$를 (1, 0, 0)을 (1, 1, 0)으로 갱신한다.

②송신 알고리즘 (2)에 따라 $VT_2(P_2)$와 $VT_2(P_1)$=(1, 0, 0), $VT_2(P_3)$=(0, 0, 0)을 비교한다.

③송신 알고리즘 (3)에 따라 프로세스 P_1에게는 (2, 1)을 piggybacking 하고, 프로세스 P_3에게는 (2, 1)(1, 1)을 piggybacking한다.

④송신한 타임스탬프 정보에 따라 각각의 프로세스에 대한 타임스탬프를 갱신하므로 $VT_2(P_1)$, $VT_2(P_3)$가 (1, 1, 0)으로 갱신된다.

프로세스 P_1

①수신 알고리즘 (1)에 따라 (2, 1)을 제외한다.

②수신 알고리즘 (4)에 따라 $VT_1(P_1)$과 $VT_1(P_2)$를 (1, 0, 0)에서 (1, 1, 0)로 갱신한다. $VT_1(P_3)$에 대해서는 m_2의 수신 여부를 확인할 수 없으므로 (1, 0, 0)이 변함이 없다.

프로세스 P_3

①수신 알고리즘 (1)에 따라 (2, 1)을 제외한다.

②수신 알고리즘 (2)에 따라 (1, 1)에 대한 $deliv_3(m_1)$을 확인한다.

③수신 알고리즘 (3)에 따라 m_1은 이미 전달했으므로 $VT_3(P_1)$은 (1, 0, 0)으로 변함이 없다. 프로세스 P_2의 m_1 수신을 확인하였으므로 $VT_3(P_1)$는 (0, 0, 0)에서 (1, 0, 0)으로 갱신된다.

④수신 알고리즘 (4)에 따라 $deliv_3(m_2)$을 수행한 후 $VT_3(P_3)$와 $VT_3(P_2)$를 (1, 0, 0)에서 (1, 1, 0)으로 갱신한다. $VT_3(P_1)$에 대해서는 m_2의 수신 여 부를 확인할 수 없으므로 (1, 0, 0)이 변함이 없다.

단계 4: 프로세스 P_3은 m_3를 송신하며 P_1, P_2는 m_3를 수신한다.

프로세스 P_3

①송신 알고리즘 (1)에 따라 $VT_3(P_3)$를 (1, 1, 0)을 (1, 1, 1)으로 갱신한다.

②송신 알고리즘 (2)에 따라 $VT_3(P_3)$와 $VT_3(P_1)$=(1, 0, 0), $VT_3(P_2)$=(1, 1, 0)을 비교한다.

③송신 알고리즘 (3)에 따라 프로세스 P_2에게는 (3, 1)을 piggybacking하고, 프로세스 P_1에게는 (3, 1)(2, 1)을 piggybacking한다.

④송신한 타임스탬프 정보에 따라 각각의 프로세스에 대한 타임스탬프를 갱신하므로 $VT_3(P_1)$, $VT_3(P_2)$가 (1, 1, 1)으로 갱신된다.

프로세스 P_2

①수신 알고리즘 (1)에 따라 (3, 1)을 제외한다.

②수신 알고리즘 (4)에 따라 $VT_2(P_2)$과 $VT_2(P_3)$를 (1, 1, 0)에서 (1, 1, 1)으로 갱신한다. $VT_2(P_1)$에 대해서는 m_3의 수신 여부를 확인할 수 없으므로 (1, 1, 0)이 변함이 없다.

프로세스 P_1

①수신 알고리즘 (1)에 따라 (3, 1)을 제외한다.

②수신 알고리즘 (2)에 따라 (2, 1)에 대한 $deliv_3(m_2)$을 확인한다.

③수신 알고리즘 (3)에 따라 m_2는 이미 전달되었으므로 $VT_1(P_1)$은 (1, 1, 0)이 변함이 없다. 프로세스 P_3의 m_2 수신을 확인하였으므로 $VT_1(P_3)$는 (1, 0, 0)에서 (1, 1, 0)으로 갱신된다.

④수신 알고리즘 (4)에 따라 $deliv_1(m_3)$을 수행한 후 $VT_1(P_1)$과 $VT_1(P_3)$를 (1, 1, 0)에서 (1, 1, 1)으로 갱신한다. $VT_1(P_2)$에 대해서는 m_3의 수신 여부를 확인할 수 없으므로 (1, 1, 0)이 변함이 없다.

4.2 제안된 알고리즘의 논리적인 증명

본 절에서는 제안된 인과관계 순서화 알고리즘이 메세지간의 인과관계 순서화를 유지하며 전달되는 가에 대해 두 단계를 이용하여 증명하고자 한다. 즉, 안전성(safety)으로서 인과관계 순서화가 만족되는가를 확인하고, 활성(liveness)으로서 멀티캐스트 메세지는 무한히 지연되지 않는다는 것을 보이고자 한다. 증명을 위하여 통신 환경은 무손실(lossless), 활성(live) 상태로 가정하였다.

