실시간 환경을 위한 효율적인 인과순서 알고리즘

장 익 현†

요 약

인과순서 메시지 전달 알고리즘은 모든 전송되는 메시지가 인과순서로 전달되도록 한다. 인과순서를 유지하기 위해서는 전송되는 모든 메시지가 제어정보를 가지고 다녀야 하고, 제어정보의 크기는 관련된 프로세스의 수에 비례하여 커지게 되므로 제어정보의 크기를 줄이는 것은 분산시스템의 주요한 관심사가 되어 왔다. 본 논문에서는 실시간 성질을 가지는 멀티미디어 자료를 위한 효율적인 △-인과순서 알고리즘을 제안하고 평가하였다. 제안된 알고리즘은 전송 부하를 줄이기 위하여 인과순서를 유지하는데 필수적이지 않은 정보를 가능한 이른 시기에 찾아내어 제거하며, 기존 알고리즘보다 훨씬 적은 전송부하를 가지게 된다.

An Efficient Causal Order Algorithm for Real-Time Environment

Ik-hyeon Jang†

ABSTRACT

Causal order of message delivery algorithm ensures that every transmitted message is delivered in causal order. It should be noted that control information should be transmitted with each message in order to enforce causal order. Hence, it is important to reduce this communication overhead because the impact of the overhead increases proportionally with the number of related processes. In this paper we propose and evaluate effective △-causal order algorithm for multimedia data which have real time property. To reduce transmission overhead, proposed algorithm eliminates redundant information as early as possible which is not explicitly required for preserving causal order. Average communication overhead of our algorithm is much smaller than other existing algorithms.

키워드 : 인과순서(causal order), △-인과순서(△-causal order), 분산시스템(distributed system), 멀티미디어(multimedia)

1. 서 론

분산시스템은 여러 프로세스들이 서로 메시지를 교환하면서 공조하여 하나의 작업을 수행한다. 그러나 메시지 전송시간은 유한하지만 전달시간의 차이 때문에 송신하는 메시지와 전달되는 메시지들의 순서가 일치하지 않을 수 있다. 이러한 메시지 순서에 관련된 불일치성을 해결하기 위하여 인과순서(causal order)가 제안되었다[4]. 분산시스템에서의 인과순서는 Lamport가 제안한 *happened before* 관계에 기초를 두고 있으며, 두 개의 메시지가 서로 관련되어 있고 같은 목적지를 가진다면 송신한 순서와 같은 순서로 목적지에 전달할 것을 요구한다[9].

멀티미디어 통신시스템에서 응용프로그램이 전달한 메시지는 전송된 후 일정 시간이 경과하면 데이터로서의 의미를 잃게 되는 특성을 갖는다. 이때 전송된 메시지의 전송제한시간을 유효시간(lifetime)이라고 하며 이 시간 안에 목적지에 전달되는 메시지는 의미가 있는 데이터로 응용프로그램에서 이용될 수 있다. 멀티미디어 응용프로그램을 지원하기 위하여 데이터의 유효시간을 고려한 △-인과순서(△-causal order)가 제안되었으며[15] 이는 인과순서의 확장으로 볼 수 있다.

인과순서와 관련된 주요 이슈로는 메시지 간의 인과관계를 파악하는 방법과 인과관계를 유지하기 위해 교환되는 제어정보의 양을 최소화하는 방법이 있다. 인과순서에서는 주로 논리적 시간을 사용하여 인과관계를 파악하고 제어정보의 양을 줄이기 위한 방법을 모색하였다[2, 5, 7, 8, 11, 12]. 그러나 △-인과순서는 실시간적 특성을 가지는 자료를 대상으로 하기 때문에 논리적 시간이 아니라 물리적 시간을 사용하여야 한다. 실시간 특성을 가지는 멀티미디어 자료는 점점 그 응용분야가 넓어지고 있으며 이와 관련된 연구도 계속되고 있다[1, 3, 13-15].

본 논문에서는 전송되는 메시지에 대한 정보의 인식 정도에 따라 통신패턴을 정의하고 분석하였으며, 인과관계의 파악은 송신프로세스를 기준으로 제어정보를 정리하는 방

† 종신회원 : 동국대학교 정보통신공학과 조교수
논문접수 : 2004년 12월 16일, 심사완료 : 2005년 1월 7일

법을 사용하여 메시지에 대한 제어정보의 우선순위 파악이 쉽도록 하였다. 통신패턴에 따라 △-인과순서를 유지하는데 필요한 제어정보의 형태와 양은 달라진다. 본 논문에서 제안한 알고리즘에서는 중복되는 제어정보의 조기 삭제를 통해 중복되는 제어정보의 전송은 억제하고 필수적인 제어정보만 전송하도록 하여 △-인과순서 유지에 필요한 제어정보의 양을 최소화하도록 하였다. 알고리즘의 정확성을 증명하고, 제어정보의 양에 대한 기존 알고리즘과의 비교를 통해 제안한 알고리즘을 평가하였다.

본 논문의 구성은 다음과 같다. 2장에서는 △-인과순서화의 개념을 설명하고 3장에서 관련 연구를 소개하며 4장에서 새로운 알고리즘을 제안한다. 5장에서는 알고리즘의 평가를 하였으며 6장에서 결론을 맺는다.

2. 이론적 배경

2.1 인과순서

분산시스템에서 사건의 순서를 정의하는 능력은 많은 응용분야에서 아주 중요하다. 순차적인 프로세스들만 고려할 경우 하나의 프로세스에서 수행되는 모든 사건들은 전체가 순서화 되어야 한다. 따라서 Lamport[9]의 *happened before* 관계를 → 기호로 나타낸다면 인과순서전달은 다음과 같이 정의할 수 있다.

