# 웜홀 스위칭하는 양방향 베니언 망에서의 두 단계 멀티캐스트

# (Two-phase Multicast in Wormhole-switched Bidirectional Banyan Networks)

권 위 남 † 권 보 섭 †† 박 재 형 ††† 윤 현 수 ††††

(Winam Kwon) (Boseob Kwon) (Jaehyung Park) (H.Yoon)

**요 약** 다단계 상호 연결망은 대규모 멀티컴퓨터의 대표적인 연결망 구조이다. 브로드캐스트와 멀티캐스트 통신은 캐쉬 관리, 리덕션, 베리어 동기화와 같은 협동 통신을 지원하기 위한 기반 기술이다. 본 논문은 대규모 멀티컴퓨터 시스템을 구성하기에 적합한 웜홀 스위칭하는 양방향 베니언 망에서 동작하는 멀티캐스트 기법을 제안한다. 제안하는 기법은 간단한 하드웨어하에서 교착상태를 일으키지 않고 두 번의 전송 단계를 거침으로써 멀티캐스트와 브로드캐스트를 수행한다. 또한, 원하는 목적 노드의 주소를 큐브로 병합하고, 헤더는 단일 큐브로 인코딩된다. 출력 링크에 경쟁이 발생하면 가장 상위 입력 링크로 부터 들어온 플릿에게 우선 순위를 줌으로써 교착상태를 방지한다. 제안하는 기법을 시뮬레이션을 통해서 통신 지연 시간의 관점에서 다른 기법과 비교 평가함으로써 제안하는 멀티캐스트 기법의 성능이 우수함을 보였다. 또한, 제안하는 브로드캐스트 기법은 팬아웃이 $2^m(2^m \geq \sqrt{N}$인 최소의 정수, $N$은 시스템 크기)인 멀티캐스트의 성능과 유사한 월등한 성능을 낸다는 것을 보였다.

***Abstract*** A multistage interconnection network is a suitable class of interconnection architecture for constructing large-scale multicomputers. Broadcast and multicast communication are fundamental in supporting collective communication operations such as reduction and barrier synchronization. In this paper, we propose a new multicast technique in wormhole-switched bidirectional multistage banyan networks for constructing large-scale multicomputers. To efficiently support broadcast and multicast with simple additional hardware without deadlock, we propose a two-phase multicast algorithm which takes only two transmissions to perform a broadcast and a multicast to an arbitrary number of desired destinations. We encode a header as a cube and adopt the most upper input link first scheme with periodic priority rotation as arbitration mechanism on contented output links. We coalesce the desired destination addresses into multiple number of cubes. And then, we evaluate the performance of the proposed algorithm by simulation.

The proposed two-phase multicast algorithm makes a significant improvement in terms of latency. It is noticeable that the two-phase algorithm keeps broadcast latency as efficient as the multicast latency of fanout $2^m$ where $m$ is the minimum integer satisfying $2^m \geq \sqrt{N}$( $N$ is a network size).

· 본 연구는 첨단정보기술연구센터를 통하여 과학재단의 지원을 받음
· 본 연구는 한국과학재단의 특정기초연구과제 "무선데이타 통신 시스템 구현을 위한 기반기술 연구"의 지원을 받았음.

† 비 회 원 : 한국과학기술원 전자전산학과 전산학전공
wnkwon@camars.kaist.ac.kr
†† 정 회 원 : 안동대학교 컴퓨터공학교육과 교수
bxkwon@andong.ac.kr
††† 비 회 원 : 한국전자통신연구원 인터넷기술연구부 연구원
jachyung@ctri.re.kr
†††† 종신회원 : 한국과학기술원 전자전산학과 전산학전공 교수
hyoon@camars.kaist.ac.kr

논문접수 : 1998년 9월 10일
심사완료 : 1999년 12월 3일

## 1. 개 요

멀티컴퓨터(multicomputer) 시스템은 프로세싱 노드(processing node)와 상호 연결망(interconnection network)으로 구성된다. 다단계 상호 연결망(multistage interconnection network; MIN)은 대표적인 상호 연결망 구조이다. 멀티컴퓨터 시스템에서는 프로세싱 노드가 메시지(message) 기반으로 통신하기 때문에, 많은 수의 프로세싱 노드를 가지는 멀티컴퓨터 시스템을 구축하기 위해서는 효율적인 통신망을 구성하는 것이 필수적이다[1]. TMC CM-5[2], IBM SP-1/2[3]와 Meiko CS-2

[4]와 같은 상용 멀티컴퓨터 시스템의 통신망은 양방향 다단계 상호 연결망(bidirectional-MIN)으로 구축되어 있다. 양방향 다단계 상호 연결망은 단방향 다단계 상호 연결망에 비해 좋은 성능을 낼 수 있는 장점이 있는 반면에 라우팅 기법을 개발하는 데에 어려움이 있다.

대부분의 상용화된 다단계 상호 연결망은 웜홀 스위칭(wormhole switching)기법으로 패킷을 전송한다. 웜홀 스위칭 기법은 패킷을 헤더 플릿(header flit)과 데이타 플릿(data flit)으로 나눈다. 이 기법은 헤더 플릿이 패킷의 라우팅을 전담하고 데이타 플릿이 헤더 플릿의 뒤를 따라 파이프라인(pipeline) 형태로 전송되기 때문에 통신 지연 시간(latency)이 짧아서 성능이 좋다. 반면에, 멀티캐스트(multicast) 시에 교착상태(deadlock)가 발생할 수 있다. 교착상태가 발생하면 통신망의 성능이 급격히 저하된다. 따라서, 멀티캐스트 기법은 교착상태를 방지하도록 설계되어야 한다[5].