⊙ 안전성(safety)

[Theorem 1] 제안된 압축 방식을 이용한 알고리즘은 항상 인과관계 순서화를 유지하며 메세지를 전달한다.

증명: 프로세스 P_j가 $send(m_1) \rightarrow send(m_2)$인 메세지 m_1과 m_2를 수신하였다고 가정하면, 다음의 두 가지 경우중의 하나가 발생한다. 그러므로 각각의 경우에 대해 논리적으로 증명을 하면 인과관계 순서화가 만족된다.

경우 1. m_1과 m_2가 같은 프로세스 P_i로부터 전송된 경우

통신 환경이 무손실, 활성 상태라고 가정하였으므로 프로세스 P_j는 메세지 m_1과 m_2를 수신하게 되며, P_i에서의 발생 순서에 따라 $VT_i(m_1)[i] < VT_i(m_2)[i]$이다. 그러므로 알고리즘에 따라 메세지 m_2는 m_1이 전달된 후에 전달된다.

경우 2. m_1과 m_2가 두 개의 다른 프로세스 P_i와 $P_{k(k\neq i)}$로부터 전송된 경우

프로세스 P_j에서 m_2가 m_1보다 먼저 전달될 수가 없음을 증명하면 다음과 같다. 먼저 m_1은 전달되지 않았고 프로세스 P_j는 M개의 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)) ≤ (i, sequece number(m_2))이다.

기본 단계: P_j가 응용 프로세스에게 전달한 첫번째 메세지는 m_2가 될 수 없다. 만약 P_j에게 어떤 메세지도 전달되지 않았다면, $VT_j(P_j)[i]=0$이다. 그러나 (i, sequence number(m_1))에서 sequence number(m_1) > 0 (왜냐하면 m_1이 P_i에 의해 전송되었으므로)이므로 수신 알고리즘의 단계 (1)과 (2)에 의해 m2는 Pj에게 첫 번째로 전달될 수 없다.

유도 단계: 프로세스 P_j는 M개의 메세지를 수신하였고, 그 중 어떤 것도 $send(m_1) \rightarrow send(m)$인 메세지 m이 아니라고 가정한다. 만약, m_1이 아직 전달되지 않았다면, $VT_j(P_j)[i]$ < sequence number(m_1)이다. $VT_j(P_j)[i]$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)) ≤ (i, sequece number(m_2))과 $VT_j(P_j)[i]$ < sequence number (m_1)에 따라 $VT_j(P_j)[i]$ < sequece number (m_2)가 유도된다. 그러므로 수신 알고리즘에 따라 프로세스 P_j에 M +1번째 전달되는 메세지는 m_2가 될 수 없다.

⊙ 활성(liveness)

[Theorem 2] 새로운 압축 방식 알고리즘은 오류가 없는 상황에서 결국 모든 메세지를 전달한다.

증명: 프로세스 P_i에 의해 전송되고 프로세스 P_j에 결코 전달될 수 없는 메세지 m이 존재한다고

가정하면, 수신한 CVT에서 수신 알고리즘 단계 (1)과 (2)에 의해 다음의 둘 중 하나가 성립된다.

- $(i, VT_i(m)[i])$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, 통신 채널은 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′→m가 반드시 존재한다. 모든 메세지는 모든 프로세스에게 송신된다는 가정 아래, m′는 이미 P_j에게 멀티캐스트되었다. 그러므로 m′는 통신 채널에 존재한다. 통신 시스템은 결국 모든 메세지를 전달하므로 m′는 P_j에게 수신되었다고 할 수 있다. 또한, m′에 적용한 같은 이유를 m에게도 적용가능하다. m이전에 전달되어야 할 메세지의 수는 유한적이고, →는 비순환이므로 이는 가정에 위배된다.

V. 성능 평가

본 장에서는 단일 그룹에 대해 기존의 VT 알고리즘, VT 압축 알고리즘, 제안된 알고리즘 등을 인과관계 순서화에 요구되는 메세지 오버헤드 측면에서 비교하였으며, 시뮬레이션 환경은 점대점(point-to-point) 네트워크로 가정하였다. 즉, 각 프로세스들이 그룹내 멤버에게 멀티캐스트를 수행할 때 메세지들간의 인과관계 순서화를 위해 메세지에 piggybacking하는 오버헤드 양을 비교하였다. 성능 비교시 acknowledgement, 네트워크 구성, 폭주, 라우팅 등은 고려하지 않았다. 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.