[정의 1] (인과순서전달) : 메시지 M_1과 M_2가 같은 목적지를 가지고 있으며 send(M_1) → send(M_2)이면 deliver(M_1) → deliver(M_2)이다.

2.2 메시지의 유효시간

메시지의 유효시간(△)은 메시지가 전송되고 나서 목적지에 도달하여 사용될 수 있는 물리적인 시간을 의미하는 것으로 유효시간이 지나서 전달된 메시지는 사용할 수 없다. 즉 모든 메시지의 유효시간은 *메시지의 출발시간* + △가 되며 유효시간을 넘긴 후에 도착한 메시지는 효용성이 없으므로 폐기된다. 따라서 유효시간 이전에 목적지에 도착된 메시지는 유효시간이 지나기 전에 목적프로세스에 전달되어야 한다.

멀티미디어시스템에서 메시지의 유효시간은 서비스의 질을 저하시키지 않는 최대지연시간으로 정의할 수 있으며 멀티미디어 응용에 따라 차이가 있다. 예를 들어 화상회의 시스템의 음성과 화상데이터는 250ms, 화상전화의 경우는 350ms, 비대화형 비디오시스템은 1초, 카탈로그 검색시스템의 최대 지연시간은 2초 정도이다[6]. 문제를 단순화하기 위하여 본 논문에서는 모든 메시지는 동일한 유효시간을 가지는 것으로 가정한다.

실시간 전송 제한조건을 만족시키기 위해서는 모든 프로세스들이 인과관계가 있는 사건들을 모두 다른 물리적 시간에 발생시킬 수 있을 만큼의 정확도(granularity와 precision)를 가지도록 하는 가상전역시간(virtual global clock)을 가정하며 가상전역시간에서는 동일한 메시지에 대한 송신과 수신이 동시에 일어날 수는 없도록 δ_t-*precedence*를 가정한다[15].

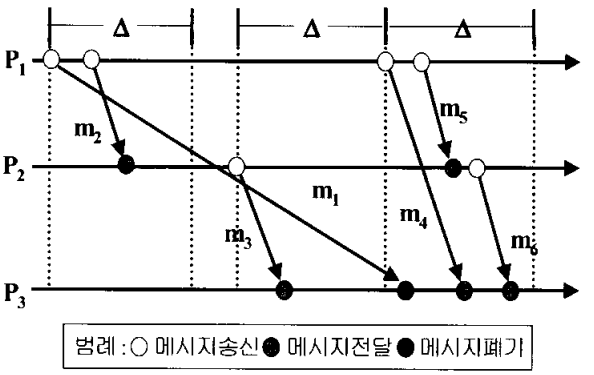
인터넷에서의 시간동기화에 대한 연구 결과 시간편차(clock drift)는 5-10ms 이내까지 제공될 수 있고[10], 각 응용에서 요구하는 유효시간은 그보다 충분히 크므로, 시간동기화 알고리즘이 작동하는 것을 가정하여 시간편차를 유효시간에 포함시키도록 한다.

2.3 △-인과순서

△-인과순서는 Yavatkar에 의해 소개된 것으로 인과관계를 기반으로 메시지가 실시간 데이터를 포함하고 있는 신뢰할 수 없는 통신시스템을 가정하며 다음과 같이 정의한다[15].

[정의 2] (△ 인과순서) : 다음 두 가지 조건을 만족하면 △-인과순서가 유지된다.

1) 유효시간 내에 도착하는 모든 메시지는 유효시간 내에 전달되고 그 외의 모든 메시지는 폐기된다.
2) 모든 전달되는 메시지는 인과순서를 준수한다.



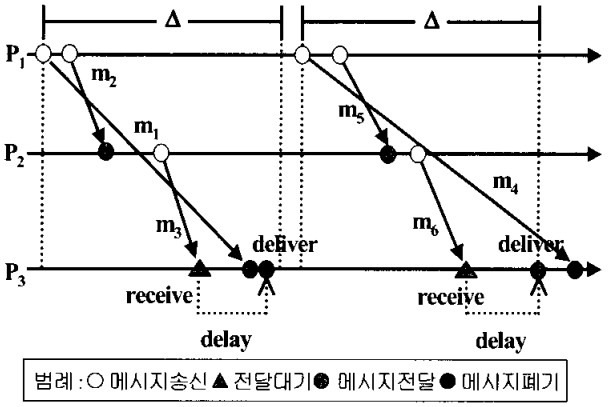
(그림 1) △-인과순서와 인과순서 전달

오디오나 비디오 데이터는 *better-never-than late* 성격을 가지기 때문에 △-인과순서 메시지 전달은 오디오나 비디오와 같은 멀티미디어 데이터의 전송에 적합한 전송방식이다. 데드라인을 맞추지 못하고 도착한 오디오, 비디오 데이터는 전혀 쓸모가 없을 뿐만 아니라 오히려 정상적인 데이터의 처리에도 방해가 된다.