멀티캐스트(multicast)는 한 프로세싱 노드가 동일한 패킷을 다수개의 프로세싱 노드에 전달하는 통신 기법이다. 양방향 베니언(banyan) 망에서 동작하는 멀티캐스트 알고리즘은 크게

소프트웨어적 접근 방법과 하드웨어적 접근 방법으로 나뉜다. 소프트웨어적 접근 방법에는 U-min[1]과 C-min[6]이 있다. 이 기법들은 추가의 하드웨어가 불필요하고 교착상태를 발생시키지 않는 반면에, 한 번의 멀티캐스트를 수행하기 위해서 $\lceil \log d \rceil$ ( $d$는 팬아웃(fanout) )단계의 전송을 하기 때문에 통신 지연 시간이 길다. 하드웨어적 접근 방법에는 [7]이 있다. [7]의 기법은 각 스위칭 소자에 중앙 버퍼가 있고, 웜홀 스위칭(buffered wormhole switching) 기법을 기반으로 블록된(blocked) 플릿은 버퍼에 저장하고 블록되지 않은 플릿은 복제하여 전송하는 비동기 복제(asynchronous replication) 방식으로 동작한다. 이 기법은 한 단계의 전송으로 멀티캐스트를 완료하기 때문에 지연 시간이 짧은 반면에, 교착상태를 막기 위해서 각 스위칭 소자마다 2차원 연결리스트로 구현된 중앙 버퍼 혹은 입력 큐를 두기 때문에 하드웨어 복잡도가 높아서 대용량 멀티컴퓨터 시스템을 구축하는 데는 적합하지 않다[8].

본 논문은, 동기 복제 로직(synchronous replication logic)과 출력 링크 당 한 플릿 크기의 출력버퍼만을 가지고 순수(pure) 웜홀 스위칭하는 양방향 베니언 망을 기반으로 하는 두 단계만에 멀티캐스트하는 기법을 제안한다. 제안하는 기법은 단계-I과 단계-II의 두 단계로 구성된다. 단계-I에서는 멀티캐스트 패킷을 하나의 소스 노드(source node)에서 미리 계산된 중간 목적 노드(intermediate destination node)로 전송한다. 단계-II에서는 중간 목적 노드가 전송 받은 멀티캐스트 패킷을 원래 목적 노드(desired destination node)로 전송한다. 또한, 헤더(header)를 큐브(cube)로 인코딩(encoding)하고, 교착 상태를 막기 위해서 최상위 입력 링크 우선 기법(the most upper input link first scheme)을 사용한다. 스위칭 소자의 구조가 단순하고 멀티캐스트 알고리즘이 멀티캐스트와 브로드캐스트(broadcast)를 효율적으로 제공하기 때문에, 제안된 기법은 대용량의 멀티컴퓨터 시스템을 구축하기에 적합하다.

본 논문은 다음과 같이 구성되어 있다. 다음 절에서 시스템 구조를 설명하고 3 절에서는 제안하는 두 단계 멀티캐스트 기법을 설명한다. 4 절은 시뮬레이션을 통해서 제안하는 기법의 성능 분석 결과를 설명하고 5절에서 결론을 맺는다.

## 2. 시스템 구조

본 논문에서는, $2k \times 2k$ 크기의 스위칭 소자를 기반으로 구성된 $N \times N$ 양방향 베니언 망을 가정한다. 설명을 간단히 하기 위해서 그림 1과 같이 통신망의 왼쪽에 모든 프로세싱 노드가 연결되어 있다고 하자.

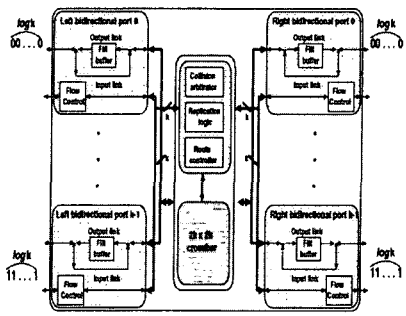


그림 1 $2k \times 2k$ 양방향 스위칭 소자

그림 1은 $2k \times 2k$ 스위칭 소자를 보여준다. 이 소자는 $2k \times 2k$ 크로스바(crossbar)와 $2k$개의 양방향 포트로 구성되어 있다. 단방향 화살표와 양방향 화살표는 각각 단방향 링크(link)와 양방향 링크를 표시한다. 양방향 포트는 입력 링크(input link)를 가진 단방향 포트와 출력 링크(output link)를 가진 단방향 포트로 구현될 수 있고, 플릿을 동시에 보내고 받을 수 있다. 각 출력 링크에는 한 플릿 크기의 출력 버퍼가 있고 입력 버퍼나 중앙 버퍼가 없다.

통신망은 동기 복제 방식과 순수 웜홀 스위칭 기법을

기반으로 패킷을 전송한다 . 패킷의 복제와 전송에 필요한 제어 정보는 피드백 로직(feedback logic)에 의하여 인접한 두 단(*stage*)의 스위칭 소자들간에 전달되고 이 정보에 의하여 망내에서의 패킷 전송이 동기화 된다. 동기 복제 방식에서는 복제된 모든 플릿이 블록되지 않은 경우에만 그 패킷을 통신망에서 라우팅하기 때문에 멀티캐스트의 팬아웃(fanout)이 커질 수록 블록될 확률이 커진다.

