웜홀 방식의 네트워크에서 효율적인 다대다 개별적 통신 알고리즘

(Efficient All-to-All Personalized Communication Algorithms in Wormhole-Routed Networks)

김 시 관 † 강 오 한 †† 정 종 인 †††
(Si-Gwan Kim) (Oh-Han Kang) (Jong-In Chung)

요 약 본 논문에서는 웜홀 라우팅 방식을 사용한 2차원 토러스에서 다대다 개별적 통신에 대한 효율적인 알고리즘을 제시한다. 다대다 개별적 통신은 집합체 통신(Collective Communication)의 일종으로 행렬 전이, FFT, 혹은 분산 테이블 검색과 같은 많은 응용 분야에 적용이 되고 있다. 이에 대한 연구는 망의 크기가 2의 멱승 혹은 4의 배수인 경우에 대한 알고리즘이 제시가 되었지만 그 크기가 일반적인 경우에 대해서는 아직은 제안되고 있지 않고 있다. 본 논문에서는 먼저 망의 크기가 2의 배수인 경우에 대한 다대다 개별적 통신에 대한 Double-Hop-2D 알고리즘을 제안한 다음 이 알고리즘을 확장하여 임의의 노드 수에 적합한 2개의 알고리즘을 제안한다. Split-and-Merge 알고리즘은 전체망을 4개의 지역으로 분할하여 각 분할된 영역이 독립적으로 영역별로 다대다 개별적 통신을 수행한 후 그 결과를 다시 결합하는 단계로 구성되어 있다. Modified Double-Hop-2D 알고리즘은 기본이 되는 Double-Hop-2D 알고리즘에서 추가적인 작업을 수행함으로써 다대다 개별적 통신을 수행한다. 마지막으로 망의 크기가 일반적인 경우에 Modified Double-Hop-2D 알고리즘이 Split-and-Merge 알고리즘보다 성능이 우수함을 보인다.

키워드 : 웜홀라우팅, 집합체 통신, 토러스, 다대다 통신

Abstract We present efficient generalized algorithms for all-to-all personalized communication operations in a 2D torus. All-to-all personalized communication, or complete exchange, is at the heart of numerous applications, such as matrix transposition, Fast Fourier Transform(FFT), and distributed table lookup. Some algorithms have been presented when the number of nodes is power-of-2 or multiple-of-four form, but there has been no result for general cases yet. We first present complete exchange algorithm called *Double-Hop-2D* when the number of nodes is in the form of multiple-of-two. Then by extending this algorithm, we present two algorithms for an arbitrary number of nodes. *Split-and Merge* algorithm first splits the whole network into zones. After each zone performs complete exchange, merge is applied to finish the desired complete exchange. By handling extra steps in *Double-Hop-2D* algorithm, *Modified Double-Hop-2D* algorithm performs complete exchange operation for general cases. Finally, we compare the required start-up time for these algorithms.

Key words : Wormhole Routing, Collective Communication, Torus, All-to-All Communication

1. 서 론

멀티컴퓨터[1]는 여러 개의 처리기가 특정한 형태로 연결되어 있는 컴퓨터이다. 각각의 처리기는 공통의 기억장치가 아닌 지역적인 기억장치를 가지고 있다. 그러므로, 처리기간에 정보 교환을 위하여 메시지 전달이 필요하며 이웃하지 않는 처리기들은 하나 이상의 중간 처리기를 통해서만 간접적으로 통신을 할 수 있다.

멀티컴퓨터에서 많이 사용되는 형태는 k-*ary* n-큐브인데 이는 각 n 차원마다 k개의 노드들로 구성되어 있

† 비 회 원 : 강오공과대학교 컴퓨터공학부 교수
sgkim@se.kumoh.ac.kr
†† 종신회원 : 안동대학교 컴퓨터교육과 교수
ohkang@andong.ac.kr
††† 종신회원 : 공주대학교 컴퓨터교육과 교수
jichung@kongju.ac.kr
논문접수 : 2000년 2월 21일
심사완료 : 2003년 3월 31일

다. 메쉬, 토러스 및 하이퍼큐브는 *k-ary n-큐브*의 일종이라 할 수 있다. 2차원 메쉬는 Intel Paragon, Touchstone DELTA과 Caltech Mosaic C에 채용되었고 3차원 메쉬는 J-Machine에서, 3차원 토러스는 CRAY T3D에서 그리고 하이퍼큐브는 nCUBE-2에 채용되었다.

웜홀방식의 라우팅은 Intel Touchstone DELTA, Intel Paragon, MIT J-machine, Caltech MOSAIC 그리고 Cray T3D과 같은 차세대 병렬컴퓨터에 채용되었다. 메시지들이 각각의 처리기에서 저장되었다가 보내어지는 저장후전송방식과 달리 웜홀 방식은 메시지의 머리부분이 입력채널에서 출력채널로 바로 보내어진다. 각각의 처리기에서는 몇 개의 플릿(flit)만 버퍼에 저장이 된다. 메시지의 머리부분을 검사한 뒤 바로 다음 채널이 결정되고 이어 그 채널로 보내진다. 따라서, 이러한 과정에서 메시지가 원천지 노드에서 목적지 노드사이의 채널에 흩어져 있게 된다. 그러므로, 메시지의 마지막 플릿이 원천지 노드를 떠나기 전에 메시지의 첫번째 플릿이 목적지 노드에 도달하는 경우도 있을 수 있다. 이런 과정에서 메시지의 헤더 플릿(header flit)이 불통이 되면(blocked) 메시지의 모든 다른 플릿이 나아갈 수 없고 현재 다른 메시지가 사용하는 채널을 요구하는 모든 다른 메시지들이 불통이 된다.

대부분의 멀티컴퓨터는 단일전송(unicast)만 지원을 한다. 최근에는 집합체 통신(collective communication)[2, 3, 4, 5, 6, 7] 기능을 추가하는 추세인데 이는 같은 메시지가 다수의 원천지에서 다수의 목적지 노드로 전달되는 통신을 말한다. 메시지 전달 기법을 채용한 멀티컴퓨터에서는 부하 조절(load balancing), 사건 동기화(event synchronization), 자료 교환(data exchange)과 같은 여러가지 작업을 위해서 처리기간에 서로 통신을 해야 할 필요가 발생한다. 송수신하는 처리기의 갯수에 따라 통신 형태는 일대일(unicast), 일대다(multicast), 방송형(broadcast), 다대다(all-to-all)로 분류할 수 있다. 또한, 보내야하는 메시지의 내용이 서로 다른 개별적(personalized) 통신과 메시지의 내용이 모두 같은 비개별적(non-personalized) 통신으로 구분된다.

행렬 전이, FFT, 분산 테이블 검색과 같은 분야에 응용이 되는 다대다 개별적 통신(All-to-All Personalized Communication)[8-12]은 아주 중요한 통신 요소 중의 하나라고 할 수 있다. 이러한 형태의 통신에서는 N개의 처리기가 각각 길이는 같지만 내용이 서로 다른 메시지가 자기 이외의 $N-1$개의 처리기로 보내어진다. 따라서, 통신 형태와 통신량이 아주 복잡하다고 할 수 있다. 다대다 개별적 통신은 완전교환(complete exchange)이라고도 부른다.

