변수화된 통신모델에서의 최적의 멀티캐스트 알고리즘 및 컴퓨터 구조에 따른 튜닝

이 주 영†

요 약

멀티캐스트는 중요한 시스템 레벨의 그룹 프로세스들을 수반하는 통신 서비스의 한 클래스이다. 소프트웨어 멀티캐스트 알고리즘을 설계하는데 있어서의 주된 문제는 성능과 이식성 사이의 교환조건(trade-off)을 고려하는 것이다. 본 논문에서 제안하는 변수화된 통신 모델은 LogP 모델의 확장으로 병렬 플랫폼의 통신 네트워크를 더 정확하게 특성화 할 수 있다. 이 변수화된 모델에서, 컴퓨터 구조에 의존적이지 않고 이식성 있는 OPT-tree라는 최적의 멀티캐스트 트리를 형성하는 알고리즘을 제안한다. 실제 여러 네트워크에 구현했을 때 진정한 최적의 수행을 달성하기 위해서 OPT-tree로 생성된 트리에서의 네트워크 위상에 따른 튜닝(tuning)에 대해 연구한다. 특히 웜홀 스위치를 사용하는 메쉬(mesh) 네트워크에서 변수화된 멀티캐스트 알고리즘의 최적화한 버전인 OPT-mesh 알고리즘을 개발하여 다른 알고리즘들과 비교하여 그 우수성을 검증한다.

Optimal Multicast Algorithm and Architecture-Dependent Tuning on the Parameterized Communication Model

Ju-Young Lee†

ABSTRACT

Multicast is an important system-level one-to-many collective communication service. A key issue in designing software multicast algorithms is to consider the trade-off between performance and portability. Based on the LogP model, the proposed parameterized communication model can more accurately characterize the communication network of parallel platforms. Under the parameterized model, we propose an efficient architecture-independent method, OPT-tree algorithm, to construct optimal multicast trees and also investigate architecture-dependent tuning on performance of the multicast algorithm to achieve the truly optimal performance when implemented in real networks. Specifically, OPT-mesh which is the optimized version of the parameterized multicast algorithm for wormhole-switched mesh networks is developed and compared with two other well-known network-dependent algorithms.

1. 서 론

통신 서브시스템은 병렬 연산에서 전체 시스템의 병목현상으로 여겨지므로 많은 노력 및 연구들이 활발히 이루어지고 있다. 집단화 통신(*Collective communication*)[1]이라 불리는 통신 서비스의 한 클래스는 그룹 프로세스들을 수반하는 서비스로써 병렬 프로그램에서 종종 야기되는 여러 가지 통신 형태들의 집합이다. 이러한 통신 서비스 중 멀티캐스트는 많은 병렬 애플리케이션에서 아주 빈번히 사용되는 서비스로써 하나의 송신 프로세서가 네트워크에 있는 여러 개의 다른 수

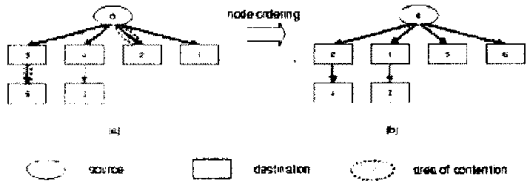
※ 본 연구는 '99학년도 덕성여자대학교 자연과학 연구소 연구비 지원으로 이루어졌음
† 정 회 원 : 덕성여자대학교 전산학과 교수
논문접수 : 1999년 7월 2일, 심사완료 : 1999년 9월 14일

된 프로세서에게 메시지를 보내는 형태를 갖는다.

소프트웨어 멀티캐스트 알고리즘을 설계하는데 있어서의 주된 문제는 성능과 이식성간의 교환조건(trade off)을 고려하는 것이다. 만약 근원적인 네트워크 구조를 고려하지 않은 멀티캐스트 알고리즘이라면, 실제 네트워크에 구현했을 경우 최적의 성능을 얻을 수 없을 것이다. 현재 가장 인기있는 여러 멀티캐스트 알고리즘(즉, U-mesh 알고리즘[8]과 U-min 알고리즘[11])에 기본으로 사용되는 이항 멀티캐스트 트리(binomial multicast tree)가 실제 총 실행시간과 오버헤드를 고려할 때 대부분의 시스템에서는 최적이 되지 못할 수 있다. 이러한 문제를 고려하여, 본 연구는 변수화 된 모델을 기본으로 멀티캐스트 문제를 접근한다. 각 시스템은 쉽게 측정이 가능한 주된 변수들에 의해 특성화된다. 이 변수화 된 모델에서 컴퓨터 구조에 의존되지 않는 즉, 이식성 있고 우수한 성능을 보이는 OPT-tree라는 멀티캐스트 알고리즘을 제안한다. 또한 OPT-tree가 최적의 멀티캐스트 트리를 만들어 냄을 증명한다. 최적의 멀티캐스트란 하나의 소스(source)가 여러 개의 도착지(destination)에게 메시지를 전송할 때 가장 빠른 시간내에 메시지 전송을 수행하는 스케쥴링 방법을 의미한다. 웜홀(wormhole) 스위칭을 사용하고 풍부한 상호연결위상을 갖는 네트워크 모델에서는 제안된 멀티캐스트 알고리즘은 거의 최적으로 수행한다.

그러나, 진정한 최적의 수행을 달성하기 위해서는 네트워크 위상과 스위칭 기법과 같은 구조에 의존적인 시스템의 특성들을 고려할 필요가 있다. 대부분의 실제 네트워크에서 메시지 전송이 적절하게 스케쥴되지 않는다면 회선경쟁이 일어나기가 쉽다. 이런 경우 실제 멀티캐스트 시간은 회선경쟁으로 인해, 변수화 된 모델에서 추정된 멀티캐스트 시간보다 더 길 수 있다. 메쉬 네트워크에 대한 U-mesh 알고리즘과 2차원의 멀티스테이지(multistage)상호 연결 네트워크에 대한 U-min 알고리즘의 예와 같이, 다양한 네트워크 위상에 대해 회선경쟁이 없는 멀티캐스트 트리를 구성하기 위한 연구들이 행해졌다. 그러나 두 가지 알고리즘 모두 이항트리를 기본으로 하며 제한된 변수의 네트워크에서만 효율적이다.

OPT-tree 알고리즘으로 생성된 트리에서는 노드들의 배열, 즉 트리의 각 노드의 위치는 고려하지 않는데 네트워크 위상과 스위칭 기법에 따라 네트워크 회선경쟁을 피하기 위해서는 노드들의 배열도 달라질 수 있다. 이러한 [illegible]의 예로 설명될 수 있다. (그림 1)의 멀티캐스트 트리가 네트워크 A에서 수행될 때 최적이라고 하자. 하지만, 노드 0이 노드 2에, 노드 5가 노드 6에 메시지를 보낼 때 생기는 회선경쟁으로 인해 최적의 실행을 달성하지 못하게 된다. (그림 1)(b)와 같이 네트워크 특성에 기반한 멀티캐스트 트리의 노드들을 재배치한다면 회선경쟁을 피할 수 있다. 본 논문에서는 OPT-tree로 생성된 트리에서 네트워크 위상에 따른 튜닝(tuning) 즉, 노드의 배열 문제를 다룬다. 메쉬 네트워크의 경우, 어떻게 노드들을 배열하여 최적의 수행을 얻는지를 보인다.