(그림 1)은 인과순서를 지키지 못한 경우에도 △-인과순서가 어떻게 지켜질 수 있는지를 보여준다. 메시지의 유효기간은 메시지가 발신된 후 △시간까지로 가정한다. 먼저 m_1은 m_3 보다 △시간 이상 전에 보내진 것이기 때문에 이들 사이에는 △ 인과관계가 없다. 따라서 m_3가 P_3에 도착한 순간에 m_1이 아직 도착하지 않았을 경우에도 m_3는 버퍼링되지 않는다. 만약에 m_3가 m_1이 도착할 때까지 버퍼링되어 m_1을 기다릴 경우 두 메시지 모두 메시지의 유효기간을 초과하게 되어 폐기되므로 기다리지 않을 경우보다 더 많

은 데이터가 폐기되기 때문이다. m_4와 m_6의 경우 두 메시지 모두 △시간 내에 목적지에 도달하였으므로 m_6의 전달은 △-인과관계를 어기지 않고 인과순서에 따라 전달된다.

그러나 두 메시지 사이에 △-인과순서가 성립될 경우에는 메시지의 데드라인을 초과하지 않을 때까지는 최대한 기다려야 한다. (그림 2)에서 m_3은 m_1보다 먼저 P_3에 도착하였으나 m_1의 유효시간이 지나지 않았기 때문에 인과순서를 유지하기 위하여 m_1이 도착할 때까지 기다렸다가 m_1이 전달된 후에 전달된다. 그러나 m_6의 경우 인과순서로 보면 m_1이 전달된 후에 전달되어야 하나 m_1이 유효시간 내에 도착하지 않았기 때문에 m_6는 데드라인까지 기다렸다 전달된다. 그 후에 m_1이 도착하면 m_1은 자신의 데드라인을 지났으므로 폐기된다.



(그림 2) △-인과순서에 따른 메시지 전달

3. 관련 연구

Yavatkar[15]는 관련된 연결간의 트래픽 전달에서의 시간적 동기화를 유지하기 위하여 실시간 데이터의 지연시간 제한을 고려한 △ 인과순서를 최초로 제안하였다. Adelstein과 Singhal[1]은 △-인과순서를 유지하는 효율적인 알고리즘을 제안하였지만 그들의 알고리즘은 2단계 이내에 있는 프로세스 간의 인과관계만 유지시켜주는 *triangle inequality* 문제를 해결한 것이다. 2단계 이상 떨어진 프로세스 간의 인과관계의 유지는 보장할 수 없으며, 멀티캐스팅도 반복적인 유니캐스트를 통해 구현하였으며 중첩되는 멀티캐스트 그룹에 대한 지원도 되지 않으므로 그들의 알고리즘은 일반적이지 못하다. Baldoni[3]는 Raynal[12]의 인과순서 알고리즘을 실시간 성질을 가지는 데이터도 처리할 수 있도록 수정한 알고리즘을 제안하였으며, Prakash[11]는 인과순서 멀티캐스트 알고리즘을 수정하여 실시간 데이터에 대한 방송 알고리즘을 제안하였다. Rodrigues[13]는 실시간 데이터를 데이터의 유효기간 내에 최대한 많이 보낼 수 있는 멀티캐스트 알고리즘을 제안하였다.

Baldoni[2]와 Kshemkalyani[8]는 인과순서를 유지하는데 필요한 최소한의 제어정보의 양을 계산하기 위한 연구를 하였으며 그들의 연구 결과는 최악의 경우에는 최소한 $O(n^2)$의 시간, 공간 복잡도를 가져야 함을 보여주고 있다. Jang[7]은 중복정보의 관리를 통해 인과순서를 유지하기 위하여 필요한 제어정보의 양을 최소화하는 방법을 제안하였다.

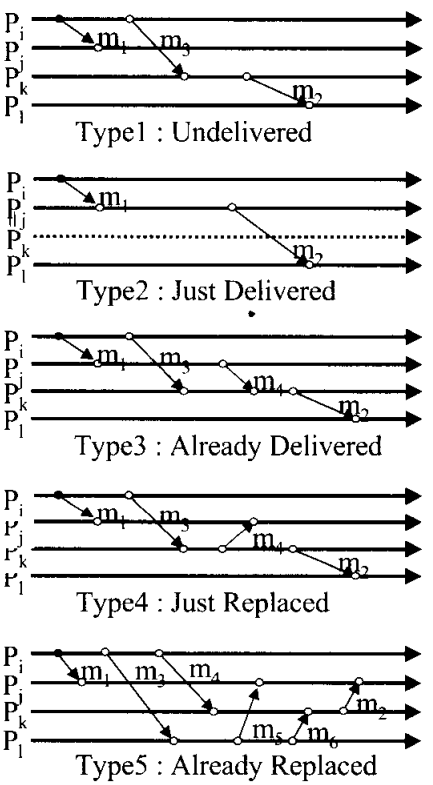
대부분의 △-인과순서 알고리즘은 멀티미디어 자료의 실시간적 특성을 고려하기 위하여 인과순서 알고리즘에서 사용하던 논리시간을 물리시간으로 변경한다. 본 논문에서는 Jang[7]의 알고리즘을 기반으로 실시간 자료를 처리할 수 있도록 한다.

4. △-인과순서 알고리즘

4.1 메시지 패턴의 분류

전송되는 메시지 m에 대한 정보를 가지는 상태를 송신프로세스와 목적프로세스를 기준으로 분류하면 모든 프로세스는 다음의 네 가지 중 하나의 상태에 속한다.