완전교환 연산에서는 각각의 노드가 자기 이외의 노드들에게 서로 다른 메시지를 보내게 된다. 즉, 모든 노드쌍들은 서로 메시지를 교환하여야 한다. 완전 교환을 구현하는 알고리즘은 직접(direct) 방식과 간접(indirect) 방식으로 분류할 수 있다. 직접 방식에서는 몇몇개의 충돌이 없는 단계를 거쳐서 수행이 된다. 각 단계마다 서로 독립적인 경로를 통하여 메시지들이 원천지에서 목적지로 직접 전달이 된다. 이와 달리 간접 방식에서는 메시지가 목적지로 전달되는 도중에 중간 노드에서 버퍼링되거나 자료들이 재정렬될 수도 있다. 메쉬나 토러스에서는 일반적으로 직접 방식보다 간접 방식이 더 효율적이다[13, 14, 15, 16, 18. 19].

Bokhari[2]와 Sundar[17]는 2차원 메쉬에서의 다대다 개별적 통신에 관한 알고리즘을 제시하였다. Yang[21]은 다단계 상호연결망에서의 다대다 개별적 통신에 관한 알고리즘을 제시하였다. [20]에서는 모든 노드가 링크 충돌을 피하기 위하여 대각선을 따라서 그룹화를 하였다. 이는 임의의 연속된 2개의 노드는 서로 다른 행과 열을 따라서 배치된다는 토러스의 특성을 이용한 것이다. 그러나, 이 알고리즘에서는 망의 크기가 2의 멱승이 되어야 한다는 제한점을 가지고 있어서 망의 크기가 일반적인 경우일 때는 부가적인 단계가 필요하게 된다. 최근 [15]는 [20]와 유사한 알고리즘을 제안하였지만 여기서는 망의 크기가 4의 배수가 되어야 하는 조건을 가지고 있다. 본 논문에서는 먼저 망의 크기가 짝수인 경우에 대한 알고리즘으르 제시한 후 이 알고리즘을 근간으로 하여 망의 크기가 일반적인 경우에 대한 두 개의 알고리즘을 제시한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 완전 교환의 개념에 대해서 설명한 뒤 새로운 알고리즘을 제시하고 3장에서는 2장에서 제시된 알고리즘을 토대로 노드의 갯수가 일반적인 경우로 확장을 시킨다. 4장에서는 복잡도를 분석하고 마지막으로 5장에서 결론을 맺는다.

2. 알고리즘

1절에서는 본 논문에서 제시하는 알고리즘에 대한 여러 가지 필요한 가정들과 용어에 대하여 설명을 한다. 1차원 토러스에 대한 완전교환 알고리즘의 개념을 설명한 뒤 2차원 토러스에 대한 알고리즘으로 확장시킨다. 2절에서는 이차원 알고리즘에 대한 자세한 내용

을 제시하고 망의 크기가 일반적인 경우는 다음 장에서 제시한다.

2.1 기본 사항

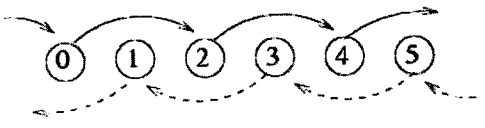
먼저 알고리즘에 필요한 가정은 다음과 같다.

- 웜홀 라우팅 방식이 사용된다.
- 메시지가 전달되기 위해서 중간 노드들에서 버퍼링, 재정렬 및 전달과정이 이루어지는 간접 방식을 가정한다.
- 노드에서 한 번에 하나의 메시지만을 주고 받을 수 있는 단일 출구(one-port) 통신을 가정한다.

완전교환 알고리즘을 구현하는 가장 원시적인 방법(naive method)은 먼저 모든 노드가 X축 방향으로 바로 오른쪽 이웃 노드에게 메시지를 전송한 다음($N-1$의 기동 시간이 소요됨)

모든 노드가 Y축 방향으로 바로 아래쪽 이웃 노드에게 메시지를 전송(N-1의 기동 시간이 소요됨)하는 방법이다. 그러므로 이 방법은 2(N-1) 만큼의 기동시간이 필요하다. 이 방법은 한 번에 한 홉(hop)씩만 메시지의 목적지로 가까워지게 된다. 한 번에 2개의 홉씩 목적지로 메시지를 전송하여 기동 시간을 줄이기 위한 방법을 아래의 예를 통하여 설명한다.

6개의 노드를 가진 1차원 토러스에서의 다대다 개별적 통신은 다음 과정을 거쳐 완료할 수 있다. 기본적인 방법은 2개의 페이즈(phase)로 구성된다. 이 경우 페이즈 1은 2개의 단계(step)로 된다. 이 페이즈에서는 짝수번의 노드들(i)은 $i+2$번의 노드들에게 순방향으로 데이타를 전송한다. 여기에서 '+'와 '-'는 모듈로 연산자를 나타낸다. 그리고, 홀수번 노드 i는 $i-2$ 노드로 역방향으로 데이타를 전송한다. 페이즈 2는 한 개의 단계만으로 구성되어 있는데 이 단계에서는 순방향으로 자기 바로 이웃 노드에게 메시지를 전송한다. 그림 1에서는 6개의 노드를 가진 1차원 토러스의 페이즈 1과 2의 메시지 전달 과정을 나타낸다.



(a) First Phase

(b) Second Phase

그림 1 1차원 토러스에서의 다대다 개별적 통신의 예

NxN 2차원 토러스의 다대다 개별적 통신은 1차원 토러스 경우를 확장한 것으로 볼 수 있다. 1단계에서는 2차원 토러스의 모든 열의 노드들이 각각 독립적으로 1차원 토러스에서와 같이 다대다 개별적 통신을 수행한다. 2단계에서는 2차원 토러스의 모든 행의 노드들이 각각 독립적으로 1차원 토러스에서와 같이 다대다 개별적 통신을 수행한다.

2차원 토러스에서의 각 노드는 $P_{i,j}$ 로 표시한다(여기에서 $0 \le i, j < N$이다). 노드 $P_{i,j}$는 노드 $P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$와 $P_{i,j+1}$에 연결되어 있다. i번째 수평 짝수번 노드 HE_i는 $HE_i = P_{i,0}, P_{i,2}, \ldots, P_{i,N-2}$의 집합으로 정의한다. (i번째 수평 홀수번 노드 HO_i는 $HO_i = P_{i,1}, P_{i,3}, \ldots, P_{i,N-1}$의 집합으로 정의한다. 마찬가지로 i번째 수직 짝수번 노드 VE_i는 $VE_i = \{P_{0,i}, P_{2,i}, \ldots, P_{N-2,i}\}$의 집합으로 정의한다. i번째 수직 홀수번 노드 VO_i는 $VO_i = \{P_{1,i}, P_{3,i}, \ldots, P_{N-1,i}\}$의 집합으로 정의한다. HE_i와 HO_i의 j번째 메시지를 각각 $HE_i[j]$와 $HO_i[j]$로 정의한다. 같은 방법으로 $VE_i[j]$와 $VO_i[j]$도 정의한다.

P와 P′를 x 혹은 y 좌표가 서로 다른 두 개의 노드라고 할 때 P에서 P′로 같은 차원을 따라 메시지를 전송하는 것을 P $d\rightarrow$ P′으로 표시할 수 있다(여기에서 d는 + 혹은 -이다). 만약 d = "+"(혹은 "-")이면 라우팅이 해당 차원을 따라서 순방향(혹은 역방향)으로 일어나는 것을 나타낸다.