(그림 1) (a) 회선경쟁 가능 (b) 회선경쟁 없음

이 논문의 주된 공헌은 첫째, 각 병렬 컴퓨터를 어떤 임계의 기계특성 변수로 특성화 하는 것이다. 멀티캐스트 통신 라이브러리의 구현은 라이브러리의 컴파일 상태 동안에 고려되는 시스템의 변수에 기반한다. 따라서 제안한 OPT-tree멀티캐스트 알고리즘은 이식성이 있고 이들 핵심 시스템 변수를 참작하므로 결론적으로 달성할 수 있는 성능은 거의 최상에 가까울 것이다. 둘째, 변수화된 멀티캐스트 알고리즘이 최상의 성능을 달성할 수 있도록 컴퓨터 구조에 의존한 튜닝에 대한 연구이다. 멀티캐스트 트리에서 회선경쟁을 피하기 위해 노드를 어떻게 배열할 것인지를 설명한다.

다음 장에서는 변수화된 통신 모델과 구조에 의존적이지 않는 이식성있는 멀티캐스트 트리의 구성에 대해 설명하고 3장에서는 웜홀 스위칭을 사용한 메쉬 네트워크에서 변수화된 멀티캐스트 알고리즘의 회선충돌없는 구현을 보여준다. 4장에서는 본 논문에서 제안한 변수화된 멀티캐스트 트리와 U-mesh 이항 트리에 대한 실험적인 결과를 비교한다. 마지막으로 5장에서 결론을 맺는다.

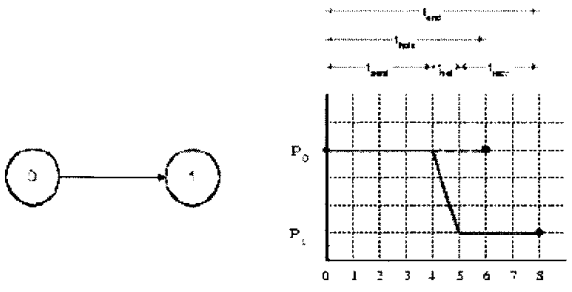
2. 컴퓨터구조에 독립적인 멀티캐스트

2.1 변수화된 통신 모델

여러 병렬처리 컴퓨터들은 서로 다른 처리 능력, 네

네트워크 상호연결 구조, 호스트 인터페이스, 운영체제등을 갖는다. 소프트웨어 통신시간을 감소시키기 위해 다양한 기술들이 사용되었다[4].

추상적인 병렬구조모델은 그 중요한 기능을 특성화하기에 충분한 수의 매개변수를 가지고 있어야 한다. 그러나 너무 많은 변수는 기계에 독립적인 소프트웨어의 디자인을 복잡하게 한다. 본 논문에서 제안하는 변수화된 통신 모델은 LogP 모델[3]의 확장으로 몇 개의 주된 특성변수로 구성된다. (그림 2)에서 보여진 sending latency(t_{send}), receiving latency(t_{recv}), network latency (t_{net}), holding latency(t_{hold}), end-to-end latency(t_{end})의 5개의 변수를 포함하는 모델이다. 변수 t_{send}는 데이터를 패킷으로 분할하는데 소요되는 시간과 체크섬(checksum) 계산과 메시지 복사하는 시간들을 포함한 송신프로세서(source)에서 메시지를 보내는 과정에서 필요한 시간이다. 이와 유사하게, t_{recv}는 수신프로세서(destination)에서 메시지를 받는 과정에서 필요한 시간이다. t_{net}는 네트워크를 통해 메시지를 전달하는데 필요한 시간이다. 네트워크 시간을 정확히 측정하기 위해서는 관련이 없는 다른 메시지에 의한 네트워크 회선경쟁의 영향을 피하도록 조절된 환경에서 벤치마킹(benchmarking)이 이루어져야 한다.

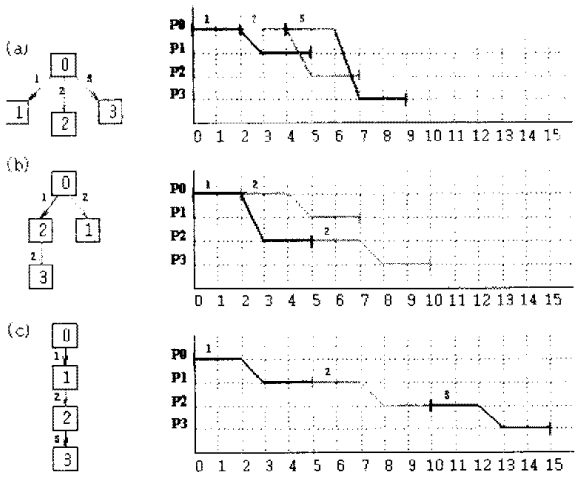


(그림 2) 점대점 통신 모델

t_{send}, t_{net}, t_{recv} 변수들의 측정은 다소 어렵다. 정확한 측정을 위해서는 하드웨어 모니터와 소프트웨어 프로브(probe)의 사용과 같은 특별한 기술이 필요하다. 따라서 t_{end}와 t_{hold} 변수를 추가적으로 정의한다. t_{end}는 송신기가 메시지를 보내기 시작한 시간부터 수신기가 수신을 끝내는 시간 사이의 간격이다. 따라서 $t_{end} = t_{send} + t_{net} + t_{recv}$이 된다. t_{hold}는 연속적인 송신 혹은 연속적인 수신을 할 때 필요한 최소한의 시간 간격이다. 대부분의 경우, 통신 서비스가 송신과 수신 동작에 기초하므로 t_{hold}와 t_{end}를 기본으로 통신 수행을 예측할 수 있다. t_{hold}와 t_{end}는 측정이 용이하며 사용자 레벨에서 수행이 가능하다[10]. (그림 2)에서는 $t_{send}=4$, $t_{net}=1$, $t_{recv}=3$, $t_{hold}=6$ 일 때 송신 프로세서 P_0가 수신 프로세서 P_1에게 메시지를 보낼 때의 타이밍도식이다. P_0에서 송신을 위한 4 units 경과 후, 1 unit동안 실제 네트워크를 통해 메시지가 전달되며 마지막 3 units동안 P_1에서 수신을 위한 시간이 경과된다.

대부분의 멀티캐스트 통신은 그룹 내에 프로세서들이 트리 모양의 구조로 통신순서가 형성된다. 멀티캐스트 트리구조에서 각각의 프로세서들은 메시지를 자식 프로세서에게 보내기 위해 점대점(point-to-point) 통신 서비스를 사용한다. 대부분의 통신 시스템에서는 단일포트(one-port) 통신구조로 멀티캐스트 단계라 부르는 한 일정시간 동안에 각 프로세서는 하나의 수신 프로세서에게만 메시지를 보낼 수 있다.