기존의 VT 압축 알고리즘을 분석해보면 다음과 같다. 한 프로세스가 멀티캐스트를 수행한 후 그룹내 모든 프로세스로부터 1개 이상의 메세지를 수신한 경우에 대해서는 다음의 멀티캐스트 수행시 벡터 타임스탬프 압축이 불가능하다. 즉, 그룹 멤버들이 순차적으로 멀티캐스트를 수행하는 환경에서는 압축이 불가능한 단점이 있다. 반면에 압축 효과가 가장 잘 나타나는 환경으로는 한 프로세스가 연속적인 멀티캐스트를 수행하는 경우(다른 프로세스로부터 메세지를 수신하지 않은 경우)이다. 이에 반해 제안된 압축 기법은 추가적인 벡터 타임스탬프를 유지함으로써 국부적인 메모리의 증가가 있지만, 메세지 오버헤드 관점에서 모든 프로세스로부터 1개 이상의 메세지를 수신하더라도 송신하고자 하는 프로세스의 벡터 타임스탬프 정보에 따라 압축이 가능하다.

표 1은 n개의 프로세스 그룹 환경에서 한 프로세스가 멀티캐스트를 수행하기 전에 타임스탬프를 송신한 프로세스에 따른 분석이다. 벡터 타임스탬프의 각 압축 field는 생략 가능한 프로세스의 순차번호를 나타낸다. 만약 다른 프로세스로부터 2개 이상의 field

표 1. VT 압축 기법과 제안된 압축 기법의 비교
Table 1. Comparison of the VT compression method and the proposed compression method

타임스탬프를 송신한 프로세스의 수	기존 알고리즘의 압축 field	새로운 알고리즘의 압축 field
0개	n−1 field	n−1
1개	n−2 field	{(n−1)+(n−2)×(n−2)}/n−1
2개	n−3 field	{(n−1)+(n−2)+(n−3)×(n−3)}/n−1
⋮	⋮	
n−3개	2 field	{(n−1)+(n−2)+...+2×2}/n−1
n−2개	1 field	{(n−1)+(n−2)+...+2+1}/n−1
n−1개	압축 불가능	{(n−1)+(n−2)+...+2+1}/n−1

가 다른 타임스탬프를 수신하는 경우, 인과관계 순서화를 위해 타임스탬프에 해당하는 메세지를 기다려야 하므로 결국 2개 이상의 프로세스로부터 타임스탬프를 수신한 것으로써 분석했다.

컴퓨터 시뮬레이션을 통하여, 인과관계 순서화를 제공한 기존 VT 알고리즘, VT 압축 알고리즘, 제안된 압축 알고리즘 등을 메세지 오버헤드 측면에서 비교, 분석하였다. 시뮬레이션은 단일 그룹의 프로세스 수를 변화시키면서 인과관계 순서화에 요구되는 메세지의 오버헤드를 비교하였다. 그림 4와 그림 5는 각 프로세스들이 많은 메세지를 멀티캐스트한 결과를 평균하여 그룹내 프로세스들이 한 번씩 멀티캐스트를 수행하는 데 필요한 메세지 오버헤드를 측정하였다.

그림 4는 그룹내 프로세스들이 같은 지연 타이머(delay timer)를 이용하여, 메세지를 랜덤하게 발생시킨 후 실험한 결과이다. 예를 들어 3개의 프로세스로 구성된 그룹에서 각 프로세스들이 한 번씩 멀티캐스트를 수행하는 데 필요한 메세지 오버헤드의 평균 field는 VT 알고리즘의 경우 18 field, VT 압축 알고리즘의 경우 14.88 field, 제안된 알고리즘의 경우 10.96 field가 소요되었다. 시뮬레이션 결과를 분석해보면, 모든 프로세스들이 같은 지연 타이머를 이용하므로 한 프로세스가 멀티캐스트를 수행하기 전에 다른 프로세스로부터 메세지를 수신하는 경우가 많이 발생하게 된다. 그러므로 기존의 VT 압축 알고리즘을 적용하면, 이전에 전송한 타임스탬프와 현재의 갱신된 타임스탬프의 많은 field가 다르게 된다. 즉, 기존 VT 압축 알고리즘의 압축 성능이 떨어진다. 그러나 제안된 알고리즘의 경우는 각 프로세스가 전송한 타임스탬프 정보를 이용하므로 VT 알고리즘보다 압축 성능이 우수하다. 시뮬레이션 결과 같은 지연 타이머를 사용할 경우 최대 30%의 성능 향상을 볼 수 있었다.