제안된 알고리즘에서는 순방향 이중홉(Forward Double Hop)과 순방향 단일홉(Forward Single Hop), 그리고 역방향 이중홉(Backward Double Hop)의 크게 3가지 형태의 메시지 통신 형태로 구성되어 있다.

제안된 알고리즘에서는 3가지 형태의 홉이 있다. **순방향 이중홉**과 **역방향 단일홉**은 짝수번 노드들 중에서 행하여지고 **역방향 이중홉**은 홀수번 노드들 중에서 행하여진다. 2개의 수평 혹은 수직 짝수번 노드에서 발생되는 순방향 이중홉은 순방향 이중 수평홉과 순방향 이중 수직홉의 2가지 형태가 있다. 즉, 각각의 형태는

$FDH_i = HE_i[m] \rightarrow \{HE_i[m+2] \mid m=0, 2, \ldots, N-2\}$,

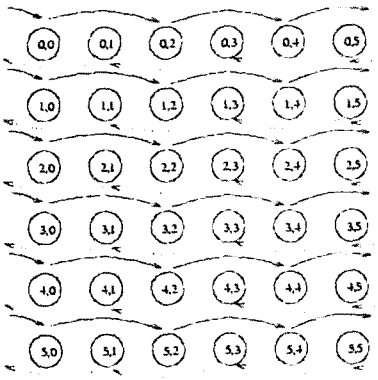
$FDV_i = VE_i[m] \rightarrow \{VE_i[m+2] \mid m=0, 2, \ldots, N-2\}$,

이다.

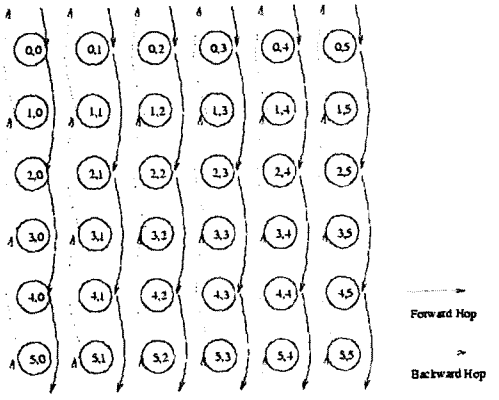
마찬가지로 역방향 이중 수평홉과 역방향 이중 수직홉의 형태는

$BDH_i = HO_i[m+2] \rightarrow \{HO_i[m] \mid m=1, 3, \ldots, N-1\}$,

$BDV_i = VO_i[m+2] \rightarrow \{VO_i[m] \mid m=1, 3, \ldots, N-1\}$,



(a) Double Horizontal Hop



(b) Double Vertical Hop

그림 2 토러스에서 순/역방향 수평/수직홉의 예

이다.

그림 2(a)는 6×6 토러스에서 순방향 이중 수평홉과 역방향 이중 수평홉의 예를 보여준다. 그림 2(b)는 6×6 토러스에서 순방향 이중 수직홉과 역방향 이중 수직홉의 예를 보여준다.

2개의 수평/수직 짝수번 노드에서 발생되는 순방향 단일 수평홉과 순방향 단일 수직홉은 다음과 같다. 즉,

$FSH_i = HE_i[m] \rightarrow \{ HE_i[m+1] \mid m=0, 1, \ldots, N-1 \}$,

$FSV_i = VE_i[m] \rightarrow \{ VE_i[m+1] \mid m=0, 1, \ldots, N-1 \}$,

상기 식 (1), (3) 및 (5)에서 메시지 이동중 임의 메시지의 목적지 노드 X 좌표 값이 현재 노드의 X축 값과 일치하면 이미 그 메시지는 목적지 노드의 X축 방향으로 목적지에 도착하였으므로 그 메시지는 이동할 필요가 없다. 마찬가지로 식 (2), (4) 및 (6)에서는 Y축에 적용이 된다. 다대다 개별적 통신에서는 각각의 노드는 자기 이외의 N^2-1개의 노드에게 고유한 메시지를 전송하여야 한다. 각 $P_{i,j}$ 노드가 자기 노드에게 보내는 1개의 가상 메시지를 포함하여 N^2개의 메시지가 필요하며 이러한 메시지들이 $M_{i,j}[0:N-1, 0:N-1]$ 배열에 한 요소당 하나의 메시지가 저장되어 있다.

본 논문에서 제안하는 알고리즘은 페이즈 X와 페이즈 Y로 구성되어 있다. 다음 절에서 이에 대한 알고리즘을 제시한다.

2.2 Double-Hop-2D 알고리즘

다대다 개별적 전송 알고리즘 Double-Hop-2D는 2개의 페이즈, 페이즈 X와 페이즈 Y로 구성되어 있다. 먼저, 각 $P_{i,j}$ 노드는 $M_{i,j}$ 형태의 메시지를 가지고 있다. 페이즈 X에서는 X 방향을 따라 2개의 홉 혹은 1개의 홉만큼 떨어진 노드들간에서 메시지의 교환이 일어나고 페이즈 Y에서는 마찬가지로 Y 방향을 따라 2개의 홉 혹은 1개의 홉만큼 떨어진 노드들간에서 메시지의 교환이 일어나게 된다.

```
// 페이즈 X:
for Step = 1  to N/2-1 // 페이즈 X의 스테이지 1
  parbegin  // N/2-1개의 단계로 구성. Step은 '단계'를 의미.
    for k=0 to N-1
      parbegin
        Perform FDH_k if k = even.
        Perform BDH_k if k = odd.
      parend
    endfor
  parend
endfor
parbegin  // 스테이지 2는 1개의 단계만 존재.
  for k=0 to N-1
    Perform FSH_k
  endfor
parend

// 페이즈 Y:
for Step=1 to N/2-1  // 페이즈 Y의 스테이지 1
  parbegin  // N/2-1개의 단계로 구성. Step은 '단계'를 의미.
    for k=0 to N-1
      parbegin
        Perform FDV_k if k = even
        Perform BDV_k if k = odd
      parend
    endfor
  parend
endfor
parbegin  // 스테이지 2는 1개의 단계만 존재.
  for k=0 to N-1
    Perform FSV_k
  endfor
parend
```

그림 3 알고리즘 Double-Hop 2D

페이즈 X는 그림 3에서와 같이 2개의 스테이지(stage)로 구성되어 있다. 첫번째 스테이지는 N/2-1개의 단계로 이루어진다. 각 단계에서는 모든 노드들은 2개의 홉만큼 떨어진 노드와 메시지를 송수신한다. 짝수번 노드들은 FDH_i에 의해서 메시지를 주고받으며 홀수번 노드들은 BDH_i에 의해서 메시지를 주고받게 된다. 두 번째 스테이지는 하나의 단계로만 되어 있다. 모든 노드들은 자기 노드보다 하나 큰쪽으로 메시지를 송신한다. 페이즈 X를 수행한 뒤의 상태는 메시지들이 자기가 원하는 목적지의 행 위치와 일치하게 된다. 페이즈 Y는 페이즈 X와 마찬가지로 2개의 스테이지로 되어 있으며 FDV_i와 BDV_i인 점을 제외하고는 페이즈 X와 동일하다. 이 Double-Hop-2D 알고리즘은 그림 3에 나타나 있다.