(그림 3)은 $t_{send}=2$, $t_{net}=1$, $t_{recv}=2$, $t_{hold}=2$일 때 3개의 다른 멀티캐스트 트리와 그에 대응하는 타이밍 도식을 보여준다. 첫 번째 트리는 순차 트리(sequential tree) 또는 분리 어드레싱(separate-addressing)으로 알려진 것으로 루트 노드(P_0)는 3개의 멀티캐스트 단계 동안 각각 3개의 프로세서에게 메시지를 보낸다. 즉, 송신프로세서 P_0가 P_1에게 메시지를 보내는데, t_{hold} 시간이 지나면, P_1에서 메시지 수신이 끝나지 않았더라도 P_0는 P_2에게 메시지를 보낸다. 마찬가지로, P_1과 P_2에서 아직 수신이 끝나지 않아도, 또 다른 t_{hold} 시간이 지나면 P_0는 P_3에게 메시지를 보낸다. (그림 3)(b)는 반복적인 더블링(doubling) 기술에 기초한 이항 트리이다. 더블링 기술이란 송신자가 메시지를 어떤 수신자에게 전달한 후, 다음 단계에서는 원래의 송신자뿐 아니라 메시지를 전달받은 수신자도 함께 다른 수신자에게 각각 메시지를 전달하는 방법이다. 즉, 이미 메시지를 받은 프로세서의 수는 각 단계 후 두 배로 증가한다. (그림 3)(b)에서 루트 노드(P_0)는 첫 단계에서 자식노드(P_2)에게 메시지를 보낸다. 두 번째 단계에서는 P_0와 P_2이 동시에 프로세서 P_1와 P_3에 각각 메시지를 보낸다. 마지막 예는 체인(chain) 트리이다. (그림 3)(c)에서와 같이 루트노드(P_0)는 첫 단계에서 자식노드 P_1에게 메시지를 보내고, 두 번째 단계에서 P_1은 메시지를 체인 내의 자신의 다음 노드 P_2에 전달한다. 체인 내의 마지막 프로세서 P_3에 메시지가 완전히 전달되면 멀티캐스트가 끝난다.



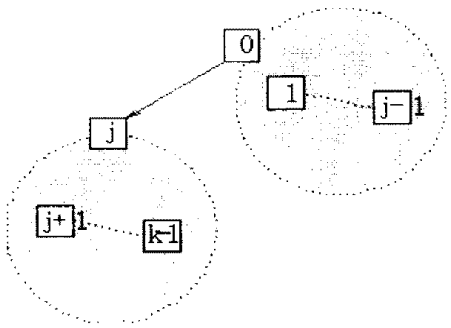
(그림 3) (a)순차트리 (b)이항트리 (c)체인트리

그룹 크기가 k인 경우, 순차적 트리와 체인 트리는 $k-1$개의 멀티캐스트 단계가 필요하고 이항 트리는 $\lceil \log_2 k \rceil$ 멀티캐스트 단계가 필요하다. 요구되는 멀티캐스트 단계의 수로 보면 이항 트리가 최선이나 실제적으로 각 멀티캐스트 트리는 네트워크의 소프트웨어 대기시간에 의존한다. (그림 3)의 예에서 $t_{send}=2$, $t_{net}=1$, $t_{recv}=2$, $t_{hold}=t_{send}$ 라고 가정할 때 송신과 수신 대기시간을 고려하면 순차적 트리가 최상의 수행시간을 갖는다.

2.2 구조에 독립적인 최적의 멀티캐스트 트리

2.1절에서 언급한 바와 같이, 이론적으로 최적의 이항 트리가 실제적으로는 최적이 아닐 수 있다. 여기서 우리는, 주어진 메시지 크기에 대해 두 개의 중요한 네트워크 변수 t_{hold}와 t_{end}를 고려하여 최적의 멀티캐스트 트리를 개발할 수 있을까? 하는 문제에 접하게 된다. 이 절에서는 이러한 문제를 다룬다.

P_0가 루트노드인 k개의 node (P_0, P_1,....,P_{k-1})를 갖는 최적의 멀티캐스트 트리를 구성해보자. 첫 단계에서 P_0는 P_j에 메시지를 보낸다. P_j가 메시지를 받아서 트리내의 다른 노드에게 보낼 수 있게 되기 전에, P_0는 t_{hold}와 t_{end} 값에 의존하여 트리의 다른 노드에게 같은 메시지를 보낼 수 있다. P_j가 메시지를 송신할 준비가 되었을 때 기본적으로 두 개의 멀티캐스트 서브트리로 나누어 생각한다. (그림 4)에서 보는 것과 같이 하나는 P_0에 루트를 둔 j개의 노드 트리(P_0, P_1,....,P_{j-1})이고 다른 하나는 P_j에 루트를 둔 ($k-j$)개의 노드 트리(P_j, P_{j+1},...., P_{k-1})이다. 문제는 이제 어떤 노드가 어떤 서브트리에 속하는가에 있다.



(그림 4) k-노드의 최적 트리

최적의 멀티캐스트 트리를 구성하기 위해서는 두 가지 조건이 만족되어야 한다. 첫째, P_j 노드를 선택할 때 형성된 멀티캐스트 트리가 최적이 되도록 해야 한다. 두 번째, 두 개의 서브트리 (P_0, P_1,....,P_{j-1}) 와 (P_j, P_{j+1},...., P_{k-1})이 각각 자체 내에서 최적이어야 한다. 이와 같은 과정이 각 멀티캐스트 서브트리에 재귀적으로 계속 적용된다. 따라서 최적의 해를 찾기 위해 이 두 가지 특성에 동적 프로그래밍(dynamic programming) 기술 [2]을 이용한다.

P_a를 루트노드로 하는 i개의 노드들 P_a, P_{a+1},...,P_{a+i-1} (단, $0 \le a \le k-1$) 사이에 메시지를 멀티캐스트 하는데 필요한 최소시간을 $t[i]$ (단, $1 \le i \le k$) 라고 두자. 그러면 반복적으로 $t[i]$를 다음과 같이 정의할 수 있다. i가 1이면 트리에 단 하나의 노드가 존재하므로 $t[i]=0$이 된다. $i>1$일 때 P_a는 메시지를 P_{a+j} 노드에 (j의 값은 $1 \le j \le i-1$) 보낸다. t_{hold} 시간이 경과하면 P_a는 그것의 서브트리(P_a, P_{a+1},...,P_{a+j-1})안의 노드에 계속하여 메시지를 전송하고, t_{end} 시간이 경과하면 P_{a+j}가 메시지를 받아 그것의 서브트리 (P_{a+j}, P_{a+j+1},...,P_{a+i-1})안의 노드에 메시지 전달을 시작할 수 있다. 따라서 i개의 노드들 사이의 멀티캐스트시간 $t[i]$는 서브트리 (P_a, P_{a+1},..., P_{a+j-1})의 멀티캐스트시간에 t_{hold}를 합한 값과 서브트리 (P_{a+j}, P_{a+j+1},...,P_{a+i-1})의 멀티캐스트 시간에 t_{end}를 합한 값 중 큰 값이다. P_a가 서브트리 (P_a, P_{a+1},...,P_{a+j-1})의 노드들에게 메시지를 보내기 전에 P_{a+j}에게 먼저 보내야 하기 때문에 t_{hold}시간이 필요하고, P_{a+j}가 서브트리 (P_{a+j}, P_{a+j+1},...,P_{a+i-1})의 노드들에게 메시지를 멀티캐스트 하기 전에 P_a로부터 메시지를 받아야 하므로 t_{end}시간이 필요하다는 점을 주목하라. 최적의 값을 얻기 위해서 멀티캐스트시간이 최소가 되도록 P_{a+j} 노드를 선