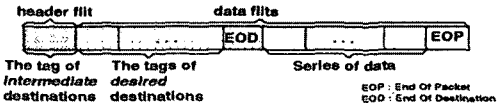
플릿은 소스 노드에서 최소 공유 단(least common ancestor; LCA)[9]까지 적응 전진 전송(adaptive forward transmission)되고, 목적지 태그(destination tag routing)에 의하여 LCA에서 회전 전송(turnaround transmission)되며, LCA에서 목적 노드까지 후진 전송(backward transmission)된다. 전진 전송은 유니캐스트(unicast)이고 회전 전송과 후진 전송은 멀티캐스트이다. 헤더 플릿이 LCA 단으로 전진 전송중이면 스위칭 소자는 비어있는 오른쪽 가장 상위 출력 링크의 출력 버퍼로 전송하고, 헤더 플릿이 회전 전송 혹은 후진 전송중이면 목적지 태그에 의해서 결정되는 왼쪽 출력 링크의 출력 버퍼로 전송한다. [7]의 멀티캐스트 방식은, 여러개의 플릿으로 나누어진 패킷을 중앙버퍼에서 제어하기 위해 중앙버퍼를 2 차원 연결 리스트로 구현하고 하드웨어로 관리해야 하기 때문에 하드웨어 복잡도가 높다. 반면, 본 논문이 가정하는 스위칭 구조는 중앙버퍼가 없고 포트마다 한 플릿 크기의 버퍼만 있기 때문에 제어가 간단하고 중앙버퍼 관리를 위한 하드웨어가 불필요하다.

## 3. 두 단계 멀티캐스트 기법

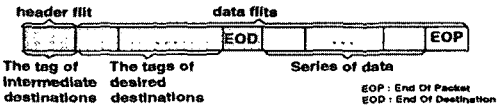
### 3.1 멀티캐스트 패킷 포맷(format)

그림 2는 멀티캐스트 패킷의 포맷을 보여준다. 단계-I과 단계-II의 패킷은 하나의 헤더 플릿과 여러개의 데이타 플릿으로 구성된다. 두 단계 멀티캐스트 기법은 단계-I에서 소스 노드가 중간 목적 노드로 패킷을 전송하고, 단계-II에서 단계-I의 중간 목적 노드들이 원래 목적 노드로 패킷을 전송함으로써 멀티캐스트를 수행한다.

단계-I과 단계-II의 헤더 플릿은 각각 중간 목적 노드의 태그와 원래 목적 노드의 태그를 포함한다. 단계-I의 데이타 플릿의 상부에는 원래 목적 노드의 태그 정보가 저장되어 있어서 중간 목적 노드가 원래 목적 노드의 정보를 알 수 있다. 원래 목적 노드의 태그의 개수와 패킷의 길이가 가변이므로 EOD(Endo-Of-Destination) 플릿과 EOP(End-Of-Packet) 플릿을 각각의 끝에 붙혀 끝을 표시한다.




(a) The multicast packet format in phaseI




(a) The multicast packet format in phaseII

그림 2 단계-I과 단계-II의 패킷 포맷

### 3.2 헤더 인코딩

제안하는 기법은 헤더를 큐브 인코딩하는데 $2k\times 2k$ 크기의 스위칭 소자를 사용하므로 아래에서 정의하는 $k$-진 큐브로 인코딩한다. 정의된 $k$-진 큐브는 모든 복제 비트가 헤더의 최하위에 연속으로 배치되는 특징이 있다. 이는 최상위 입력 링크 우선 기법으로 교착상태를 방지하기 위한 것이다. $k$-진 큐브는 이진 큐브로 구현되어 패킷의 복제와 라우팅에 사용된다.

**$k$-진 큐브 정의:** 이진 큐브 $y=b_{\log k-1}\cdots b_m x_{m-1}\cdots x_0$ (단, $b_j\in 0,1$이고 $x_i$는 2진 복제 비트), $X=x_{\log k-1}\cdots x_0$를 $k$-진 복제 비트라고 하면, $N\times N$ 크기의 망에서 쓰이는 **크기가 $2^m k^c$인 $k$-진 큐브**는 $d_{n-1}\cdots d_{c+1}(y)X_{c-1}\cdots X_0$ (단, $n=\log_k N, 0\le c\le n-1, d_i$는 $k-$진수)로 정의된다.

예를 들면, $4\times 4$ 스위칭 소자로 구성된 $64\times 64$ 크기의 망은 $n$이 3이고, $k$가 4이다. 이 시스템내의 220부터 233까지의 프로세싱 노드는 4-진 큐브 $2(1x)X$로 인코딩되고, 이진수 $101xxx$로 표현된다.

### 3.3 멀티캐스트 알고리즘

본 논문이 제안하는 멀티캐스트 알고리즘은 단계-I과 단계-II로 구성된다. 멀티캐스트를 수행하기 위해서, 단계-I에서 소스는 원래 목적 노드의 태그와 중간 목적 노드의 태그를 계산하고 멀티캐스트 패킷을 구성하여 전송한다. 단계-II에서 중간 목적 노드는 자신에게 할당된 원래 목적 노드의 태그를 전송 받은 패킷에서 읽어내어 멀티캐스트 패킷을 재구성하고 원래 목적 노드로 전송한다. 원래 목적 노드가 멀티캐스트 패킷을 받는 시점에서 멀티캐스트가 종료된다.