각 단계별로 메시지를 송수신할 때 중간 노드에서는 어떻게 메시지가 정렬(혹은 이동)이 되는지를 고려하여야 한다. 이러한 과정은 아래와 같다. 페이즈 X에서 FDH_i와 BDH_i에 관련된 모든 노드들의 메시지 정렬 알고리즘은 그림 4와 그림 5에 설명되어 있다. 마찬가지로, 페이즈 Y에서 FDV_i와 BDV_i에 관련된 모든 노드들의 메시지 정렬 알고리즘은 그림 6과 그림 7에 설명되어 있다. 이 그림에서 $C_X(P_{i,j})_k$는 FDH_k의 경우 부분배열 $M_{i,j}[0 : N-1, 2k : 2(k+1)-1]$과 BDH_k인 경우 부분배열 $M_{i,j}[0 : N-1, 2k+1 : 2k+2]$를 표시한다. $C_Y(P_{i,j})_k$는 FDH_k의 경우 부분배열 $M_{i,j}[2k : 2(k+1)-1, 0 : N-1]$과 BDH_k인 경우 부분배열 $M_{i,j}[2k+1 : 2k+2, 0 : N-1]$를 표시한다.

각 단계가 수행된 이후 $M_{i,j}$의 각 요소 값들이 일부 변하게 되므로 이 변화되는 상태를 $M_{i,j}^{\langle s,p\rangle}$로 표기한다. 여기서 s = 1, 2, ..., N/2-1, 즉, 단계를 나타내고, p는 페이즈 X 혹은 페이즈 Y를 의미한다. 이는 Tseng[20]의 표기법을 토대로 한 것이다. 또, $M_{i,j}$의 초기값을 $M_{i,j}^{\langle 0,X\rangle}$로 표기하고 페이즈 X를 마친 후의 $M_{i,j}$의 상태를 $M_{i,j}^{\langle 0,Y\rangle}$로 표기한다.

페이즈 X의 첫번째 스테이지 수행 후 부분배열 $M_{i,j}^{\langle N/2-1,X\rangle}[0:N-1, j+2:j+N-1]$의 상태는 임의의 $P_{i,j} \in HE_i$에 대하여 $\{P_{i,j+v} \mid 2 \le v < N\}$가 목적지인 $P_{i,j-2}$로부터 전송된 메시지를 가지고 있게 된다. 또,

```
FDH_i와 관계된 각 P_{x,y} 노드마다
 for Step=1 to N/2-1  // '+'는 모듈로 N/2 연산
   for k=0 to N/2-1
     parbegin
       C_X(P_{x,y})_{k+1} ← C_X(P_{x,y+2})_{k-Step+1}
       for j=2 to N/2-Step
         C_X(P_{x,y})_{k+j} ← C_X(P_{x,y+2})_{k+j}
       endfor
     parend
   endfor
 endfor
```

그림 4 페이즈 X에서의 FDH_i의 정렬

```
BDH_i와 관계된 각 P_{x,y} 노드마다
 for Step=1 to N/2-1  // '+'는 모듈로 N/2 연산
   for k=0 to N/2-1
     parbegin
       C_X(P_{x,y+3})_{k+1} ← C_X(P_{x,y+1})_{k-Step}
       for j=2 to N/2-Step
         C_X(P_{x,y+3})_{k+j} ← C_X(P_{x,y+1})_{k+j}
       endfor
     parend
   endfor
 endfor
```

그림 5 페이즈 Y에서의 FDH_i의 정렬

```
FDH_i와 관계된 각 P_{x,y} 노드마다
 for Step=1 to N/2-1  // '+'는 모듈로 N/2 연산
   for k=0 to N/2-1
     parbegin
       C_X(P_{x,y})_{k+1} ← C_X(P_{x+2,y})_{k-Step+1}
       for j=2 to N/2-Step
         C_X(P_{x,y})_{k+j} ← C_X(P_{x+2,y})_{k+j}
       endfor
     parend
   endfor
 endfor
```

그림 6 페이즈 X에서의 BDH_i의 정렬

```
BDH_i와 관계된 각 P_{x,y} 노드마다
 for Step=1 to N/2-1  // '+'는 모듈로 N/2 연산
   for k=0 to N/2-1
     parbegin
       C_X(P_{x+3,y})_{k+1} ← C_X(P_{x+1,y})_{k-Step}
       for j=2 to N/2-Step
         C_X(P_{x+3,y})_{k+j} ← C_X(P_{x+1,y})_{k+j}
       endfor
     parend
   endfor
 endfor
```

그림 7 페이즈 Y에서의 BDH_i의 정렬

부분배열 $M_{i,j}^{\langle N/2-1,X\rangle}[0{:}N-1, j-2{:}j-N+1]$의 상태는 임의의 $P_{i,j} \in HO_i$에 대하여 { $P_{i,j-v}$ | 2 ≤ v < N}가 목적지인 $P_{i,j+2}$로부터 전송된 메시지를 가지고 있게 된다. 마찬가지로 VE_i와 VO_i에 대해서도 적용이 된다.

지금까지의 결과는 다음의 소정리들로 표시할 수 있다. 여기서 p→q는 p에 있는 메시지가 q의 위치로 이동함을 나타낸다. 소정리 1과 2는 각 메시지가 페이즈 X에서 짝수 및 홀수번째 노드에서 어떻게 중간 노드를 거쳐서 이동되었는가를 보여주고 있다.

소정리 1 원천지 노드가 (s_1,s_2)이고 목적지 노드가 (d_1,d_2)인 $M_{i,j}^{\langle k,X\rangle}[0{:}N-1,0{:}N-1]$ 경우를 고려한다. 페이즈 X이고 첫번째 스테이지 과정 중 k 단계에서 짝수번 노드들에 대해 $M_{i,j}^{\langle k,X\rangle}[a,b]$ (단, 0 ≤ a, b < N)은 다음과 같은 값을 가지게 된다.

$M_{i,j}^{\langle k,X\rangle}[a,b] \rightarrow M_{i,j}^{\langle k+1,X\rangle}[a,b]$, 만약 b+j= s_2+d_2 이면 (7)

→ $M_{i,j+2}^{\langle k+1,X\rangle}[a,s_2+d_2-(j+2)]$, 만약 b=j+1 혹은 j+2 이면

→ $M_{i,j+2}^{\langle k+1,X\rangle}[a,b]$, 기타 경우

, 단, k=1,2, ..., N/2-2.

페이즈 X이고 두번째 스테이지 과정 중 k 단계에서 모든 노드들에 대해

$M_{i,j}^{\langle k,X\rangle}[a,b]$ (단, 0 ≤ a, b < N)은 다음과 같은 값을 가지게 된다.