택해야 한다. 따라서 $t[i]$에 대해 다음과 같은 순환을 얻는다.

$$t[i]=\begin{cases} 0 & \text{if } i=1 \\ \min_{1\le j<i}\{\max(t[j]+t_{hold},\ t[i-j]+t_{end})\} & \text{if } i>1 \end{cases}$$

k 노드 트리의 최적의 멀티캐스트시간은 $t[k]$이고 동적 프로그래밍기법을 이용하여 $O(k^2)$시간에 얻어질 수 있다. 특히 $t[1]$, $t[2]$,..... $t[k]$ 순으로 값을 구하여 배열에 저장할 수 있다. $t[i]$를 (단, $1<i\le k$) 구하기 위해 각 j 값을 1에서 $i-1$까지 고려하여 $\max(t[j]+t_{hold},\ t[i-j]+t_{end})$가 최소가 되는 j값을 선택한다. 따라서 전체 실행시간은 $O(k^2)$이다.

실행시간을 $O(k)$까지 향상시키기 위해 각 단계에서 고려해야 할 j값들의 선택을 제한한다. j값의 제한은 다음 정리에서 설명될 $t[i]$의 특성에 기반한다.

[정리 2.1] j_i 를 $t[i]$가 최소가 되는 j 값이라고 정의하자. (즉, i-노드 트리를 분할하는 가장 좋은 방법이 각각 j_i 개와 $i-j_i$ 개의 노드를 갖는 서브트리들로 나누는 것이다.) 그러면, j_{i+1}은 $t[i+1]$이 최소가 되는 j 값으로 i 가 1일 때 $j_{i+1} = j_2 = 1$이고, i 가 $2\le i \le k-1$범위일 때 $j_{i+1} = j_i + 1$ 또는 $j_{i+1} = j_i$ 이다.

위 정리에 대한 증명은 [7]을 참조하라. 정리2.1을 기초로 하여 전체 실행시간을 줄이기 위해 알고리즘을 다음과 같이 수정할 수 있다.

$$t[i]=\begin{cases} 0 & \text{if } i=1 \\ t_{end} & \text{if } i=2 \\ \min\{\max(t[j_{i-1}+1]+t_{hold},\ t[i-1-j_{i-1}]+t_{end}),\ \max(t[j_i-1]+t_{hold},\ t[i-j_{i-1}]+t_{end})\} & \text{if } i>3 \end{cases}$$

단, 여기서 $j_2 = 1$이다.

동적 프로그래밍을 이용한 수정된 알고리즘을 *OPT -tree*라 부르고, 알고리즘 1에 서술되어 있다. <표 1>은 노드 수가 9개이고 t_{hold}=20, t_{end}=55일 때 OPT -tree에 의해 연산된 각각의 j_i 값($1\le i\le 9$)과 멀티캐스트 시간을 나타내고 이에 대응되는 최적의 멀티캐스트 트리는 (그림 5)에서 보여진다. 이 OPT-tree 알고리즘으로부터 다음과 같은 정리를 얻는다.

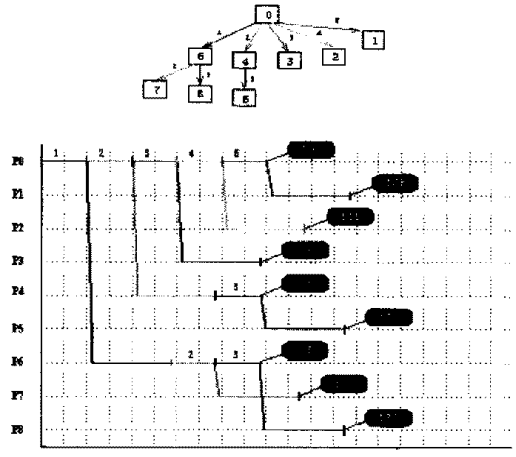
[정리 2.2] k-노드 트리의 최적 멀티캐스트 실행시간은 $O(k)$시간 안에서 계산될 수 있다.

〈알고리즘 1〉 OPT-tree 알고리즘

Input
t_{hold} : 연속적인 송신/수신시 필요한 시간간격
t_{end} : 송신시작부터 수신완료 사이의 시간간격
k : 멀티캐스트 노드 수 (1개의 source 포함)
Output
j_i : i개 노드를 가진 트리에서 source를 포함한 서브 트리의 노드 수 ($1\le i \le k$)
$t[i]$: i개 노드를 멀티캐스트하는 시간
Procedure
$t[1] = 0$; $t[2] = t_{end}$;
for $i = 3$ **to** k
 if $\max\{t[j_{i-1}]+t_{hold},\ t[i-j_{i-1}]+t_{end}\} < \max\{t[j_{i-1}+1]+t_{hold},\ t[i-1-j_{i-1}]+t_{end}\}$
 then $t[i] = \max\{t[j_{i-1}]+t_{hold},\ t[i-j_{i-1}]+t_{end}\}$;
 $j_i = j_{i-1}$;
 else $t[i] = \max\{t[j_{i-1}+1]+t_{hold},\ t[i-1-j_{i-1}]+t_{end}\}$;
 $j_i = j_{i-1}+1$;
 endif
endfor

〈표 1〉 k= 9, thold=20, and tend=55

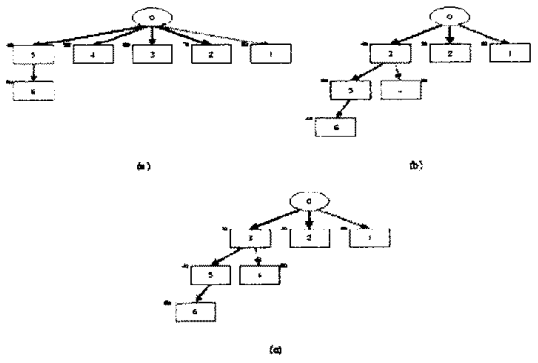
i	j_i	$i-j_i$	$t[j_i]+t_{hold}$	$t[i-j_i]+t_{end}$	$t[i]$
1	-	-	-	-	0
2	1	1	20	55	55
3	2	1	75	55	75
4	3	1	95	55	95
5	3	2	95	110	110
6	4	2	115	110	115
7	5	2	130	110	130
8	5	3	130	130	130
9	6	3	135	130	135