1) 송신프로세스의 상태
2) 목적프로세스의 상태
3) 송신프로세스와 목적프로세스 사이의 프로세스의 상태
4) 목적프로세스 이후에 있는 프로세스의 상태



(그림 3) 추상통신패턴

만약 두 개의 프로세스가 동일한 상태 정보를 가지고 있다면 그 두 프로세스는 메시지 m에 관한 정보에 대해서는 하나의 프로세스로 볼 수 있다. 따라서 분산시스템은 4개의 프로세스로 이루어진 시스템으로 변환이 가능하며, 4 프로

세스 시스템에서의 모든 유효한 통신 패턴은 (그림 3)에 있는 추상통신패턴 중의 하나로 변환이 가능하다[7].

(그림 3)의 추상통신패턴은 프로세스 P_k가 메시지 m_2를 전송하는 시점에서 m_1에 대한 정보를 어떻게 가지고 있는가를 기준으로 분류한 것으로 Type1에서 Type5까지로 분류된다.

먼저 송신프로세스 자체와 송신프로세스와 목적프로세스 사이에 있는 프로세스는 메시지가 목적프로세스에 전달되었는지를 알지 못한다. 이런 프로세스들은 Type1로 취급된다. 목적프로세스는 해당 메시지(m_1)가 전달되었다는 것을 바로 알 수 있으며 Type2가 된다. 반면에 목적프로세스 이후에 있는 프로세스들은 m_1이 목적프로세스에 전달되었다는 것을 이미 알고 있으므로 Type3가 된다.

목적프로세스와 송신프로세스 사이에 있는 프로세스들은 m_1 이후에 인과순서상 뒤에 있는 다른 메시지가 동일한 목적프로세스로 전송될 경우에 그들 간의 순서를 반영하여 인과관계를 판단할 수 있다. 즉 인과순서상 뒤에 있는 메시지가 전달되었다면 그들 간의 인과순서를 유지하기 위하여 앞의 메시지는 당연히 전달되었을 것이므로 인과순서상 뒤의 메시지에 대한 정보만 유지해도 된다. 이런 경우에 앞의 메시지에 대한 정보는 뒤의 메시지에 의해 대치되었다고 한다. 따라서 Type4에서 m_1은 m_4에 의해 대치되었다고 한다. 만약에 어떤 메시지 m을 프로세스 P_j에 m_4와 인과관계를 유지하여 전달하였다고 하자. 이 경우 m_4가 P_j에 전달되기 위해서는 m_1이 전달된 후에 전달되어야 하므로 m은 메시지 m_1과도 인과관계를 유지하게 된다. 이런 경우의 통신패턴이 Type4가 되며 이를 일반화하면 Type5가 된다.

4.2 중복정보의 제거

Type1 프로세스들은 m_1이 목적지프로세스에 도달하였는지의 여부를 알지 못하므로 m_1에 대한 정보는 인과관계의 유지를 위해 꼭 필요한 정보이다. 따라서 Type1 프로세스들은 m_1에 대한 정보를 유지해야 한다. Type2 프로세스들은 m_1이 이미 목적지 프로세스에 도달하였다는 것을 알고 있다. 따라서 m_2 이후의 메시지들은 m_1에 대한 정보를 유지할 필요가 없어진다. 만약 Type2 프로세스들이 어떤 이유로 메시지 m_1에 대한 정보를 가지고 있어야 한다면 그 정보는 인과순서를 유지하는데 필수적이지 않으므로 중복정보로 취급된다. 즉 중복정보는 Type2부터 Type5 사이에 해당하는 프로세스들이 인과순서를 유지하는데 필수적이지 않은 정보를 가지는 경우로, 이러한 정보는 파악되고 제거되어야 한다.

효과적인 인과순서알고리즘은 중복정보의 유지와 전송을 최소로 해야 한다. 본 논문에서는 중복의 성격에 따라 중복정보를 일차중복정보와 이차중복정보로 분류한다. 일차중복정보는 인과순서를 유지하는데 필수적이지 않은 정보로 (그림 3)에서 정의한 추상통신패턴의 Type2부터 Type5까지의 유형에서 나타난다. 일차중복정보를 제거하면 모든 제어정보는 인과순서의 유지를 위한 필수적인 정보로만 한정된다. 그러나 필수적인 정보도 두 번 이상 전송된다면 해당 정보는 결과적으로 불필요하게 전송된 것이므로 중복정보가 된다. 이런 종류의 중복정보를 이차중복정보로 분류하며 이차중복정보의 제거를 위해서는 별도 정보의 전송을 필요로 하게 된다. 따라서 이차중복정보의 제거는 반드시 도움이 된다고 하기는 어렵다. 따라서 본 논문에서는 일차중복정보의 제거에 대해서만 다루기로 한다.

동일한 프로세스에서 전송된 메시지들 간의 인과순서는 송신순서를 이용하면 쉽게 파악된다. 하나의 프로세스에는 동일한 프로세스에서 전송한 메시지에 대해서 최종 메시지에 대한 정보만 유지하면 되므로 나머지 메시지에 대한 정보는 중복정보가 된다. 중복정보는 전송된 메시지와 함께 수신된 제어정보와 각 노드에 있는 제어정보를 비교하여 파악하며 인과순서상 가장 뒤에 있는 메시지에 대한 정보만 유지하도록 하여 중복정보를 제거한다. 단 이 경우 모든 송신 노드에 대해 최소한 하나의 메시지에 대한 정보는 남겨 두어서 이후에 사용할 수 있도록 한다.