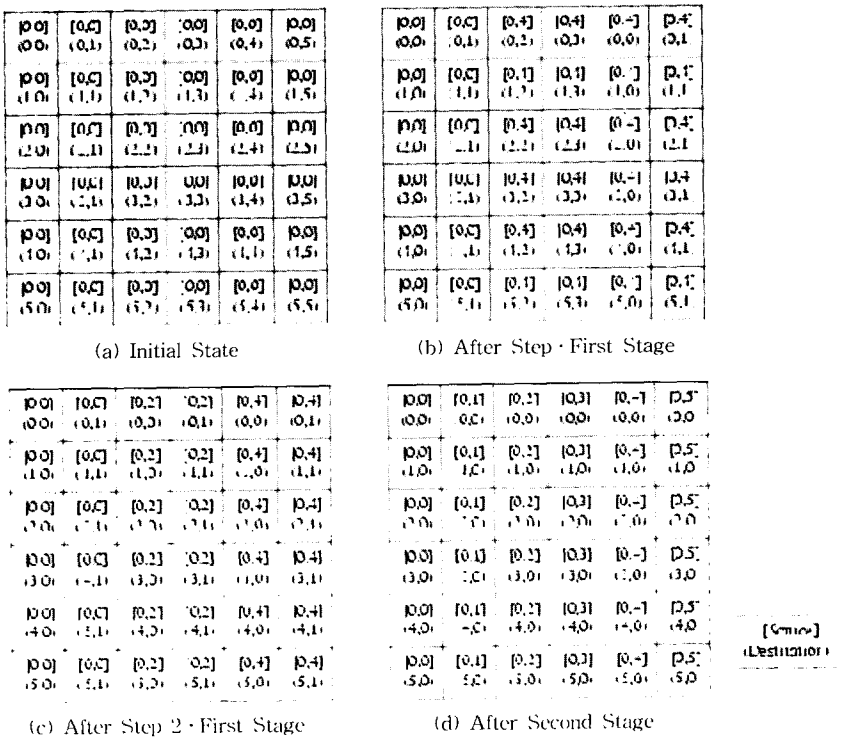
$M_{i,j}^{\langle k,X\rangle}[a,b] \rightarrow M_{i,j}^{\langle k+1,X\rangle}[a,b]$, 만약 b가 짝수이면 (8)

→ $M_{i,j+1}^{\langle k-1,X\rangle}[a,b-1]$, 만약 b가 홀수이면

, 단, k=N/2-1.

증 명 케이스별로 증명을 한다. 페이즈 X의 첫번째 스테이지에서 :

- 경우 1 : 만약 b+j= s_2+d_2이면 이 경우는 메시지가 이미 자기의 목적지 노드를 찾은 경우를 의미하므로 더 이상 중간 노드로 전달할 필요가 없다.
- 경우 2 : 만약 b=j+1 혹은 j+2이면 이 경우는 메시지가 원하는 목적지를 찾았음을 의미한다. 따라서, 메시지의 적절한 위치를 정하기 위하여 재정렬이 필요하다.



(a) Initial State (b) After Step · First Stage

(c) After Step 2 · First Stage (d) After Second Stage

그림 8 6×6 토러스(페이즈 X)에서의 라우팅 예

• 경우 3 : 이 외의 경우는 메시지가 자기 목적지에 도달하기 위하여 중간 노드를 더 거쳐야 함을 의미한다. 따라서, 색인 $M_{i,j}$는 $M_{i,j+2}$로 바뀌어져야 한다.

다음으로 페이즈 X의 두번째 스테이지에 대하여 :

페이즈 X의 첫번째 스테이지를 수행 한 뒤 목적지가 짝수인 모든 메시지들은 원하는 행 위치에 도달하게 된다. 그러므로, 이러한 경우는 페이즈 X의 두번째 스테이지에서는 더 이상 이동할 필요가 없다. 그러나, 목적지가 홀수인 모든 메시지들은 원하는 행 위치보다 한 홉 만큼 떨어져 위치하고 있다(식 (8)에 준함).

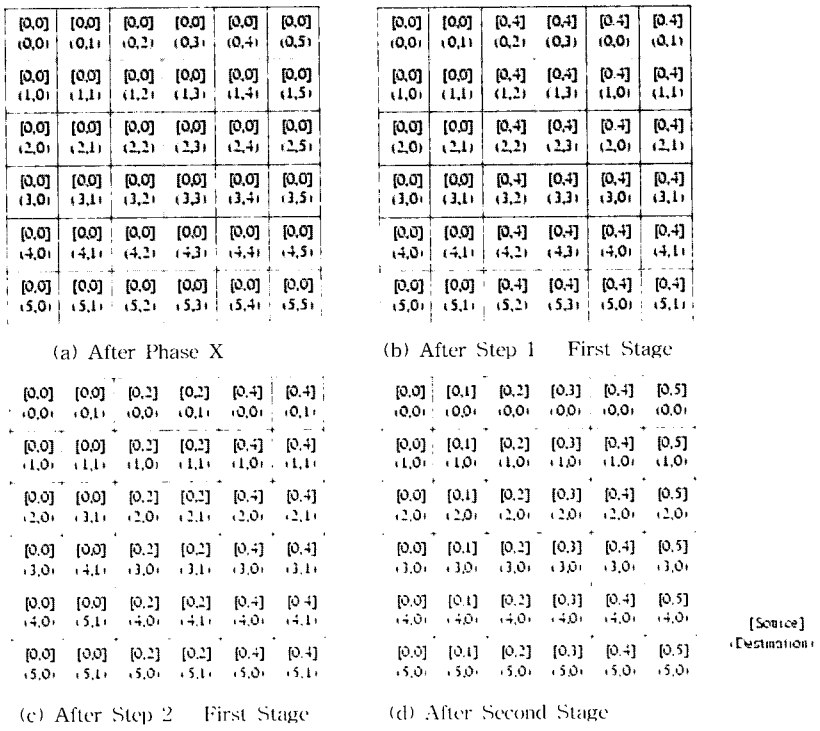
6×6 토러스에서 노드 $P_{0,0}$에서 페이즈 X에 대한 라우팅의 예가 그림 8에 나타나 있다. 그림 8(a)는 메시지 $M_{0,0}$의 초기 상태를 나타내는 것으로 각 상자는 하나의 메시지이며 [a, b] 및 (c, d)는 원천지 노드 $P_{a,b}$와 목적지 노드 $P_{c,d}$임을 나타낸다. 그림 8(b)에서는 0행과 1행의 메시지들은 그 전 상태(즉, 그림 8(a))를 유지하고 있으며 2, 3, 4, 5행의 메시지들은 원천지가 $P_{0,4}$에서 수신되는 메시지로서 2행과 3행의 메시지들은 목적지를 향해서 이동 중이며, 4행과 5행의 메시지들은 자기 목적지에 도착한 메시지임을 알 수 있다.

그림 8(c)에서는 0, 1, 4, 5행의 메시지들은 그 전 상태(즉, 그림 8(b))와 동일한 위치를 유지하고 있으며 2, 3행의 메시지들은 원천지가 $P_{0,2}$이며 이는 $P_{0,4}$를 거쳐서 수신되는 메시지로서 이 메시지들은 목적지를 향해서 이동 중이다. 그림 8(d)에서는 1, 3, 5행의 메시지들이 바로 왼쪽에 인접한 $P_{0,1}$, $P_{0,3}$, $P_{0,5}$로부터 수신이 된다. 이 상태가 되면 모든 메시지들의 X 좌표는 목적지와 일치하게 된다. 마찬가지 방법으로 그림 9와 같이 Y방향으로 알고리즘을 실행하면 원하는 완전교환 통신이 이루어진다.