(그림 5) 9-노드의 최적 멀티캐스트 트리

(그림 6)(a)는 OPT-tree 알고리즘을 t_{hold}=10 과 t_{end}=40인 네트워크에 구현할 때 7개의 노드들의 최적 멀티캐

스트 트리가 어떻게 형성되는 지를 보여준다. 트리의 모양은 (그림 6)(b)에 있는 7개의 노드들의 이항트리[2, 12]와 다르다. 이항트리란 순서트리로서 재귀적으로 다음과 같이 정의된다. 이항트리 B_0는 하나의 노드로 이루어진 트리이고, 이항트리 B_k는 두 개의 이항트리 B_{k-1}로 구성되는데 이들 중 어느 하나의 이항트리 B_{k-1}의 루트가 다른 이항트리 B_{k-1}의 루트의 가장 왼쪽 자식이 되도록 연결되어진 트리이다. 같은 네트워크에서, 우리가 구한 최적 트리의 멀티캐스트 시간(즉, 80)은 이항트리의 멀티캐스트 시간(즉, 120)보다 훨씬 좋다. (그림 6)(c)는 t_{hold}=20 과 t_{end}=20을 갖는 네트워크에 구현될 때 최적 멀티캐스트 트리를 나타낸다. (그림 6)(a)와 (그림 6)(c)의 트리들의 노드 수는 같지만 네트워크 변수에 의해 그들의 모양이 서로 다르다. 한 가지 주목할 것은 $t_{hold}=t_{end}$ 일 때, 최적 멀티캐스트 트리는 이항 트리가 된다는 것을 (그림 6)(c)에서 알 수 있다.



(그림 6) 메쉬에서 두 멀티캐스트 트리 비교

지금까지 일반적인 1-port 통신구조에 바탕을 두고 있으나 TMC CM-5와 Intel/CMU iWARP과 같은 병렬기계에서는 각각의 프로세서가 다중 통신 포트를 가지고 있다. 본 논문에서 제시한 OPT-tree를 다중포트 통신 모델에 근간을 둔 최적의 멀티캐스트 트리로 일반화할 수 있다[7].

주어진 t_{hold}와 t_{end}에 대해, OPT-tree 알고리즘은 구조에 독립적인 최적의 멀티캐스트 트리를 구성한다는 것이 증명되었다. 이 최적성은 트리노드의 순서에 관계없이 주어진 메시지 크기에 대해 t_{hold}와 t_{end}가 일정하게 유지된다는 가정하에 가능하다. 그러나 웜홀 스위치 네트워크에서 통신 시간은 거리에 거의 무관하다는 특성이 있지만, 회선경쟁이 발생했을 때, t_{end}를 지연시킬 수 있다는 것이다. 이로 인해 트리의 멀티캐스트 [illegible] 다음을 증가시켜 트리가 최적화 되지 못할 수 있다. 그러므로, 회선경쟁 피하는 좋은 방법으로 공통의 채널이 두개의 다른 송신측에 의해서 사용되지 않게 송신노드를 적절한 순서로 정렬하는 것이다. OPT-tree 알고리즘으로 얻어진 최적의 멀티캐스트 트리에서 각 노드들을 컴퓨터 구조에 따라 어떻게 순서를 정하느냐 하는 문제를 메쉬 구조의 시스템을 예로 다음 장에서 설명한다.

3. 메쉬 네트워크에서의 최적 멀티캐스트

각 차원(dimension)에 m개의 노드를 가진 n차원 메쉬에서 노드 x의 주소는 $\sigma_{n-1}(x)\sigma_{n-2}(x)\cdots\sigma_0(x)$로 표시한다. (여기서, $\sigma_i(x) \in \{0, 1, ..., m-1\}$이다.) 차원순서 라우팅(dimension-ordered routing)을 적용한 노드 a에서 노드 b까지의 경로는 $\rho(a,b) = (a; x_1, x_2, \cdots, x_k ; b)$로 표시하는데, x_i들은 라우팅 경로를 구성하는 중간 라우터들이다. 이진 관계인 차원순서관계(dimension ordered relation)는 $<_d$로 표기하고, 두 개의 노드 a와 b 사이에서 다음과 같이 정의된다. a와 b의 주소가 같거나 $j+1 \le i \le n-1$인 모든 i에 대하여 $\sigma_i(a)=\sigma_i(b)$ 이고 $\sigma_j(a)<\sigma_j(b)$인 j가 존재한다면, $a<_d b$ 관계가 성립한다. 노드연속 $\{x_1, x_2, \cdots, x_m\}$이 각 원소가 서로 다르고 차원순서로 되어있다면 (즉, $1 \le i<j \le m$인 모든 i, j에 대해서 $x_i <_d x_j$이라면) *차원순서체인(dimension-ordered chain)*이라고 부른다.

웜홀 라우팅 메쉬 네트워크에서 회선경쟁이 없는 멀티캐스트 트리를 구성하는 문제는 [8]에서 연구되어졌다. 차원순서체인과 재귀적인 더블링(doubling)기술을 기본으로 한 U-mesh 알고리즘은 메쉬 네트워크에 대해 회선경쟁이 없는 효율적인 멀티캐스트 트리를 구성할 수 있다. 그러나, U-mesh 트리는 이항 트리로써 $t_{hold}=t_{end}$인 네트워크에서만 최적이 된다. 이 장에서는 메쉬 네트워크에서 변수화된 멀티캐스트 트리를 최소 시간에 구현하는 OPT-Mesh라고 불리는 네트워크 구조에 튜닝된 새로운 알고리즘을 제안한다. 즉, 2장에서의 OPT-tree로 구한 최적의 멀티캐스트 트리에서 회선경쟁이 없도록 적절히 노드순서를 정하는 알고리즘이다. 차원순서체인과 변수화된 멀티캐스트 트리를 기본으로 한 OPT-Mesh 트리는 네트워크 변수에 따라 회선경쟁이 없는 최적의 멀티캐스트 트리를 구성한다.

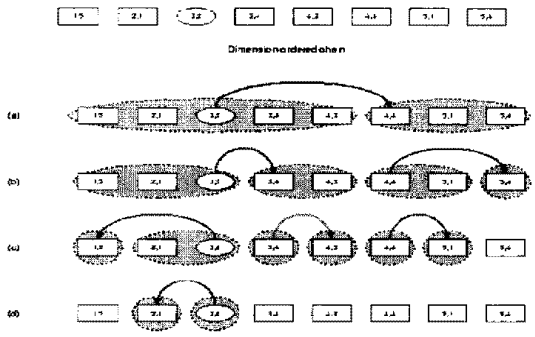
[정리 3.1] [8] $u<_d v<_d x<_d y$ 라면, (i) $\rho(u,v)$와 $\rho(x,y)$에 있는 각각의 아크(arc)들은 서로 공통되는 것이 없다. (ii) $\rho(y,x)$와 $\rho(v,u)$에 있는 각각의 아크들은 서로 공통되는 것이 없다. 그리고 (iii) $\rho(v,u)$와 $\rho(x,y)$에 있는 각각의 아크들은 서로 공통되는 것이 없다.

j_i를 알고리즘 1에서 보여준 OPT 트리 알고리즘에 의해서 연산된 결과 값으로 하자. (i 노드를 가지는 변수화된 멀티캐스트 트리는 두 개의 서브 트리를 가지는데, 그 중 하나는 루트를 포함하는 서브트리로써 노드 개수가 j_i 이고 다른 서브트리는 노드 개수가 $i-j_i$ 이다.) 알고리즘 2에 주어진 OPT-mesh 알고리즘에서는 소스와 도착지 주소들이 차원순서체인 Φ에 정렬된다. 소스는 노드 개수가 i인 Φ를 연속적으로 두 부분으로 나눈다. 만약 소스 노드가 하위노드그룹(상위노드그룹)에 있다면, 상위노드그룹(하위노드그룹) 중 가장 낮은(높은) 순위($<_d$에 의한 순위)의 노드에게 메시지를 보낸다. 이렇게 메시지를 받은 노드는 같은 OPT-mesh 알고리즘을 사용하여 상위노드그룹(하위노드그룹) 내의 다른 노드들에게 메시지를 전달한다. 각 메시지는 데이터뿐 아니라 메시지를 받을 노드가 메시지를 받은 후 그것을 전달해야 할 도착지의 주소도 가지고 있다. 각 멀티캐스트 단계에서 소스는 메시지를 받는 노드와 소스가 포함되지 않는 다른 노드그룹의 노드들을 리스트 Φ로부터 지운다. 리스트 Φ에 소스 자신의 주소만 남을 때까지 이러한 과정을 계속한다.