4.3 인과관계의 정리

인과관계의 파악은 제어정보를 정리하는 방법에 따라 두 가지로 분류할 수 있다. 첫 번째 방법은 목적프로세스에 따라 인과관계를 정리하는 것으로 지금까지 대부분의 알고리즘은 이 부류에 속한다. 이 방법은 전달조건의 검사가 쉽다는 장점을 가지고 있으나 인과관계들 간의 우선순위를 파악하기 위해서는 전체 제어정보를 확인해야 하는 어려움이 있다. 두 번째 방법은 송신프로세스를 기준으로 인과관계를 정리하는 것으로 전송되는 메시지와 그와 함께 전송되는 제어정보를 하나의 객체로 처리한다. 이 방법에서는 목적프로세스를 기준으로 인과관계를 검사해야 하는 전달조건의 검사는 상대적으로 어려워서, 전달조건을 검사하기 위해서는 전체 인과관계정보를 확인해야 한다. 그러나 우선순위 관계를 파악하는 것이 쉽다는 장점을 가진다.

본 논문에서는 제어정보들 간의 우선순위를 찾는 것이 중요하므로 두 번째 방법을 채택하며 전달조건의 확인을 빨리 하기 위하여 메시지 송신 시에 전달조건 확인을 위한 정보를 목적프로세스를 기준으로 별도로 추출하여 전달하도록 한다.

4.4 자료구조 및 전달조건

가. 자료구조 및 표기법

실시간 자료를 처리하기 위한 △-인과순서에서는 논리시간을 사용할 수 없으므로 변수 *c_time*을 사용하여 현재의 물리시간을 나타내도록 한다. *c_time*은 현재의 시간을 나타내도록 계속적으로 갱신된다고 가정한다. m^{τ}_{k}는 P_k가 τ 시간에 보낸 메시지를 의미하며 *m*.D는 *m*의 목적지를 나타낸다.

제어정보는 (τ, k)의 형태로 송신노드 별로 분류한다. 노드 P_i는 노드 P_j에 대한 제어정보를 $CI_i[j]$로 관리하며 메시지를 보낼 때마다 함께 보내며 메시지와 함께 보내는 제어정보는 CIm으로 표기한다. 만약에 $(\tau, k) \in CI_i[j]$ 이면 이

는 P_i가 P_k에게 보내는 모든 메시지는 P_j가 P_k에게 τ시간 이전에 보낸 모든 메시지가 전달된 후에 전달되어야 한다는 것을 나타낸다. 목적지에서의 전달조건을 신속하게 확인하기 위하여 P_i가 P_k에게 보내는 메시지에는 $(\tau, m.D) \in CI_i[j]$ 이고 $k \in m.D$이면 (j, τ)가 $DC_m[k]$에 포함되도록 제어정보를 DC로 별도로 추출하여 메시지와 함께 보낸다. 또 각 노드에는 노드 k로부터 노드 i에 전달된 최종 메시지의 송신시간을 나타내는 $DLV_i[k]$를 유지한다.

나. 전달조건

(정의 2)의 Δ-인과순서를 유지하기 위해서는 전달되는 모든 메시지는 다음과 같은 전달 조건을 만족시켜야 한다. 어떤 프로세스 P_i에 도착한 메시지는 메시지와 함께 전송된 제어정보에 포함된 인과관계 중에서 P_i를 목적지로 하는 메시지들이 모두 P_i에 전달되었거나(DCD_2) 그들의 유효시간이 지났다는 것이 확인되면(DCD_1) P_i에 전달될 수 있다. 이 조건이 만족되지 않으면 해당 메시지는 조건이 만족될 때까지 목적프로세스에 전달되지 않고 대기하여 Δ-인과순서를 유지하도록 하며, 이는 다음 식으로 표시된다.

$\forall (k,\tau) \in DC_m[i]$:

$\tau + \Delta < c\_time$: (DCD_1)

$\tau \le DLV_i[k]$: (DCD_2)

4.5 알고리즘

모든 인과순서알고리즘은 기본적으로 메시지 m을 송신하는 경우에 m보다 인과순서상 앞선 메시지에 대한 제어정보를 함께 전송해야 한다. 프로세스 p_i가 메시지 m을 $p_j \in m.D$인 프로세스에게 보내는 멀티캐스트 송신알고리즘은 다음과 같다.

가. 송신알고리즘

```
begin
    τi := c_time;
    DCi := ∅;
    for j ∈ m.D and m^τk ∈ CIi
        if (j ⊂ m^τk.D) then
            DCi[j] := DCi[j] ∪ {(k, τ)};
                m^τk.D := m^τk.D - {j};
            endif
    endfor
    for m^τk ∈ CIi
            if ((τ + Δ) < c_time) then m^τk.D := ∅;
    endfor
    for m^τk, m^υk ∈ CIi
            if (m^τk.D = ∅ and τ < υ) then CIi := CIi - m^τk;
    endfor
    for j ∈ m.D
            SEND (pi, τi, m.D, CIi, DCi[j], m) to pj;
    endfor
    CIi[i] := CIi[i] ∪ {(τi, m.D)};
end
```