**단계-I 알고리즘** : 소스 노드 $s$에서의 멀티캐스트

**입력** :

$D=\{D_0,\ldots,D_{d-1}\}$ : 정렬된 원래 목적 노드의 주소

$p$ : 보낼 데이타
**begin**
$C(=\{C_0,\ldots,C_{c-1}\}) \leftarrow Encode(D)$;
$2^l \geq |C|$ 인 최소의 정수 $l$을 계산;
크기가 $2^l$인 $k$-진 큐브 $H$를 임의로 선택;
멀티캐스트 패킷 $P(=H|C|EOD|p|EOP)$를 구성;
$P$를 중간 목적 노드인 $H$로 전송;
**end**

단계-I 알고리즘과 단계-II 알고리즘은 노드가 멀티캐스트 패킷을 구성하는 과정을 보여준다. 멀티캐스전에 단계-I에서 소스 노드는 첫 째, 원래 목적 노드의 태그들( $C$)을 *Encode* 알고리즘을 이용하여 계산한다. 둘 째, 계산한 원래 목적 노드의 태그의 개수( $c$)이상인 최소의 2의 멱승( $2^l$)을 계산한다. 세 째, 크기가 $2^l$인 $k$-진 큐브( $H$)를 임의로 선택한다. 그 후에, 소스 노드는 그림 2.(a)와 같이 중간 목적 노드의 태그 $H$를 헤더 플릿에, 원래 목적 노드의 태그들 $C$와 데이타 $p$를 데이타 플릿에 기록함으로써 멀티캐스트 패킷( $P$)을 구성한다. 마지막으로, 소스 노드가 구성한 멀티캐스트 패킷을 통신망에 전송하고 종료한다.

**단계-II 알고리즘** : 중간 목적 노드 $i$에서의 멀티캐스트
**입력** :
$P(=H|C|EOD|p|EOP)$ : 단계-I의 멀티캐스트 패킷
**begin**
**if** $i$가 단계-I의 멀티캐스트 패킷 $P$를 받는다
**then**
$d \leftarrow i$가 $H$내에서 몇 번째 노드인지를 계산;
**if** $d > c-1$ **then** $P$를 소거;
**else if** $C_d = i$ **then** $P$를 흡수;
/* $i$가 원래 목적 노드 */
**else**
/* $i$는 해당하는 원래 목적 노드로 멀티캐스를 수행 */
$P$에서 $C_d$와 $p$를 추출;
멀티캐스트 패킷 $P'(=C_d|p|EOP)$를 재구성;
$P'$를 원래 목적 노드 $C_d$로 전송;
**endif**
**endif**
**end**

중간 목적 노드가 단계-I의 멀티캐스트 패킷 $P$를 받자마자 단계-II가 시작된다. 패킷을 받은 중간 목적 노드는 단계-II를 위한 멀티캐스트 패킷 $P'$을 재구성한다. $d$ 번째로 주소가 적은 중간 목적 노드는 $P$에서 $d$번째 원래 목적 노드의 태그 $C_d$와 전송할 자료 $p$를 읽어서 각각 패킷 $P'$의 헤더 플릿과 자료 플릿에 적고, EOP 플릿을 끝에 덧 붙임으로써 그림 2.(b)와 같이 $P'$를 재구성한다. 재구성된 패킷을 통신망을 통하여 원래 목적지로 전송한다.

### 3.4 *Encode* 기법

*Encode*는 직관적으로 no-operation 혹은 목적 노드 주소를 오름차순 혹은 내림차순으로 정렬하도록 설계할 수 있다. 그러나, 동기 복제 방식을 사용하는 구조에서는 원래 목적지의 수가 증가할수록 팬아웃이 커져서 블록되는 확률이 커지기 때문에, 멀티캐스트의 지연 시간이 길어진다. 그림 4.(c)의 점선이 보여주는 바와 같이 팬아웃이 시스템내의 프로세싱 노드의 개수(이하 시스템의 크기라 함)의 반 이상이면 지연시간이 급격히 증가한다. 본 논문에서는 팬아웃을 시스템 크기의 반 이하로 유지시키기 위하여 **Coalesce** 알고리즘을 *Encode* 기법으로 제안한다.

**Coalesce** 알고리즘은 원래 목적 노드의 주소를 여러 개의 $k$-진 큐브로 병합한다. 아래의 알고리즘에서 최대 크기의 $k$-진 큐브는 원래 목적 노드 집합 $D$의 부분 집합으로 구성할 수 있는 최대 크기의 $k$-진 큐브를 의미한다.

**Coalesce 알고리즘** : $k$-진 큐브로 병합
**입력** :
$D=\{D_0,\ldots,D_d\}$ : $k$-진수의 집합
**begin**
$C \leftarrow \emptyset$;
**do**
$D$의 부분집합으로 구성할 수 있는 최대크기의 $k$-진 큐브 $C_i(=d_{n-1}\cdots(y)x_{c-1}\cdots x_0)$를 구성;
$C \leftarrow C \cup \{C_i\}$;
$D \leftarrow D - \{C_i\}$;
**until** $D \equiv \emptyset$;
$C$를 출력;
**end**

예를 들어 **Coalesce** 알고리즘으로 구현된 *Encode*는 4-진수의 집합 { 00, 02, 03, 12, 22, 23, 30, 31, 32, 33 }을 { 00, 0(1 $x$), 12, 2(1 $x$), 3 $X$ }로 병합한다.

### 3.5 브로드캐스트

브로드캐스트의 경우에는 **Coalesce**의 특성상 단계-II의 팬아웃이 시스템 크기이기 때문에 병합효과가 없다. 단계-I의 팬아웃을 작은 값으로 유지하고 단계-II의 팬아웃 값을 줄이기 위해서 *Encode*는 $N$개의 원래 목적 노드 주소를 아래의 **Partition** 알고리즘을 이용해서 여러 개의 $k$-진 큐브로 분할한다.

**Partition 알고리즘** : $2^m$개의 크기가 $2^l$인 $k$-진 큐브로 분할

**입력** : $N(=2^n)$

**begin**

$l \le \frac{n}{2}$인 최대의 정수 $l$을 계산;

$m \ge \frac{n}{2}$인 최소의 정수 $m$을 계산;

0에서 $N-1$까지의 숫자를 $2^m$개의 크기가 $2^l$인 $k$-진 큐브로 분할;

계산된 $k$-진 큐브를 출력;

**end**

예를 들어, $N$이 256이면 **Partition** 알고리즘은 256개의 원래 목적 노드 주소를 16개의 크기가 16인 $k$-진 큐브로 분할한다.