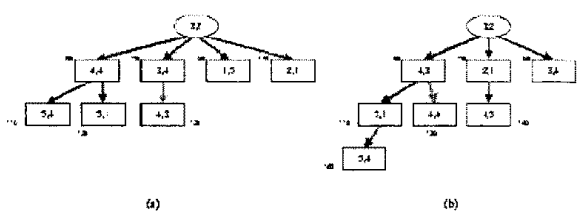
〈알고리즘 2〉 OPT-mesh 알고리즘

```
Input: Φ: 차원순서체인 (x0, x1, ···, xd)
  xs : source 노드의 주소
  ji : source 포함한 서브트리의 크기 (i-r 1)
Procedure
While l < r do
  if s < l + ji
  then rec ← l + ji ;
       D ← {xrec, xrec+1, ···, xr};
       r ← rec − 1;
  else rec ← r − ji ;
       D ← {xl, xl+1, ···, xrec};
       l ← rec + 1;
  endif
  send a message to node xrec with the address field D;
endwhile
```

OPT-mesh를 사용한 멀티캐스트 구현은 (그림 7)에서 보여지는데, $t_{hold}=20$이고 $t_{end}=55$인 2차원 메쉬 6×6에서 7개의 도착지 노드를 가지는 멀티캐스트이다. (OPT-tree 알고리즘에 의해서 연산된 각각의 값 j_i 는 테이블 1을 참고한다.) 소스노드는 (3,2)이고 7개의 도착지 노드는 {(1,5), (2,1), (3,4), (4,3), (4,4), (5,1), (5,4)}이다. 처음에 8개의 노드들이 차원순서체인으로 정렬된다. 소스 (3,2)는 제일 먼저 Φ에서 상위 8-j_8노드 그룹 중 가장 낮은 순서의 주소를 가진 노드인 (4,4)에 메시지를 보낸다. (이때 테이블 1에서 $j_8=5$이므로 5개 노드를 가진 그룹과 3개 노드를 가진 그룹으로 나누어진다.) 상위 3개의 노드들은 Φ에서 삭제되고 Φ에 남아 있는 노드들은 {(1,5), (2,1), (3,2), (3,4), (4,3)}이 된다. 그리고 소스 (3,2)는 Φ의 상위 5-j_5 노드들 중 가장 낮은 주소를 가진 노드인 (3,4)에 메시지를 보낸다. ($j_5=3$이다.) 이때 메시지를 받은 각 노드들은 자신의 그룹(즉, 서브트리)에 있는 노드들에게 전달받은 메시지를 위의 알고리즘을 사용하여 전송한다. OPT-mesh 알고리즘으로 구한 최적의 멀티캐스트 트리는 (그림 8)(a)이며 멀티캐스트 시간은 130이다. (그림 8)(b)는 이항트리에 기초한 U-mesh 알고리즘[8]에 의한 트리로써 멀티캐스트 시간은 165이다. OPT-mesh 알고리즘은 회선경쟁이 없는 경로들을 사용하며 이 알고리즘에 의해서 얻어진 유니캐스트(unicast)경로들은 (그림 9)에서 보여준다.



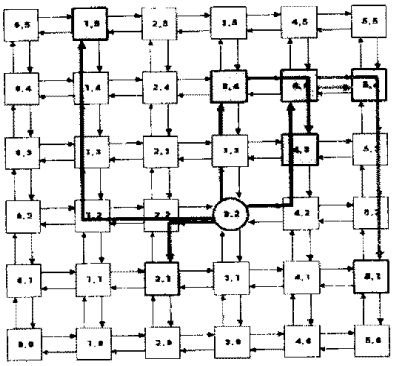
(그림 7) OPT-mesh 알고리즘을 사용한 예



(그림 8) 메쉬에서 두 멀티캐스트 트리 비교

(그림 9)를 보면 ((4,4), (5,4)) 또는 ((3,2), (2,2))에서 회선충돌이 존재하는 것 같이 보인다. 경로 ρ((4,4), (5,4))

와 경로 ρ((4,4), (5,4))를 공통의 회선 ((4,4), (5,4))를 지고 있다. 그러나, 경로 ρ((4,4), (5,4))에서 사용되는 회선 ((4,4), (5,4))는 t_0+t_{hold} 시간 (즉, 75)부터 이용 가능할 것이다. 여기서, t_0는 노드(4,4)가 노드(5,4)에 메시지를 보내기 시작할 때의 시간을 나타낸다. 따라서 이 회선은 경로 ρ((4,4), (5,1))에 회선경쟁 없이 사용될 수 있다. 회선 ((3,2), (2,2))경우도 이와 유사하게 설명된다. 다음 정리는 차원순서 라우트를 사용하는 OPT-mesh 알고리즘에는 어떠한 회선경쟁도 없다는 것을 증명한다.



(그림 9) 메쉬에서의 멀티캐스트

[정리 3.2] OPT-mesh 알고리즘을 사용한 메쉬에서의 변수화된 멀티캐스트 트리의 구현은 최적이다.

증명 : u, v, x, y는 4개의 서로 다른 노드들이라 하자. OPT mesh 알고리즘에 의해서 사용되는 어떤 두 개의 경로 ρ(u,v)와 ρ(x,y)에 대해, u, v, x, y 사이에 가능한 순서는 모두 6개이다. 즉, $u<_d v<_d x<_d y$, $v<_d u<_d x<_d y$, $v<_d u<_d y<_d x$, $x<_d y<_d u<_d v$, $y<_d x<_d u<_d v$, $y<_d x<_d v<_d u$들이다. 각 경우, ρ(u,v)와 ρ(x,y)는 정리 3.1에 의해서 공통된 아크를 갖지 않는다.