나. 수신알고리즘

프로세스 p_i가 메시지 m을 프로세스 p_j로부터 받아 처리하는 수신알고리즘은 다음과 같다. 먼저 수신한 메시지의 데드라인을 검사하여 데드라인을 지난 메시지는 폐기한다. 데드라인 검사를 통과한 메시지에 대해서는 DC_m과 DLV_i를 비교하여 전달조건을 검사한다.

```
begin
    Sif (τm + Δ < c_time) then
        Discard m
    wait (∀(k,τ) ∈ DCm : (τ ≤ DLVi[k] or (τ+Δ) < c_time));
    Deliver m to pi;
    DLVi[j] :=τm;
    for m^τk ∈ CIi, m^υk ∈ CIm
        if (m^τk ∉ CIm and τ < υ) then CIi := CIi - m^τk;
        if (m^υk ∉ CIi and τ > υ) then CIm := CIm - m^υk;
    endfor
    for m^τk ∈ CIi, m^υk ⊂ CIm and τ= υ
        m^τk.D := m^τk.D ∩ m^υk.D;
        CIm := CIm - m^υk;
    endfor
    CIi := CIi ∪ CIm;
    for m^τk, m^υk ∈ CIi
        if (τ < υ and l ∈ m^τk.D ∩ m^υk.D) then m^τk.D := m^τk.D - {l};
    endfor
    for m^τk ∈ CIi
        if ((τ + Δ) < c_time) then m^τk.D := ∅;
    endfor
    for m^τk, m^υk ∈ CIi
        if (m^τk.D = ∅ and τ < υ) then CIi := CIi - m^τk ;
    endfor
end
```

5. 알고리즘 평가

5.1. 정확성 증명

제안한 알고리즘이 Δ-인과순서를 보장한다는 것의 증명은 다음의 두 단계를 통해 이루어진다. 첫 번째 단계는 모든 전달은 Δ-인과순서를 유지한다는 것이며(*safety*), 두 번째 단계는 데드라인 이내에 수신된 모든 메시지는 데드라인 이내에 목적지에 전달되고 데드라인을 지나서 수신된 메시지는 모두 폐기된다는 것이다(*liveness*).

[정리 1] [*Safety*] 알고리즘은 Δ-인과순서 메시지 전달을 보장한다.

증명 귀류법으로 증명한다. 먼저 두 개의 메시지 m_1과 m_2가 각각 p_i와 p_j에 의해서 p_k에게 전송되었으나 이들이 인과순서에 어긋나게 전달되었다고 하자. 즉, $send(m_1) \rightarrow send(m_2)$이지만 $deliver(m_2) \rightarrow deliver(m_1)$라고 하자. m_2가 p_k에 전달되기 위해서는 전달조건 DCD_1이나 DCD_2가 만족되어야 한다. 즉, $\tau_{m2} + \Delta < c_time$이거나 $\tau_{m2} \le DLV_k[j]$이어야 한다.

■ $\tau_{m2} + \Delta < c\_time$ 인 경우

δ_t-*precedence*에 의해 *send* $(m_1) \rightarrow$ *send* (m_2)이면 $\tau_{m1} < \tau_{m2}$가 된다. 따라서 $\tau_{m1} < \tau_{m2} < c_time$이 성립한다. 따라서 첫 번째 경우에 m_2가 전달되는 순간에 m_1이 도착하지 않았다면, 향후 m_1이 도착하게 되면 m_1은 폐기되게 되며 이는 m_1 이 폐기되지 않고 전달된다고 하였던 가정에 위배된다. 두 번째 경우에 만약에 m_2가 전달되는 순간에 m_1이 도착해 있었다면 $\tau_{m1} < \tau_{m2}$이므로 m_1에 대한 전달조건 DCD_2가 참이 된다. 따라서 m_2의 전달조건보다 m_1의 전달조건이 먼저 만족되어 m_1이 먼저 전달되므로 *deliver*$(m_2) \rightarrow$ *deliver*(m_1)이라는 가정에 위반된다.

■ $\tau_{m2} \leq DLV_k[j]$ 인 경우

δ_t-*precedence*에 의해 *send* $(m_1) \rightarrow$ *send* (m_2)이면 $\tau_{m1} < \tau_{m2}$가 된다. 따라서 $\tau_{m1} < DLV_k[j]$가 성립된다. 이는 $DLV_k[j]$ 이전에 p_j에서 p_k로 보낸 모든 메시지는 전달되었거나 폐기되었다는 것을 의미한다. 가정에서 m_1이 전달되었으며, *send*$(m_1) \rightarrow$ *send*(m_2)이고, $\tau_{m1} < DLV_k[j]$이므로 m_1은 이미 전달된 것이 된다. 따라서 m_2가 먼저 전달되었다고 하는 가정에 위반된다.

따라서 제안된 알고리즘은 Δ-인과순서 메시지 전달을 보장한다.

[정리 2] [*Liveness*] 알고리즘은 다음을 보장한다. (i) 데드라인 안에 수신된 메시지는 모두 데드라인 이내에 인과순서를 유지하면서 목적지에 전달된다. (ii) 데드라인을 지나 수신된 메시지는 모두 폐기된다.

증명 (ii) 항목은 수신알고리즘의 첫 번째 문장에 의해서 자동으로 증명된다. (i) 항목은 귀류법으로 증명한다. 먼저 데드라인 이전에 도착하였으나 데드라인 이내에 전달되지 않은 메시지 m이 있다고 가정하자. 따라서 m의 데드라인 순간에 전달조건 DCD_1과 DCD_2에 의해 다음 조건 NDCD_1과 NDCD_2가 모두 참이 되어야 한다.

$$\forall\ (k, \tau) \in DC_m[i]:$$
$$\tau + \Delta \geq c\_time \quad : \quad \text{(NDCD\_1)}$$
$$\tau_{m1} < DLV_i[k] \quad : \quad \text{(NDCD\_2)}$$

메시지 m의 데드라인 순간에 $\tau_m + \Delta = c_time$이 된다. 따라서 NDCD_1에 의해 $\tau \geq \tau_m$이 된다. 이것은 가상전역시간이 δ_t-*precedence*를 따른다는 가정에 위배된다. 따라서 데드라인 이전에 도착한 모든 메시지는 데드라인 전에 모두 전달된다.