### 3.6 교착상태 방지

두 단계 멀티캐스트 알고리즘은 여러개의 멀티캐스트 패킷이 망에서 동시에 전송되면 교착상태를 발생시킬 수 있다. 베니언 망에서는 스위칭 소자가 서로 독립적으로 동작하기 때문에 제어가 분산되어 있어서, 복제된 플릿들은 다른 플릿들의 블록 여부를 알 수 없다. 따라서, 출력 링크에 경쟁이 발생할 경우 원형 대기(circular waiting) 상태에 빠져서 교착상태가 발생할 수 있다.

본 논문에서는 경쟁이 발생했을 때에 가장 상위 입력 링크로 들어오는 플릿에 우선 순위를 둠으로써(the most upper input link first scheme) 원형 대기를 제거하여 교착상태를 방지하였다. 이 방식은 우선 순위를 회전시키는 하드웨어를 추가함으로써 낮은 우선 순위를 가진 입력 링크로 들어오는 패킷이 기아(starvation) 상태에 빠지는 것을 방지할 수 있다. 채택한 교착상태 방지 방식은 베니언 망의 특성때문에 교착상태를 발생시키지 않고 패킷을 전송시킨다.

### 3.7 두 단계 알고리즘의 예

그림 1은, 8×8 스위칭 소자로 구성한 16×16 양방향 큐브 망에서 소스 노드 10이 두 단계 알고리즘을 이용해서 패킷을 원래 목적 노드 { 00, 02, 03, 12, 22, 23, 30, 31, 32, 33 }으로 전송하는 예를 보여준다.

먼저, 소스는 단계-I에서 소스는 원래 목적 노드를 { 00, $0(1x)$, 12, $2(1x)$, $3X$ }로 인코딩한다. 인코딩된 원래 목적 노드의 태그가 5개이므로 크기가 8인 4-진 큐브 $(1x)X$를 임의로 선택한다. 소스 노드는 멀티캐스트 패킷을 구성하고 중간 목적 노드 $(1x)X$로 전송한다. 단계-II에서는 중간 목적 노드중에서 { 20, 21, 22, 23, 31 }은 멀티캐스트 패킷을 재구성하여 각각에 할당된 원래 목적 노드 { 00, $0(1x)$, 12, $2(1x)$, $3X$ }로 전송한다. 나머지 중간 목적 노드는 받은 패킷을 버리고 종료한다. 모든 원래 목적 노드가 패킷을 다 전송 받으면 멀티캐스트가 종료된다.

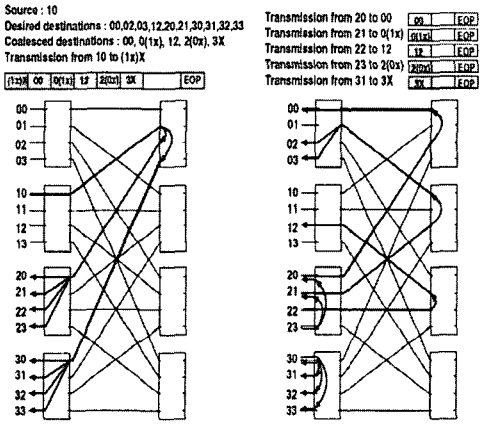



그림 3 두 단계 멀티캐스트 기법의 예

## 4. 성능 분석

두 단계 멀티캐스트 기법의 성능을 평가하기 위해서 시뮬레이션을 통해 u-min[1]과 비교 평가했다. [7]은 본 논문의 시스템 구조하에서는 교착상태를 발생시키기 때문에 비교 대상에서 제외시켰다.

### 4.1 시뮬레이션 인자와 환경

멀티캐스트 트래픽을 평가하기 위한 측정치는 시뮬레이션마다 다르다. 첫 째, 멀티캐스트 지연 시간은 마지막 패킷이 목적 노드에 도달한 시간, 혹은 각 패킷이 도달한 시간의 평균으로 정의될 수 있다. 마지막 패킷이 목적 노드에 도달한 시간이 협력 통신(collective communication)의 성능에 더 중요한 인자이고[10], 공유 메모리 시스템에서 캐쉬 라인(cache line)을 무효화(invalidation)시킬 때에 더 중요하기 때문에, 본 논문에서는 이것을 지연 시간으로 계산한다. 둘 째, 멀티캐스트 패킷에 의해서 망에 가해지는 부하는 입력 부하(injected load)와 실제 부하(effective load)로 정의될

수 있는데, 본 논문에서는 실제 부하값을 바로 알기 어렵기 때문에 망의 부하를 입력 부하로 정의한다. 마지막으로, 유니캐스트는 시동시간(startup) 오버헤드는 무시될 수 있을 정도로 작은 반면에 멀티캐스트의 시동시간 오버헤드는 멀티캐스트 패킷을 구성하는데 드는 시간이 길기 때문에 무시될 수 없다.