소정리 2 원천지 노드가 (s_1, s_2) 이고 목적지 노드가 (d_1, d_2)인 $M_{i,j}^{\langle k,X\rangle}$[0:N-1, 0:N-1] 경우를 고려한다. 페이즈 X이고 첫번째 스테이지 과정 중 k 단계에서 홀수번 노드들에 대해

$M_{i,j}^{\langle k,X\rangle}$[a,b] (단, 0 ≤ a, b < N)은 다음과 같은 값을 가지게 된다.

$$M_{i,j}^{\langle k,X\rangle}[a,b] \rightarrow M_{i,j}^{\langle k+1,X\rangle}[a,b], \text{ 만약 } b+j=s_2+d_2 \text{ 이면} \quad (9)$$



(a) After Phase X (b) After Step 1 First Stage

(c) After Step 2 First Stage (d) After Second Stage

그림 9 6×6 토러스(페이즈 Y)에서의 라우팅 예

→ $M_{i,j+2}^{\langle k+1,X\rangle}[a, s_2+d_2-(j-2)]$,
만약 b=j-1 혹은 j-2 이면
→ $M_{i,j-2}^{\langle k+1,X\rangle}[a, b]$, 기타 경우,
단, k=1,2, ..., N/2-2.

$M_{i,j}^{\langle k,X\rangle}[a, b]$ → $M_{i,j}^{\langle k+1,X\rangle}[a, b]$, 만약 b가 홀수이면 (10)
→ $M_{i,j+1}^{\langle k+1,X\rangle}[a, b-1]$,
만약 b가 짝수이면,
단, k=N/2-1.

증명 소정리 1과 같은 방법으로 증명이 가능하다.

페이즈 X를 수행한 후 소정리 3과 소정리 4는 각각의 페이즈 Y를 수행하면서 각각의 노드들에서 메시지들이 어떻게 최종 목적지로 전달이 되는지를 보여준다. 6×6 토러스에서 노드 $P_{0,0}$의 내용이 페이즈 Y에서 어떤 단계를 거쳐 변하는지를 그림 9에서 보여준다.

소정리 3 원천지 노드가 (s_1, s_2) 이고 목적지 노드가 (d_1, d_2)인 $M_{i,j}^{\langle k,Y\rangle}$[0:N-1, 0:N-1] 경우를 고려한다. 페이즈 Y이고 첫번째 스테이지 과정 중 k 단계에서 짝수번 노드들에 대해
$M_{i,j}^{\langle k,Y\rangle}$[a,b] (단, 0 ≤ a, b < N)은 다음과 같은 값을 가지게 된다.

$M_{i,j}^{\langle k,Y\rangle}[a, b]$ → $M_{i,j}^{\langle k+1,Y\rangle}[a, b]$, 만약 a+i= s_1+d_1 이면 (11)
→ $M_{i+2,j}^{\langle k+1,Y\rangle}[s_1+d_1-(i+2), b]$,
만약 a=i+1 혹은 i+2 이면
→ $M_{i+2,j}^{\langle k+1,X\rangle}[a, b]$, 기타 경우,
단, k=1,2, ..., N/2-2

$M_{i,j}^{\langle k,Y\rangle}[a, b]$ → $M_{i,j}^{\langle k+1,Y\rangle}[a, b]$, 만약 a가 짝수이면 (12)
→ $M_{i+1,j}^{\langle k+1,Y\rangle}[a, b-1]$,
만약 a가 홀수이면,
단, k=N/2-1

증명 소정리 1과 같은 방법으로 증명이 가능하다.

소정리 4 원천지 노드가 (s_1, s_2)이고 목적지 노드가 (d_1, d_2)인 $M_{i,j}^{\langle k,Y\rangle}$[0:N-1, 0:N-1] 경우를 고려한다. 페이즈 Y이고 첫번째 스테이지 과정 중 k 단계에서 홀수번 노드들에 대해 $M_{i,j}^{\langle k,Y\rangle}$[a,b] (단, 0 ≤ a, b < N)은 다음과 같은 값을 가지게 된다.

$M_{i,j}^{\langle k,Y\rangle}[a, b]$ → $M_{i,j}^{\langle k+1,Y\rangle}[a, b]$,
만약 a+i= s_1+d_1 이면 (13)
→ $M_{i-2,j}^{\langle k+1,Y\rangle}[s_1+d_1-(i+2), b]$,
만약 a=i+1 혹은 i+2 이면
→ $M_{i-2,j}^{\langle k+1,X\rangle}[a, b]$, 기타 경우,
단, k=1,2, ..., N/2-2.

$M_{i,j}^{\langle k,Y\rangle}[a, b]$ → $M_{i,j}^{\langle k+1,Y\rangle}[a, b]$, 만약 a가 짝수이면 (14)
→ $M_{i+1,j}^{\langle k+1,Y\rangle}[a-1, b]$,
만약 a가 홀수이면,
단, k=N/2-1.

증명 소정리 1과 같은 방법으로 증명이 가능하다.

정리 Double-Hop-2D 알고리즘은 다대다 개별적 통신을 수행한다

증명 소정리 1부터 소정리 4까지로부터 증명이 가능하다.

3. 일반적인 경우로 확장

본 절에서는 망의 크기가 일반적일 때의 경우를 다룬다. 가장 간단하고도 원시적인 방법은 1차원 토러스인 경우 (N-1) 단계를 사용하여 메시지를 자기 이웃 노드로 전달하는 방법이다. 이를 이용하여 2차원 토러스로 확장할 수 있다. 그러나, 이러한 방법의 큰 단점은 개시 지연 시간을 고려하지 않은 점이다. 개시 지연 시간을 줄이기 위하여 제시된 Double-Hop-2D을 확장시키는 것이다. 확장시키는 방법은 2가지 알고리즘으로 설명될 수 있다.

Split-and-Merge 알고리즘은 전체 망을 짝수개로 구성된 4개의 영역으로 나눈다. 나누어진 각 영역별로 독립적으로 Double-Hop-2D 알고리즘을 적용시켜 다대다 개별적 통신을 수행한 후 그 결과를 통합하는 단계를 거쳐 원하는 다대다 개별적 통신을 완료한다. Modified Double-Hop-2D 알고리즘은 Double-Hop-2D 알고리즘에서 각 단계별로 메시지의 이동을 조금 수정함으로써 다대다 개별적 통신을 완료한다.

3.1 Split-and-Merge 알고리즘

Split-and-Merge 알고리즘은 분리(split) 단계와 병합(merge) 단계의 두단계로 구성되어 있다. 이 알고리즘에서는 임의의 노드에서 목적지가 2개 이상인 경우에 해당되는 멀티캐스트를 할 때 U-mesh[8] 알고리즘을 사용하는데 이 알고리즘은 목적지들을 노드의 번호 크기 순서대로 배치한 뒤 트리 형태로 메시지를 전송하면 충돌이 없이 메시지를 전달할 수 있는 기법이다.

분리 단계에서는 그림 10과 같이 전체망을 4개의 영역으로 분리시킨다. 영역 4를 제외한 모든 지역들의 노드 갯수가 2의 배수이므로 각 영역은 독립적으로 영역

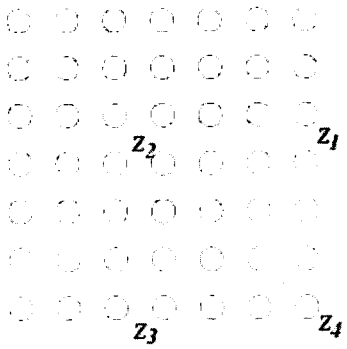


그림 10 7×7 토러스에서 전체망을 분리시킨 후의 영역 모습

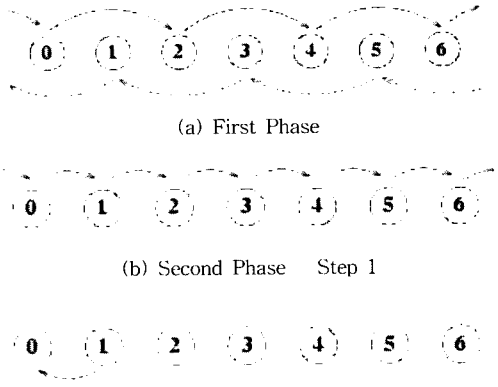
2는 Double-Hop-2D 알고리즘을 영역 1과 3은 1차원 토러스 경우의 다대다 개별적 통신을 수행한다. 영역 4는 단 하나의 노드만 가지고 있다. 그림 10에서와 같이 7×7 토러스 경우 영역 2는 6×6의 2차원 토러스이고 영역 1과 3은 노드가 6개인 1차원 토러스의 형태로 분리될 수 있다.