5.2. 알고리즘 복잡도 분석

대부분의 기존 인과순서 알고리즘에서 전송되는 제어정보의 양에 대한 최악공간복잡도는 $O(n^2)$이다[2, 3, 5, 8, 11, 12]. Adelstein[1]은 O(n)으로 훨씬 적은 복잡도를 가지는 Δ-인과순서 알고리즘을 제안하였지만 *triangle inequality*만 해결하여 두 단계 이상의 인과관계에 대해서는 처리하지 못하는 단점이 있다.

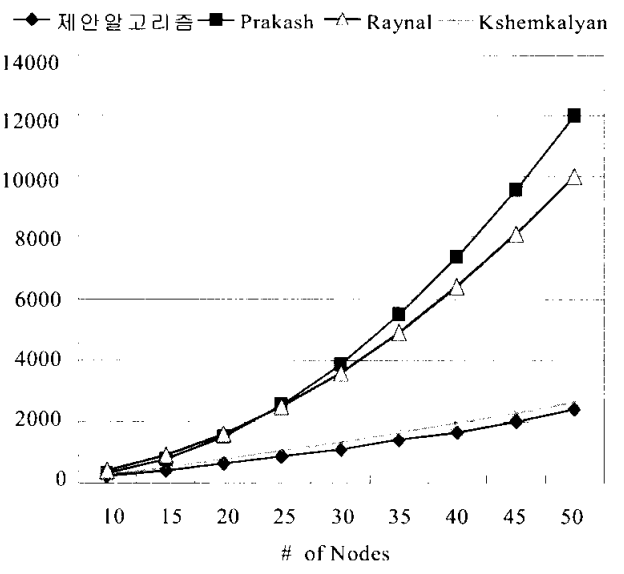
제안된 알고리즘의 공간복잡도를 계산하기 위해서는 CI, DC, DLV 세 개의 주요 자료구조에 대해 계산하면 된다. 여기서 DC는 CI의 일부분이므로 고려에서 제외하며, DLV도 일차원 배열로 공간복잡도는 $O(n)$이며 메시지와 함께 전송되지 않으므로 계산에서 제외한다.

CI 벡터의 개수는 n이며 각 항목에는 (τ_i, p_j) 형태의 자료가 들어있고 그 수는 시스템 내의 프로세스의 수로 제한된다. 따라서 CI 벡터의 최악공간복잡도는 기존의 알고리즘과 같은 $O(n^2)$이 된다. 그러나 평균공간복잡도는 다음의 모의실험 결과가 보여주듯이 중복정보의 조기 제거로 기존의 다른 알고리즘에 비해 상당히 적게 된다.

5.3. 모의실험

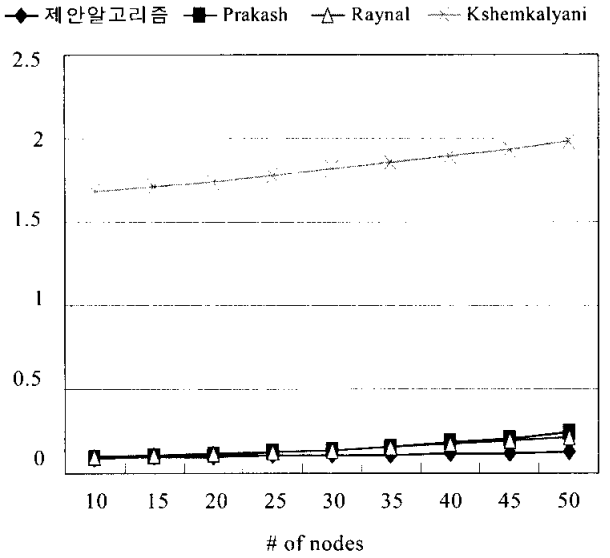
제안한 알고리즘과 기존 알고리즘과의 제어정보 전송량에 대한 평균공간복잡도를 비교하기 위하여 모의실험을 하였으며 시험 환경 및 조건은 다음과 같다.

- 시험시스템 : UltraSparc-II (296MHz, 512MB), SUN OS 5.6
  시험방법
  - 연속된 송신 간의 평균 시간은 0.1초이며 지수분포를 가정
  - 전송되는 메시지의 크기는 1024바이트로 고정
  - 최초의 10,000 메시지에 대한 자료는 버리고 이후의 50,000번의 메시지 전송에 대한 결과를 시험결과로 사용
- 비교대상 알고리즘 : Raynal[12]과 Prakash[11], Kshemkalyani[8]의 알고리즘



(그림 4) 전송되는 제어정보의 크기 비교

(그림 4)를 보면 제안된 알고리즘은 Kshemkalyani의 알고리즘과는 큰 차이를 보이지 않지만 Raynal이나 Prakash의 알고리즘보다는 훨씬 적은 제어정보(단위:바이트)를 전송하는 것을 알 수 있다. 특히 대상 노드의 수가 많아져도 전송되는 제어정보의 양이 선형증가를 하며, 기존 알고리즘과의 차이는 점점 커지는 것을 볼 수 있다. 이는 제안한 알고리즘의 확장성이 우수하다는 것을 보여준다. 결과적으로 제안된 알고리즘은 평균공간복잡도에서 기존 알고리즘에 비해 우수하다는 것을 알 수 있다.