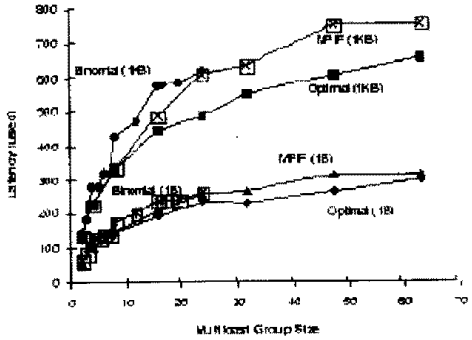
u, v, x, y 중 어떤 것이 다른 노드와 동일한 노드인 경우를 살펴보자. 만약 OPT-mesh 알고리즘에 의해 사용된 두 개의 경로 ρ(u,v)와 ρ(x,y)가 어떤 아크를 공유한다면, u와 x가 서로 동일한 노드인 경우이다. 그러므로 u가 t_0 시간에 v에게 메시지를 보내고, t_1 시간에 (단, $t_0< t_1$) y에게 메시지를 보낸다면, $t_1 \geq t_0+t_{hold}$ 이다. u에서의 output 포트는 v에게 메시지를 보낸 후 t_0+t_{hold}시간부터 이용 가능하게 되므로, 경로 ρ(u,v)안에 있는 아크들은 u에서 가까운 아크부터 시작하여 연속적으로 이용 가능하다. 그러므로 두 개의 경로 ρ(u,v)와 ρ(u,y)사이에 회선충돌이 존재하지 않는다. 이상으로 OPT-mesh 알고리즘이 최소시간에 멀티캐스트를 구현함이 증명된다.

4. 실험 및 성능평가

4.1 OPT-tree 알고리즘 성능 평가

128-노드 IBM/SP에서 실험을 수행하며 MPI-F library[5]를 사용하여 구현한다. 고성능 스위치를 충분히 활용하기 위해 옵션 us와 함께 라이브러리 equlib가 사용된다. 성능을 비교 및 분석하기 위하여 3가지 멀티캐스트 트리 즉, 가장 많이 쓰이는 이항트리, MPI-F에 쓰이는 블락을 기본으로 한 이항트리(block-based binomial tree), 본 논문에서 제안한 최적의 멀티캐스트 트리(OPT-tree) 등을 두 개의 다른 메시지 크기로 측정하여 결과를 분석한다. 순차적 트리와 체인 트리는 IBM/SP에서 열등하게 수행됨을 이미 알고 있으므로 이 실험에서는 고려하지 않는다.

(그림 10)은 메시지 크기를 1바이트와 1킬로바이트로 실험했을 때의 3개의 서로 다른 멀티캐스트 트리의 멀티캐스트 시간을 보여준다. 3개의 트리 모두가 멀티캐스트 그룹 크기(프로세서의 수)가 증가할수록 멀티캐스트 시간은 증가하며, 멀티캐스트 그룹 크기가 작은 경우에는 3개의 트리가 서로 거의 비슷한 성능을 보인다. 본 논문에서 제안한 최적의 멀티캐스트 트리(OPT-tree)가 어떠한 경우에도 가장 우수한 성능을 보인다. 메시지 크기가 1바이트일 때 멀티캐스트 시간을 비교해보면 OPT-tree가 MPI-F 트리보다 작은 차이로 우월하지만, 메시지 크기가 1킬로바이트일 때는 현저한 차이를 보이며 우월하다.



(그림 10) 3개의 멀티캐스트트리 시간비교

4.2 OPT-mesh 알고리즘 성능 평가

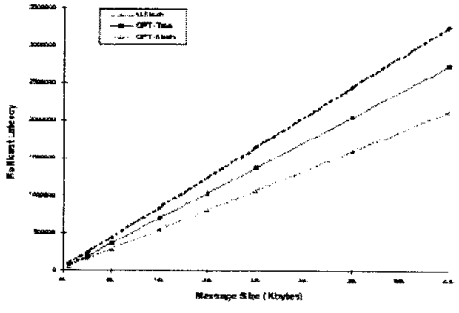
제안한 OPT-mesh 알고리즘을 평가하기 위해 윔홀-

스위치 메쉬를 위한 flit-level 시뮬레이터를 구현한다. 메쉬 네트워크는 X Y라우팅을 지원하고 단일포트인 16×16 메쉬를 기본으로 한다. 시뮬레이터는 송신준비시간, 송신지연시간, 채널지연시간, 수신준비시간, 수신지연시간등의 타이밍 변수들을 고려한다. 송신준비시간(s_s)은 메시지 복사, 체크섬 계산 등과 같이 메시지송신 과정에서 필요한 시간으로 메시지가 커질수록 길어진다. 송신지연시간(s_d)은 송신노드가 메시지의 여분의 플릿(flit)을 처리하는데 필요한 시간이다. 채널지연시간(c_d)은 채널로 플릿을 전송할 때 채널에서 초래되는 시간으로 네트워크가 제공할 수 있는 최고 대역폭과 대응한다. 이것은 노드에서 라우터(스위치), 혹은 두 개의 라우터(스위치)간에 플릿이 채널위로 전송될 때 적용되는 시간이다. 수신준비시간(r_s)과 수신지연시간(r_d)은 수신노드에서 메시지를 받을 때 생기는 시간으로 송신준비시간과 송신지연시간과 비슷하다. m 플릿의 메시지를 k 채널(회선)만큼 떨어진 노드에 보낼 때 회선경쟁이 없을 경우, $t_{send}=s_s+m\times s_d$, $t_{net}=k\times c_d+(m-1)\times c_d$, $t_{recv}=r_s+m\times r_d$이고, t_{hold}는 연속적인 송신 혹은 수신을 위한 최소한의 시간 간격으로 t_{send}의 값과 같다. 또한, $t_{end}=s_s+r_s+(k-1)\times c_d+m=3\times(s_d+c_d+r_d)$이다.

프로세서의 위치에 따라 회선경쟁의 가능성이 변화하므로, 무작위로 프로세서의 위치를 다르게 하여 같은 input 변수로 16번의 독립적인 실험을 한다. 결과로 나타나는 각 데이터들은 이들의 평균치이다.

16×16 메쉬에 세 가지 다른 알고리즘인 OPT-mesh, OPT-tree, U-mesh를 $s_s=2000$, $s_d=2$, $c_d=2$, $r_s=3500$, $r_d=3$의 값으로 평가한다. 이 값들은 $t_{hold}=20+0.02\times m$, $t_{end}=55+0.07\times m$인 IBM/SP 컴퓨터에 근거로 한다.

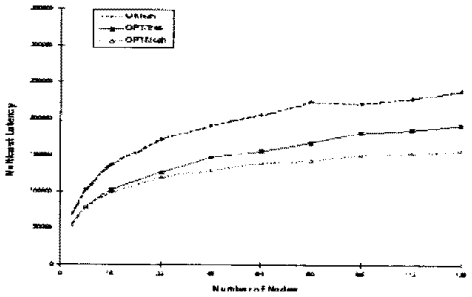
(그림 11)은 32개 노드의 멀티캐스트 트리인 경우,



(그림 11) 32-노드 멀티캐스트 트리비교

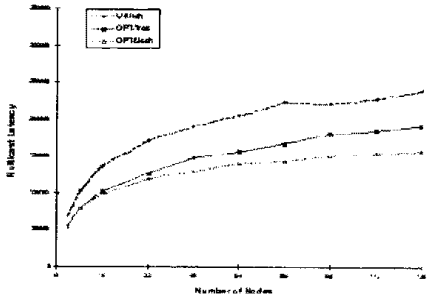
3개의 각 알고리즘에 의한 멀티캐스트 시간을 측정한 결과이다. OPT-mesh가 가장 우수한 성능을 보이는 반면 U-mesh는 가장 열등하다. 이와 비슷하게, 128개 노드의 멀티캐스트 트리일 때의 실험이 (그림 12)에 나타난다. 결과는 32개 노드의 경우와 비슷하다.