본 논문에서는 시동시간 오버헤드를 $2\log N$으로 가정한다[1]. 본 논문은 시스템이 $2K\times 2K$ 크기의 스위칭 소자로 구성된 $N\times N$ 베니언 망을 가정한다. 지연 시간은 한 플릿이 한 스위칭 소자에서 다음 스위칭 소자로 블라킹없이 전송되는 데 걸리는 시간을 1 시간 단위로 하여 측정했고, 측정된 값은 패킷이 소스 노드에 도착한 순간부터 원래 목적 노드에 도착을 완료한 순간까지의 시간을 의미한다. 플릿의 크기는 2 바이트이고, 32 바이트 길이의 자료가 평균 도착율이 $2/I$인 지수분포를 따라 통신망에 입력된다고 가정한다. 멀티캐스트 패킷의 팬아웃을 $F$로 고정시키고 원래 목적 노드의 분포는 균일 분호를 따른다고 가정한다. 시스템이 안정된 이후의 측정치를 얻기 위하여 처음 50,000번의 측정치는 사용하지 않았다.

본 논문에서는 $N$= 256, $K$= 4, 16이고, $F$=16, 64, 128, 192, 240, 256일 때와, $N$= 64, $K$= 4이고 $F$= 16, 32, 48, 64 때를 시뮬레이션 했다. 이 시뮬레이션 인자들은 일반적으로 널리 가정되는 것들로서 비교되는 관련 연구의 시뮬레이션 환경을 따른 것이다.

### 4.2 시뮬레이션 결과

두 단계 멀티캐스트 기법의 시뮬레이션 결과가 그림 4에 있다. 그림 4.(a)와 (b)는 두 단계 멀티캐스트 기법을 u-min 기법과 비교한 결과이다. 두 단계 알고리즘이 u-min에 비해서 낮은 지연 시간을 가지고 지연시간의 증가율도 더 느림을 볼 수 있다. 그림 4.(b)는 두 기법의 팬아웃에 따른 지연 시간의 비를 보여준다. $F/N$이 증가함에 따라 지연 시간의 비가 증가하고 두 단계 알고리즘이 u-min에 비하여 최소한 2배의 성능을 낸다는 것을 알 수 있다.

그림 4.(c)는 두 단계 알고리즘에서 사용되는 **Coalesce**의 병합효과를 보여준다. **Coalesce**를 사용하지 않는 두 단계 알고리즘은 $N/2$ 이상의 팬아웃을 가지는 경우 지연 시간이 갑자기 커지는 현상을 보이지만 **Coalesce**를 사용하는 경우는 팬아웃이 $N/2$보다 크더라도 지연시간이 어느 한계 이상은 증가하지 않는다. 이는 팬아웃이 $N/2$이상인 경우는 반드시 원래 목적 노드의 주소가 병합되기 때문에 팬아웃이 $N/2$이하로 유지되어 블라킹 확률이 어느 한계이상으로 증가하지 않기 때문이다. 또한, $F/N$가 어떤 수치 이상으로 증가하면 병합 효과에 의해서 팬아웃이 오히려 줄어들기 때문에 지연 시간이 더 감소한다.

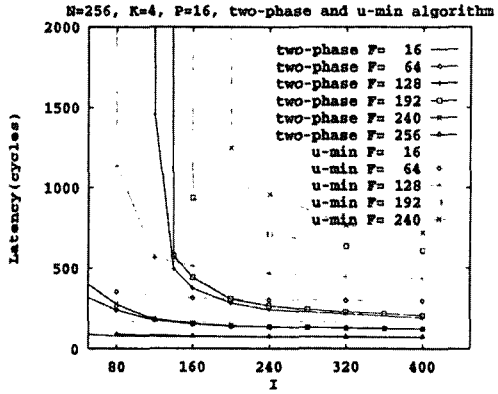
그림 4.(d)는 멀티캐스트 패킷의 오버헤드를 보여준다. 오버헤드는 망을 통해서 지나간 헤더 플릿의 총 개수, 원래 목적 노드의 태그를 저장한 플릿의 총 개수, EOD 플릿의 총 개수, 그리고 EOP 플릿의 총 개수의 합으로 정의한다. 두 단계 알고리즘의 멀티캐스트 패킷 오버헤드는 u-min보다 작고, u-min은 팬아웃이 증가하면 오버헤드가 증가하는 데 반해 두 단계 알고리즘은 팬아웃이 어느 수준 이상으로 증가하면 감소하기 시작한다. 이는 원래 목적 노드 주소의 병합율이 증가하기 때문이다.

그림 4.(e)는 스위칭 소자의 크기가 $8\times 8$인 경우와 $32\times 32$인 경우에 팬아웃에 따른 지연시간을 보여준다. 스위칭 소자의 크기가 클수록 확연히 짧은 지연 시간을 가짐을 볼 수 있다.

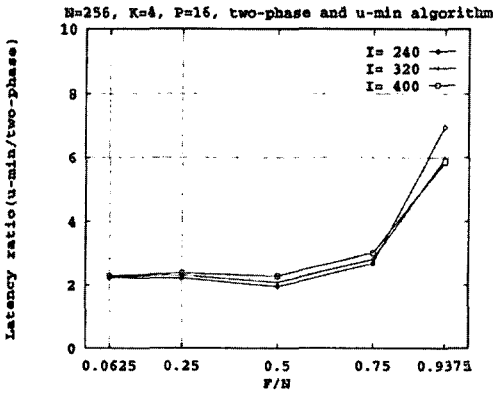
그림 4.(f)는 통신망 크기가 지연 시간에 미치는 영향을 보여준다. 통신망의 크기가 변해도 지연시간 그래프의 모양은 비슷하지만, 통신망의 크기가 커질 수록 거쳐야 하는 단의 수가 늘어나기 때문에 블라킹 확률이 커져서 지연 시간이 길어진다. 또한, 통신망의 크기가 커질 수록 팬아웃 값에 따른 지연시간의 변화량도 크다는 것을 알 수 있다.