(그림 5) 평균 메시지 전달 시간

(그림 5)는 제안한 알고리즘과 기존 알고리즘의 메시지 전달 평균시간(단위:ms)에 대한 결과를 보여준다. 제안한 알고리즘은 Kshemkalyani의 알고리즘을 제외한 나머지 알고리즘보다 약간 나은 정도의 성능을 보이고 있다. 앞의 (그림 4)와 함께 보면 Kshemkalyani의 알고리즘은 제어정보의 양은 제안한 알고리즘과 유사한 정도로 우수하지만 전달시간이 아주 많이 걸리는 것을 알 수 있다. Kshemkalyani의 알고리즘은 제어정보의 양은 적은데도 불구하고 전송시간이 많이 걸리는 것에서 알 수 있듯이 특히 많은 계산을 요구하고 있다는 것이다. 따라서 제안한 알고리즘은 제어정보를 계산하는데 필요한 시간도 많이 걸리지 않으면서 제어정보의 양도 적은 효율적인 알고리즘이라는 것을 알 수 있다.

6. 결 론

멀티미디어 데이터는 제한된 시간 내에 사용되지 않으면 데이터의 효용성을 상실하게 된다. 본 논문에서는 분산 환경에서 사용되는 멀티미디어 응용을 위한 효율적인 △-인과순서 알고리즘을 제안하였다.

△-인과순서에서는 기존의 인과순서에서 사용하던 논리시간을 사용할 수 없기 때문에 동기화된 물리시간을 사용하였으며, 인과관계를 목적지별로 분류, 처리하던 방식을 개선하여 발신지 별로 인과관계를 분류하면서 전달조건을 빨리 검사하기 위하여 필요정보를 별도로 추출하여 두 가지 방식의 장점을 취하였다. 인과순서를 제공하는 유효한 통신 패턴을 분석하여 5가지의 추상통신패턴으로 분류하고 인과순서를 유지하는데 필수적이지 않은 정보를 중복정보로 분류하여 제거함으로써 전송되는 제어정보의 양을 최소화하였다.

제안한 알고리즘의 정확성을 증명하였으며 기존 알고리즘과의 비교를 통해 제안한 알고리즘의 우수성을 보였다.

참 고 문 헌

[1] F. Adelstein and M. Singhal, "Real-Time Causal Message Ordering in Multimedia Systems," Proc. 15th ICDCS, pp.36-43, Jun. 1995.

[2] R. Baldoni and G. Melideo, "On the Minimal Information to Encode Timestamps in Distributed Computations," IPL, Vol.84, No.3, pp.159-166, Nov., 2002.

[3] R. Baldoni, R. Prakash, M. Raynal, & M. Singhal, "Efficient △-Causal Broadcasting," Int. Jour. of Comp. Syst. Sci. & Eng., pp.263-271, Sep., 1998.

[4] K. Birman and T. Joseph, "Reliable Communication in the Presence of Failure," ACM Trans. Comp. Syst., pp.47-76, 1987.

[5] W. Cai, B. Lee, and J. Zhou, "Causal Order Delivery in a Multicast Environment: An Improved Algorithm," JPDC, Vol.62, No.1, pp.111-131, Jan., 2002.

[6] D. Ferrari, "Client Requirements for Real-Time Communication Services," IEEE Communication Magazine, pp.65-72, Nov., 1990.

[7] I. Jang, J. Cho, and H. Yoonl, "An Efficient Causal Multicast Algorithm for Distributed System," IEICE Trans. Info. & Syst., Vol.E81-D, No.1, pp.27-36, 1998.

[8] A. Kshemkalyani and M. Singhal, "An Optimal Algorithm for Generalized Causal Message Ordering," Proc. 15th Sym. PODC, pp.87-94, May, 1996.

[9] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," Comm. of the ACM, Vol.21, No.7, pp.558-564, Jul., 1978.

[10] D. Mills, "Internet Time Synchronization: The Network Time Protocol," IEEE Trans. Comm., Vol.39, No.10, pp.1482-1493, Oct., 1991.

[11] R. Prakash, M. Raynal, and M. Singhal, "An Efficient Causal Ordering Algorithm for Mobile Computing Environments," Proc. 16th ICDCS, pp.744-751, May, 1996.

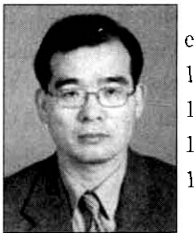
[12] M. Raynal, A. Schiper, and S. Toueg, "The Causal Ordering Abstraction and a Simple Way to Implement It," IPL, Vol.39, pp.343-350, Sep., 1991.

[13] L. Rodrigues, R. Baldoni, et al., "Deadline-Constrained Causal Order," Proc. 3rd IEEE Int. Sym. Object-Oriented Real-Time Distributed Computing, pp.234-243, Mar., 2000.

[14] P.Verissimo, "Causal Delivery Protocols in Real-Time Systems: A Generic Model," Real-Time Systems, Vol.10, No.1, pp.45-73, 1996.

[15] R. Yavatkar, "MCP: A Protocol for Coordination and Temporal Synchronization in Multimedia Collaborative Applications," Proc. 12th ICDCS, pp.606-613, 1992.

장 익 현



email : ihjang@dongguk.ac.kr

1984년 서울대학교 계산통계학과(학사)

1986년 KAIST 전산학과(공학석사)

1998년 KAIST 전산학과(공학박사)

1986년~1999년 (주)데이콤 종합연구소 책임연구원

1999년~현재 동국대학교 정보통신공학과 조교수

관심분야 : 컴퓨터통신, 분산시스템, 인터넷 프로토콜 등