두 실험 모두에서 3개의 각 알고리즘에 의한 멀티캐스트 시간은 메시지 크기에 대해 선형임을 보인다. 여기서 멀티캐스트 시간은 각 알고리즘에 의해 형성되는 멀티캐스트 트리의 깊이(depth), 회선경쟁, 점대점 통신시간과 관계 있다는 것을 알 수 있다. 멀티캐스트 노드 수가 일정한 경우 U-mesh는 네트워크 변수에 관계없이 언제나 일정한 같은 깊이의 이항트리를 생성한다. U-mesh는 회선경쟁이 없는 알고리즘이므로 멀티캐스트 시간은 메시지 크기에 좌우되는 점대점 통신시간과 깊은 관계가 있다. t_{hold}와 t_{end}의 준비시간이 크게 중요하게 작용되지 않으므로 OPT-mesh나 OPT-tree의 깊이와 형태를 관여하는 t_{hold}/t_{end}의 비율은 일정하다. 그래서 두 트리의 깊이는 일정하게 유지되며 트리의 성능이 메시지 크기에 대해 선형이다.

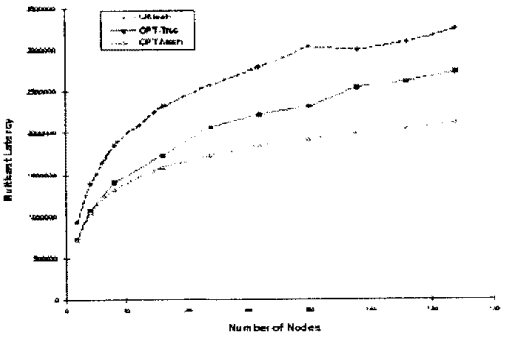


(그림 12) 128-노드의 멀티캐스트 트리비교

(그림 13)과 (14)는 메시지 크기가 각각 4 Kbyte와 64 Kbyte일 때 멀티캐스트 트리의 성능을 보여주며 제안한 OPT-mesh가 OPT-tree와 U-mesh보다 훨씬 효율적이라는 것을 확인해준다.



(그림 13) 메시지 크기 : 4 Kbyte일 때



(그림 14) 메시지 크기 : 64Kbyte일 때

5. 결 론

알고리즘을 디자인할 때 그 알고리즘이 다른 병렬 시스템에서도 우수한 성능을 가지기 위해서는 이식성이 있어야 한다. 본 논문에서는 변수화된 통신 모델을 근거하여, 단일-포트뿐 아니라 다중-포트 통신 구조에 대해 최적의 멀티캐스트 트리를 형성하는 효율적이고 구조에 독립적인 알고리즘(OPT-tree)을 제안하였다. 또한 본 논문에서는 구조에 독립적인 멀티캐스트 알고리즘의 각 네트워크 구조에 따른 튜닝을 연구했다. 제안한 OPT-tree 알고리즘은 현재 개발된 다른 멀티캐스트 알고리즘들에 비해 월등한 성능을 보이지만 네트워크 회선경쟁을 없애기 위한 네트워크 구조에 따른 튜닝을 한다면 그 성능은 더 향상할 수 있다. 따라서, 웜홀 스위치 메쉬 네트워크에서의 회선경쟁이 없는 변수화된 멀티캐스트를 생성하는 OPT-mesh 알고리즘을 제안했다.

본 논문에서 제안한 회선경쟁을 피하기 위한 방법은 어떤 특정한 네트워크에 한정되지 않는다. 이 개념은 네트워크가 회선경쟁이 없는 프로세서 클러스터로 분할될 수 있는 어떠한 네트워크에도 적용할 수 있다.

향후 연구 과제는 이러한 분할이 가능하지 않는 단방향 MIN(uni-directional Multistage Interconnection Network) [9]과 같은 네트워크에서의 멀티캐스트 알고리즘의 튜닝에 관한 연구이다. 이 경우에는 가능한 한 네트워크 회선경쟁을 최소화하는 것이다. 어떤 통신 채널은 두 개의 다른 송신측에 의해서 공유되기도 하지만 같은 통신 채널을 공유하는 송신측들이 가능한 한 동시에 메시지를 보내지 않도록 적절히 그 순서를 정렬시켜야 할 것이다.

참 고 문 헌

[1] Message Passing Interface Forum, "MPI: A message-passing interface standard," March, 1994.

[2] T. H. Cormen, et al., Introduction to Algorithms, MIT Press, 1992.

[3] D. Culler et al., "LogP: Towards a Realistic Model of Parallel Computation," Proc. of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp.1-12, ACM, May, 1993.

[4] T. von Eicken et al., "Active Messages: A Mechanism for Intergrated Communication and Computation," Proc. of Annual International Symposium on Computer Architecture, pp.256-266, May 1993.

[5] H. Franke et al., "MPI-F: An Efficient Implementation of MPI on IBM-SP1," Proc. of International Conference on Parallel Processing, pp. 197-201, Aug., 1994.

[6] W. Gropp, et al., "Users Guide for the ANL IBM SP-1 DRAFT," Tech. Rep. ANL/MCS-TM-00, Argonne National Laboratory, Feb., 1994.

[7] J.-Y. Lee (Park) et al., "Construction of Optimal Multicast Trees Based on the Parameterized Communication Model," Proc. of International Conference on Parallel Processing, pp.180-187, August, 1996.

[8] P. K. McKinley, et al., "Unicast-based Multicast Communication in Wormhole-Routed Networks," IEEE Trans. on Parallel Distributed Systems, vol. 5(12), pp.1252-1265, December, 1994.

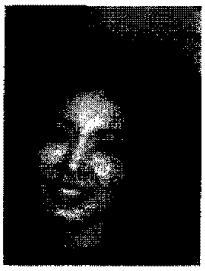
[9] L. M. Ni, et al., "Performance evaluation of switch-based wormhole networks," Proc. of International Conference on Parallel Processing, Vol I, August 1995.

[10] N. Nupairoj and L. M. Ni, "Benchmarking of Multicast Communication Services," Tech. Rep. MSU-CPS-ACS-103, Michigan State Univ., Dept. of Computer Science, Apr. 1995.

[11] H. Xu, et al., "Optimal software multicast in wormhole-routed multistage networks," Proc. of Supercomputing'94, pp.703-712, Nov., 1994.

[12] Gilles Brassard, et al., Fundamentals of Algo-

rithmics, Prentice Hall, Inc., 1996.



이 주 영

e-mail : jylee@namhae.duksung.ac.kr

1984년 이화여자대학교 수학과 졸(학사)

1991년 The George Washington Unvi. 전산학과 졸업(석사)

1996년 The George Washington Univ. 전산학과 졸업(박사)

1996년~현재 덕성여자대학교 전산학과 전임강사

관심분야 : 알고리즘, 병렬처리, 그래프이론