분리 단계에서는 영역 4만 제외하고 모든 지역이 각기 독립적으로 다대다 개별적 통신을 수행한다. 영역 2에서는 짝수번 노드들에 대해서 순방향 연산이 이루어질 때 (N-2)번째의 행(그리고 열)이 목적지인 메시지들은 0번째 행(그리고 열)로 전달한다. 또, 홀수번 노드들에 대해서 순방향 연산이 이루어질 때 0번째의 행(그리고 열)이 목적지인 메시지들은 N-2번째 행(그리고 열)로 전달한다. 영역 1에서는 목적지가 영역3과 4로 향하는 모든 메시지들은 (N-2)번째 열로 전달한다. 영역 3에서는 목적지가 영역1과 4로 향하는 모든 메시지들은 (N-2)번째 행으로 전달한다.

통합 단계에서는 다대다 개별적 통신을 완료하기 위하여 모든 영역에 대하여 부가적인 작업이 필요하다. 먼저, 영역 3의 (N-2)번째 노드에서 영역 4가 목적지인 모든 메시지들은 영역 4로 전달된다. 그런 후, 영역 1과 영역 2의 (N-2)번째 노드에서 영역 3과 영역 4가 목적지인 모든 메시지들(또한 이와 반대인 경우도 해당됨)은 U-mesh[8] 알고리즘을 사용하여 영역 1과 영역 3으로 전달된다. 마지막으로 영역 2와 영역 3의 (N-2)번째 노드에서 영역 1과 영역 4가 목적지인 모든 메시지들(또한 이와 반대인 경우도 해당됨)은 U-mesh [8] 알고리즘을 사용하여 영역 2와 영역 3으로 전달된다.

3.2 Modified Double-Hop-2D 알고리즘

Modified Double-Hop-2D 알고리즘에서는 메시지가



(a) First Phase

(b) Second Phase Step 1

(c) Second Phase - Step 2

그림 11 7개의 노드를 가진 토러스에서의 다대다 개별적 통신의 예

2개의 페이즈를 거쳐서 목적지에 전달이 된다. 1차원 토러스인 경우를 통하여 그 개념을 살펴본다. 첫번째 페이즈에서는 모든 짝수번 노드 i는 i+2인 노드로 메시지를 순방향으로 전달한다. 단, 목적지가 노드 N-1인 메시지는 노드 0로 보낸다. 이 작업은 7개의 노드를 가진 토러스인 경우 3개의 단계를 필요로 한다. 그리고, 모든 홀수번 노드 i는 i-2인 노드로 메시지를 역방향으로 전달한다. 이 작업은 7개의 노드를 가진 토러스인 경우 2개의 단계를 필요로 한다. 기본 알고리즘인 Double-Hop-2D와 다른 점은 역방향으로 메시지를 전달할 때 홀수번 노드인 경우에 한해서 목적지가 노드 0로 향하는 메시지들이 노드 1로 전달되는 점이다.

두번째 페이즈는 2개의 스테이지로 구성되어 있다. 첫번째 스테이지에서는 기본 알고리즘인 Double-Hop-2D와 같이 모든 노드 i는 노드 i+1로 메시지가 전달된다. 두 번째 스테이지에서는 노드 1만이 노드 0이 목적지인 메시지만이 전달된다. 그림 11에서는 7개의 노드를 가진 토러스에서 Modified Double-Hop-2D 알고리즘을 적용하여 다대다 개별적 교환 통신의 과정을 보여주는 예이다. 이와같이 1차원 토러스에서의 개념을 확장하여 2차원 토러스에도 쉽게 적용할 수 있다.

4. 복잡도 분석

본 절에서는 기동시간(start-up time)과 전송시간(data transmission time)의 관점에서 제시된 알고리즘에 필요한 복잡도를 분석한다.

- 기동시간: 이 시간은 채널간에 통신을 위해 기동에 소요되는 시간을 말한다. 페이즈 X의 첫번째 스테이지는

N/2-1의 단계가 필요하고 페이즈 X의 두번째 스테이지는 1개의 단계만 요구된다. 따라서, 본 알고리즘에서는 N개의 단계가 소요된다. 그러므로, Double-Hop-2D 알고리즘의 총 기동시간은 Nt_s이다. 여기에서 t_s는 하나의 메시지를 전송하는데 필요한 기동 시간을 나타낸다. 원시적 방법(naive method)의 기동 시간은 $(2(N-1))\,t_s$이다. Split-and-Merge와 Modified Double-Hop-2D 알고리즘의 기동시간은 각각 $(N+2\log(N-1))\,t_s$와 $2(\lceil N/2\rceil+1)\,t_s$이다. 표 1에서 지금 설명한 망의 크기가 제한이 없는 경우의 3가지 알고리즘의 기동시간을 비교하였다.

표 1 망의 크기가 제한이 없는 다대다 개별적 통신 알고리즘의 기동시간 비교

망의 크기	원시적 방법	Split and Merge	Modified Double Hop 2D	Tseng[20]
7×7	12	13	10	12
11×11	20	19	14	20
15×15	28	23	18	20
33×33	64	45	36	68
63×63	124	75	66	68
129×129	256	145	132	260
255×255	508	271	258	260

만약 망의 크기가 2의 멱승 혹은 4의 배수일 때는 제안된 알고리즘, [20], [15]와 [16]의 알고리즘의 성능은 거의 동일하지만 망의 크기가 일반적인 경우(즉, 2의 멱승이나 4의 배수가 아닐 때) [20], [15]와 [16]의 알고리즘은 망의 크기를 일반적인 경우로 적용을 시킨다면 전반적인 복잡도는 약 4배 이상으로 높아지게 된다. 왜냐하면, [20]와 [15]의 알고리즘은 2의 멱승이나 4의 배수에 맞도록 기존의 노드가 가상의 노드로 동작하도록 설정해야 하기 때문에 많은 수의 메시지 경로(disjoint path)들의 충돌(conflict)로 부가적인 메시지 경로 생성의 기동 시간이 증가하기 때문이다.

- 데이타 전송 시간: 이는 망을 통하여 실제 메시지가 전송되는데 소요되는 시간을 말한다. Double-Hop-2D 알고리즘의 경우를 고려한다. 페이즈 X의 첫번째 스테이지에서 $\sum_{i=1}^{N/2-1} N(N-2i)$개의 메시지가 두개의 노드간에서 전송이 된다. 페이즈 X의 두번째 스테이지에서는 단 한 개의 시지가 두개의 노드간에서 전송이 된다. 이는 페이즈 Y에도 똑같이 적용된다. 그러므로, 총 전송 시간은 하나의 메시지가 전송되는 시간을 t_d라고 할 때 $(N^2(N-2)/2+2)\,t_d$이다.