두 단계 알고리즘의 브로드캐스트 지연 시간은 u-min에 비해서 월등히 낮고, **Partition** 알고리즘의 특성상 팬아웃이 $2^m$($m$은 $2^m\geq\sqrt{N}$을 만족하는 최소의 정수)인 멀티캐스트 지연 시간과 유사하다. 그림 4.(f)는 $N$= 256(64)일 때 브로드캐스트 지연 시간의 그래프이다. 이 그래프는 팬아웃이 16(8)인 멀티캐스트 지연 시간과 유사함을 보여준다. 두 단계 알고리즘의 브로드캐스트는 u-min에 비해 월등한 성능을 낸다.

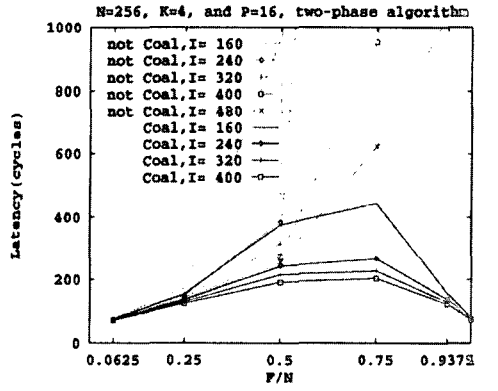
기존의 소프트웨어적 기법이 복잡도가 낮은 하드웨어를 기반으로 log급 전송 단계를 거치는 멀티캐스트를 수행하는 특징이 있는 반면, 기존의 하드웨어적 기법은 복잡도가 높은 하드웨어를 기반으로 단 한번의 전송 단계를 거치는 특징이 있다. 또한, 기존의 소프트웨어적 기법은 각 단계의 끝단에서 헤더를 재구성하기 때문에 각 단계의 끝단에서의 지연시간이 긴 특징이 있고, 기존의 하드웨어적 기법은 전송시에 거치는 스위칭 소자 내부의 중앙 버퍼에서 대기하는 시간이 긴 특징이 있다. 그런데, 제안하는 기법은 소프트웨어적 기법의 하드웨어에 복제 로직만 추가한 간단한 하드웨어를 기반으로 오직 두 전송 단계만을 거침으로써 멀티캐스트를 수행하



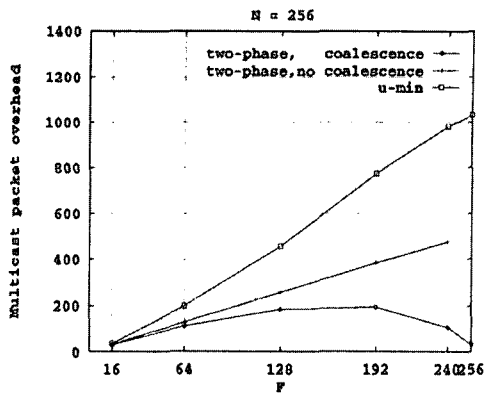

(a) 지연 시간 대 평균 도착 시간 간격(I/2)



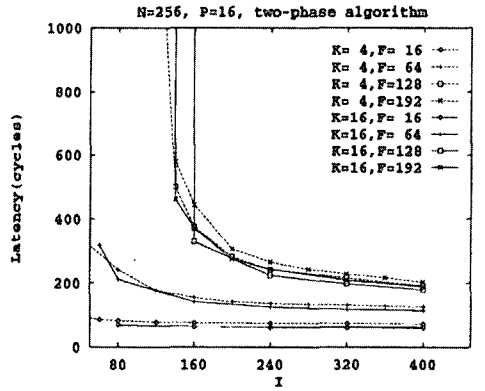

(b) (두 단계 멀티캐스트의 지연 시간)/(u-min의 지연 시간) 대 (팬아웃)/(통신망 크기)



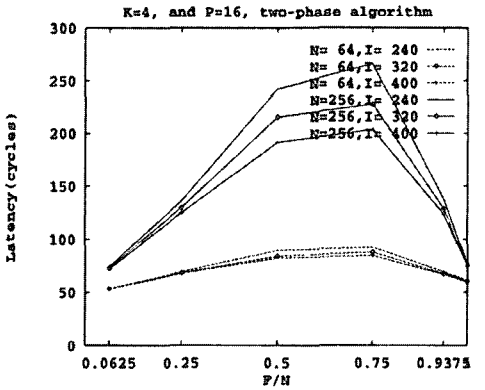

(c) 두 단계 멀티캐스트 알고리즘에서 Coalesce의 영향 : 지연시간 vs. (팬아웃)/(통신망 크기)




(d) 팬아웃에 따른 멀티캐스트 패킷의 오버헤드




(e) 두 단계 멀티캐스트의 스위칭 소자의 크기에 따른 지연 시간 : 지연시간 대 팬아웃




(f) 두 단계 멀티캐스트의 통신망 크기에 따른 지연 시간 : 지연시간 대 (팬아웃)/(통신망 크기)

그림 4 시뮬레이션 결과

는 특징이 있으며, 1 단계 전송 후에 1회의 헤더 재구성만으로 스위칭 소자의 내부의 버퍼에서 대기하지 않고 멀티캐스를 완료하는 특징이 있다. 또한, 제안하는 기법은 팬아웃이 큰 멀티캐스트의 경우에는 기존의 방법에 비해 월등한 성능을 가진다.

지금까지의 결과로부터, 제안한 기법은 단순한 하드웨어 구조하에서 효과적인 멀티캐스트와 브로드캐스트 통신을 제공하기 때문에 대용량의 멀티컴퓨터 시스템을 구축하기에 적합하다는 것을 알 수 있다.