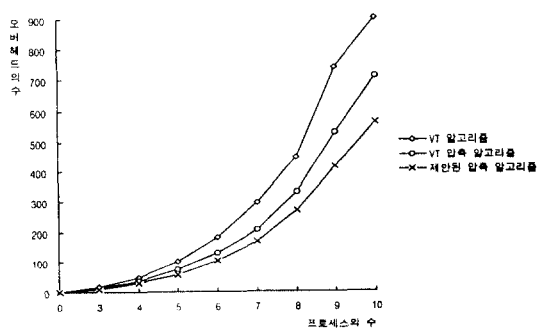

그림 4. 같은 지연 타이머를 사용할 때 메세지 오버헤드 비교

Fig. 4 Comparison of the message overhead with the same delay timer

VT 압축 알고리즘의 분석에 따르면 다른 프로세스로부터 메세지를 수신하기 않고 연속적인 멀티캐스트를 수행할 경우가 압축 성능이 가장 좋다. 이를 확인하기 위해 그림 5는 그룹내 각 프로세스들이 서로 다른 지연 타이머를 사용한 환경에서 메세지를 랜덤하게 발생한 결과이다. 즉, 프로세스들이 서로 다른 지연 타이머를 사용하므로 임의의 프로세스는 연속적으로 메세지를 멀티캐스트할 경우가 많이 발생한다. 그림 5의 결과를 살펴보면, 기존의 VT 압축 알고리즘의 성능이 많이 향상됨을 확인할 수 있다. 그러나 시뮬레이션 결과 다른 지연 타이머를 사용하는 환경에서도 제안된 알고리즘이 기존의 VT 압축 알고리즘보다 최대 20%의 성능 향상을 볼 수 있었다.

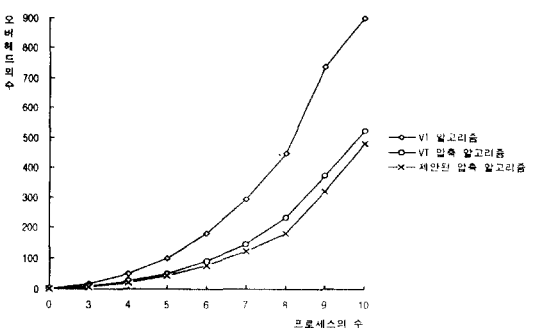

그림 5. 다른 지연 타이머를 사용할 때 메세지 오버헤드의 비교

Fig. 5 Comparison of the message overhead with the different delay timer

Ⅵ. 결 론

본 논문에서는 하나의 프로세스가 정적인 단일 그룹에 속하는 환경에서 프로세스간에 송수신되는 메세지들의 인과관계 순서화를 위한 효율적인 벡터 타임스탬프 기법을 제시하였다.

기존의 VT 압축 알고리즘은 이전에 전송한 메세지의 벡터 타임스탬프와 현재의 갱신된 벡터 타임스탬

프를 비교하여 변화된 부분만 전송하므로, 프로세스들간의 메세지 송수신이 많은 환경에서는 압축 성능이 떨어진다. 이에 반해 제아된 압축 알고리즘은 국부적으로 유지하고 있는 다른 프로세스의 벡터 타임스탬프 정보를 이용하여 개별적인 압축을 수행하므로 메세지 송수신 양에 관계없이 효율적으로 압축을 수행한다. 또한 제안된 압축 방식의 인과관계 순서화를 논리적으로 증명하였으며, 시뮬레이션을 통해 기존의 VT 압축 알고리즘과의 성능 비교를 수행하였다. 그 결과 제안된 압축 알고리즘은 프로세스간의 통신 형태에 관계없이 인과관계 순서화에 요구되는 메세지 오버헤드의 압축 성능이 우수함을 확인하였다. 그러므로 새로운 압축방식을 멀티캐스트 통신 환경에 이용하면, 기존의 압축 방식에 비해 프로세스간 인과관계 순서화에 요구되는 메세지 오버헤드를 더욱 줄일 수 있다.

향후 연구 과제로는 제안된 압축 방식을 다중 그룹 환경 및 동적인 그룹 환경으로 확장하여 적용하는 것과 통신망 상에서 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 Communication", 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.

권 봉 경(Bongkyung Kwon) 정회원

1995년 : 광운대학교 전자통신공학과 학사
1997년 : 광운대학교 전자통신공학과 석사
1997년~현재 : 대우통신 교환연구단 교환연구5실 연구원
※주관심분야 : 멀티미디어통신, 분산처리

정 광 수(Kwangsue Chung) 정회원

1981년 : 한양대학교 전자공학과 학사
1983년 : 한국과학기술원 전기 및 전자공학과 석사
1991년 : 미국 University of Florida 전기공학과 박사(컴퓨터공학전공)
1983년~1993년 : 한국전자통신연구소 선임연구원
1991년~1992년 : 한국과학기술원 대우교수
1993년~현재 : 광운대학교 전자공학부 부교수(신기술연구소 연구원)
※주관심분야 : 멀티미디어통신, 컴퓨터통신, 분산처리