5. 결 론

본 논문에서는 토러스 형태를 가진 대용량 병렬 컴퓨터에서 망의 크기에 제한이 없는 다대다 개별적 통신 알고리즘을 제안하였다. 먼저 기본이 되는 Double-Hop-2D 알고리즘은 망의 크기가 짝수인 경우로 제한하였고 이 알고리즘을 확장시킴으로써 망의 크기에 제한이 없는 2가지 알고리즘을 제안하였다. Split-and-Merge 알고리즘은 전체 망을 4개의 영역으로 분할한 뒤 각 영역이 기본이 되는 Double-Hop-2D 알고리즘을 독립적으로 수행하고 그 중간 결과를 통합시킴으로써 원하는 다대다 개별적 통신 효과를 얻는다. Modified Double-Hop-2D 알고리즘은 기본이 되는 Double-Hop-2D 알고리즘을 근간으로하여 각 단계마다 부가적인 작업을 행함으로써 망의 크기가 제한이 없는 다대다 개별적 통신 알고리즘을 완료한다.

제안된 알고리즘은 기존의 알고리즘이 망의 크기가 2의 멱승 혹은 4의 배수등으로 제안하고 있다는 점과 비교할 때 효율적임을 알 수 있다. 그리고, 제안된 Modified Double-Hop-2D 알고리즘은 복잡도 측면에서 볼 때 Split and Merge 알고리즘보다 성능이 우수하다는 점을 알 수 있다. 또한, 제안된 알고리즘이 2차원 경우에만 설명이 되었지만 3차원 토러스인 경우도 용이하게 확장시킬 수 있음을 알 수 있다.

참 고 문 헌

[1] W.C. Athas and C.L. Seitz, "Multicomputers: Message Passing Concurrent Computers," *IEEE Computers*, Vol. 21, No. 8, pp.9 24, Aug. 1988.

[2] S. Bokhari and H. Berryman, "Complete exchange on a circuit switched mesh," *Proc. of the 1992 Scalable High Performance Computing Conference*, pp.300 306, 1992.

[3] J. Li and M. Chen, "Compiling communication efficient programs for massively parallel machines," *IEEE Tran. on Parallel and Ditributed Systems*, vol. 2, pp. 361 375, July 1991.

[4] X. Lin and L.M. Ni, "Multicast Communications in Multicomputer Networks," *Proc. of Int'l Conf. on Parallel Processing*, 1990, Vol. III, pp.114 118.

[5] X. Lin and L. Ni, "Multicast communication in multicomputer networks," *IEEE Tran. on Parallel and Ditributed Systems*, vol. 4, no. 10, pp. 1104 1117, October 1993.

[6] P.K. McKinley, H. Xu, and L.M. Ni, "Efficient Communication Services for Scalable Architectures," Technical Report, MSU CPS ACS-58,

Dept. of Com. Sci., Michigan State University, East Lansing, MI, Apr. 1992.

[7] P.K. McKinley, Y.J. Tsai and D.F. Robinson, "A Survey of Collective Communication in Wormhole Routed Massively Parallel Computers," Technical Report, MSU CPS 94 35, Michigan State University, June 1994.

[8] P. K. McKinley, H. Xu, A. H. Esfahanian, and L. M. Ni. "Unicast based Multicast Communication in Wormhole routed Networks," *IEEE Transactions on Parallel and Distributed Systems*, 5(12):1252 1265, Dec 1994.

[9] V. Kumar, A. Grama, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Redwood City, CA: Benjamin/Cummings, 1994.

[10] Message Passing Interface Forum, "Document for standard message passing interface," Technical Report CS 93 214, University of Tennessee, Nov. 1993.

[11] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Computer*, Vol. 26, No. 2, pp.62 76, Feb. 1993.

[12] B. Nitzberg and V. Lo, "Distributed shared Memory: A survey of issues and algorithms," *IEEE Computers*, Vol. 24, pp.52 60, Aug. 1991.

[13] D.A. Reed and R.M. Fujimoto, *Multicomputer Networks: Message Based Parallel Processing*, MIT Press, Cambridge, MA, 1987.

[14] D.F. Robinson, D. Judd, P.K. McKinley, and B.H.C. Cheng, "Efficient Collective Data Distribution in All Port Wormhole Routed Hypercubes," *Proc. of Supercomputing '93*, Nov. 1993, pp.792 801.

[15] Y. Suh and S. Yalamanchili, "Efficient Algorithms for Complete exchange in 2D Tori," *Proc. of the 9th IASTED Int'l Conference Parallel and Distributed Computing and Systems*, pp.113 119, 1997.

[16] Y. Suh and Kang G. Shin, "All to All Personalized Communication in Multidimensional Torus and Mesh Networks," *IEEE Tran. on Parallel and Ditributed Systems*, Vol. 12, No. 1, pp. 38 59, Jan. 2001.

[17] N. Sundar, D. Jayasimha, D. Panda, and P. Sadayappan, "Complete exchange in 2D Meshes," *Proc. of the 1994 Scalable High Performance Computing Conference*, pp.406 413, 1994.

[18] R. Thakur and A. Choudhary, "All to all communication on meshes with wormhole routing," *Proc. of the 1994 International Parallel Processing Symposium*, pp.561 565, 1994.

[19] Y.C. Tseng and S. Gupta, "All-to All Personalized Communication in a Wormhole Routed Torus," *IEEE Tran. on Parallel and Ditributed Systems*, Vol. 7, No. 5, pp.498 505, May 1996.

[20] Y.C. Tseng, T.H. Lin, S. Gupta and D.K. Panda, "Bandwidth Optimal Complete Exchange on Wormhole Routed 2D/3D Torus Networks: A Diagonal Propagation Approach," *IEEE Tran. on Parallel and Distributed Systems*, Vol. 8, No. 4, pp. 380 396, Apr. 1997.

[21] Yuanyuan Yang and Jianchao Wang, "Optimal All to All Personalized Exchange in Self Routable Multistage Networks," *IEEE Tran. on Parallel and Distributed Systems*, Vol. 11, No. 3, pp. 261 274, Mar. 2000.

김 시 관

1982년 경북대학교 전자공학과 졸업 1984년 한국과학기술원 전산학과 석사 1984년~1995년 삼성전자, LG정보통신 2000년 한국과학기술원 전산학과 박사 2002년~현재 금오공대 컴퓨터공학부 교수. 관심분야는 컴퓨터구조, 병렬처리, 이동컴퓨팅등

강 오 한

1982년 경북대학교 전자계열 전산모듈 졸업. 1984년 한국과학기술원 전산학과 석사. 1992년 한국과학기술원 전산학과 박사. 1984년~1994년 (주) 큐닉스컴퓨터 1994년~현재 안동대학교 컴퓨터교육과 교수. 관심분야는 병렬처리, 스케줄링, WBI

정 종 인

1981년 경북대학교 전자공학과(전산전공) 학사. 1985년 경북대학교 전자공학과(전산전공) 석사. 1995년 서강대학교 전자계산학과 박사. 1999년~2000년 USC Dept. of EE post-doc. 1985년~1997년 우송공업대학 전산과 부교수. 1997년~현재 공주대학교 컴퓨터교육과 부교수. 관심분야는 병렬처리 구조, 정보보호, 컴퓨터네트워크 등