## 5. 결 론

순수 웜홀 스위칭하는 양방향 메니언 망에서 동작하는 두 단계 멀티캐스트 기법을 제안했다. 망을 구성하는 스위칭 소자에는 $2K$개의 양방향 포트와 동기 복제 로직을 가지고 있다. 양방향 포트는 입력 링크와 한 플릿 크기의 출력 버퍼를 가진 출력 링크로 구성된다.

본 논문은 제안한 기법을 시뮬레이션을 이용해서 멀티캐스트 지연 시간과 패킷 오버헤드의 견지에서 성능을 비교 평가했다. 제안한 기법은 단지 두 단계만에 멀티캐스트를 수행하기 때문에 지연시간이 짧다. 단계-I에서는 소스 노드가 중간 목적 노드로 패킷을 전송하고 단계-II에서는 중간 목적 노드가 원래 목적 노드로 패킷을 전송한다. 각 단계의 헤더는 한 개의 $k$-진 큐브로 인코딩되고 원래 목적 노드는 여러개의 $k$-진 큐브로 병합되기 때문에 팬아웃이 큰 경우에도 효율적으로 동작한다. 또한, 출력 링크에 대한 경쟁이 발생하면 가장 상위 입력 링크로 들어오는 플릿에게 우선순위를 줌으로써 교착상태를 방지했다.

원래 목적 노드의 수가 증가할 수록 두 단계 알고리즘의 지연 시간과 패킷 오버헤드가 증가하기는 하지만 **Coalesce** 알고리즘의 병합 효과때문에 이 값들이 어느 수준이하로 제한되고 팬아웃이 높을 수록 기존의 방식에 비해 지연시간이 짧아진다. 제안하는 브로드캐스트 기법이 타 방식에 비해서 지연 시간이 짧을 뿐 아니라, 팬아웃이 $2^m$($m$은 $2^m \geq \sqrt{N}$을 만족하는 최소의 자연수)인 멀티캐스트 통신의 지연 시간과 유사한 월등한 성능을 낸다. 결과적으로, 제안한 기법은 단순한 하드웨어 구조하에서 효과적인 멀티캐스트와 브로드캐스트 통신을 제공하기 때문에 대용량의 멀티컴퓨터 시스템을 구축하기에 적합하다.

## 참 고 문 헌

[1] H. Xu and Y.-D. Gui and L. M. Ni, "Optimal Software Multicast in Wormhole-Routed Multistage Networks," Proc. of Supercomputing, pg. 1252-1265, Dec. 1994

[2] C. E. Leiserson and et al, "The Network Architecture of the Connection Machine CM-5," Proc. of the ACM Symposium on Parallel Algorithms and Architectures, pg. 272-285, 1992

[3] C. B. Stunkel and et al, "Architecture and implementation of Vulcan," Proc. of the Int'l Parallel Processing Symposium, pg. 268-274, June 1995

[4] Meiko Limited, Waltham, MA, "Computing Surface: CS-2," Communications Networks, June 1993

[5] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," IEEE Computer, Vol.26, pg. 62-76, Feb. 1993

[6] C. -M. Chiang, "Multicasting in Multistage Interconnection Networks," Dept. of Comp. Sci., Michigan State Univ., East Lansing, MI, 1995

[7] C. B. Stunkel, R. Sivaram and D. K. Panda, "Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact," Proc. of the 24th ACM Annual Symposium on Computer Architecture, pg. 50-61, June 1997

[8] D. K. Panda and R. Sivaram, "Fast Broadcast and Multicast in Wormhole Multistage Networks with Multidestination Worms," OSU-CISRC-4/95-TR21, Dept. of Comp. and Info. Sci., Ohio State Univ., 1995

[9] I. D. Scherson and D. -H. Chien, "Least Common Ancestor Networks," Proc. of the 7th Int. Parallel Processing Symposium., pg. 507-513, 1993

[10] N. Nupairoj and L. M. Ni, "Issues in Designing Efficient and Practical Algorithms for Collective Communication in Wormhole-Routed Systems," 1st International Workshop on Communication and Architectural Support for Network-Based Parallel Computing, pg. 212-226, Feb. 1997

권 위 남

1995년 한국과학기술원 전산학과 학사. 1997년 한국과학기술원 전산학과 석사. 1997년 현재 한국과학기술원 전산학과 박사과정 재학중. 관심분야는 Internet, 고속 IP 라우터, QoS, Multicast, Message-Passing Multicomputer

권 보 섭

1983년 경북대학교 전자공학과 학사. 1990년 충남대학교 전자공학과 석사. 1997년 한국과학기술원 전산학과 박사. 1983년 ~ 1985년 LG전자 연구원. 1985년 ~ 1998년 한국전자통신연구원 선임연구원. 1998년 ~ 현재 안동대학교 컴퓨터교육과. 관심분야는 상호연결 네트워크, 병렬 컴퓨터 구조, 컴퓨터 성능평가임

박 재 형

1991년 연세대학교 전산과학과 졸업. 1993년 한국과학기술원 전산학과 석사. 1997년 한국과학기술원 전산학과 박사. 1998년 Perdue Univ. CS Dept. 포닥. 1998년 ~ 현재 한국전자통신연구원 연구원. 관심분야는 Parallel Computer Architecture, Message-Passing Multicomputer and Routing Algorithm, ATM Switch Architecture, Internet

윤 현 수

1979년 서울대학교 전자공학과 학사. 1981년 한국과학기술원 전산학과 석사. 1981년 ~ 1984년 삼성전자 연구원. 1988년 오하이오 주립대학 전산학 박사. 1988년 ~ 1989년 AT&T Bell Labs. 연구원. 1989년 ~ 현재 한국과학기술원 전산학과 교수. 관심분야는 상호연결 네트워크, ATM switch, 병렬 컴퓨터 구조임