論文 96-21-5-9

Bisectional 상호연결 네트워크에서 하이퍼큐브의 구현과 경로배정 알고리즘

正會員 최 창 훈*, 정 영 호**, 김 성 천*

An Implementation of Hypercube with Routing Algorithm in Bisectional Interconnection Network

Chang-Hoon Choi*, Young-Ho Jeong**, Sung-Chun Kim* *Regular Members*

요 약

병렬 처리 컴퓨터 시스템을 구성하는 기본 네트워크는 많은 사용자의 요구에 따라 그 내부에 여러 위상을 포함할 수 있는 성질을 갖도록 요구된다. 이러한 성질을 만족하는 네트워크로서 bisectional 상호연결 네트워크가 있으며, 이 네트워크는 여러 위상으로 최적화되어 적용될 수 있다. 또한, 근래에는 메시지 전달 다중 컴퓨터 시스템으로 하이퍼큐브에 큰 관심이 있기 때문에 bisectional 네트워크에서 하이퍼큐브의 구현은 그만큼 중요하다고 할 수 있다. 본 논문에서는 응용이 매우 자유로운 bisectional 네트워크에서 하이퍼큐브를 구현시키고, 여기에 필요한 경로배정 알고리즘과 방송 알고리즘을 제안한다. 따라서 기존의 bisectional 네트워크가 가지는 위상인 선형 배열, 완전 이진 트리, 메쉬 구조 뿐만 아니라 하이퍼큐브를 구현시킴으로써, bisectional 네트워크가 범용의 슈퍼 컴퓨터 통신 구조로서 활용될 수 있게 하였다. 본 논문에서 제안하는 bisectional 네트워크상의 하이퍼큐브 경로배정 알고리즘과 방송 알고리즘은 기존의 하이퍼큐브에서 제공하는 각 알고리즘의 성질을 그대로 수용하는 범용 알고리즘이다.

ABSTRACT

On demand of many users, basic networks of a parallel computer system are required to have a property that can embed various topologies. Bisectional interconnection network is known to satisfy this property, and it can embed various topologies optimally. Nowadays one is very interested in the hypercube as a message passing multicomputer system, so it is very important to implement a hypercube in bisectional network. In this paper, a hypercube is implemented in a versatile bisectional network, and its routing and broadcasting algorithm are

*서강대학교 공과대학 전자계산학과
**삼성전자(주) 기술총괄 근무
論文番號:9447
接受日字:1994年 2月 15日

proposed. Conventional bisectional network can accomodate linear array, complete binary tree and mesh structure as its topology. Now hypercube is implemented to be utilized as a general purpose supercomputer communication architecture. The proposed routing and broadcasting algorithm embedded in bisectional network are general purpose algorithms which satisfy property of conventional hypercube.

I. 서 론

VLSI 기술의 발달로 최근에는 병렬 처리 시스템(parallel processing system)의 개발에 많은 관심이 집중되고 있으며, 이러한 시스템을 지원하기 위한 구조에 대하여 많은 연구가 진행되고 있다[1-8]. 이러한 병렬 처리 시스템 구조를 지원하기 위한 시스템으로서 bisectional 상호연결 네트워크(이하 bisectional 네트워크)가 있다[9]. Bisectional 네트워크는 시스템 구조 특성상 각종 위상(topology)으로 적용될 수 있으며, 이 위상에는 선형 배열(linear array), 완전 이진 트리(complete binary tree), 메쉬 구조(mesh structure)가 있다[10, 11].

또한, 최근에 다중 컴퓨터(multicomputer) 시스템을 지원하기 위해 많이 연구되고 있는 하이퍼큐브(hypercube) 위상은, 직경(diameter)이 노드(node) 수에 대수적으로 증가하며, 구조적인 규칙성(regularity)과 간단한 경로배정(routing) 알고리즘, 노드간의 충분한 병렬 경로의 존재등의 성질을 갖기 때문에 노드들을 상호연결시키는 메시지 전달 구조로서 큰 관심이 집중되고 있다[12-18].

본 논문에서는 bisectional 네트워크가 제공하는 기본 위상 이외에, 최근 많은 관심이 집중되어 연구되고 있는 하이퍼큐브 구조를 구현하며, 여기에 필요한 경로배정 알고리즘과 방송(broadcasting) 알고리즘을 제안한다. Bisectional 네트워크는 세가지 위상인 선형 배열, 완전 이진 트리, 메쉬 구조로 적용되며 특히, 모든 메쉬 구조는 하이퍼큐브 구조로 적용될 수 있기 때문에, 적용된 메쉬 구조를 기초로 항상 하이퍼큐브 구조의 구현이 가능하다[19].

Bisectional 네트워크는 홀수 정도(odd degree) 'd'를 갖는 구조로서, 각 노드(node)의 코드(codeword)는 이진 짝수-비중(binary even-weight)을 갖는다.

bisectional 네트워크에서 적용된 하이퍼큐브는 네트워크의 정도가 d일 경우, d−1차원의 하이퍼큐브가 생성된다. 각 노드의 코드에서 비트 위치(bit position)를 i라 할 때, i는 LSB부터 '0123...'의 순서를 갖는다. 하이퍼큐브의 생성조건은 기본 네트워크의 링크에서 $d \geq 5$인 경우 $i = d-4$, $d < 5$인 경우에는 $i = 0$인, i번째 같은 위치를 갖는 링크는 연결하지 않는 것이다. 위와 같은 조건으로 생성된 하이퍼큐브는 다음과 같은 성질을 갖는다.

각 노드는 이진 짝수-비중을 갖는 d비트 이진수로 표현되기 때문에 그 하이퍼큐브 구조의 차원보다 더 많은 비트를 갖게 된다. 또한 특정 조건($i = d-4$, $i = 0$)을 만족하는 링크는 연결하지 않기 때문에 원래 네트워크에서의 정도보다 하나 작은 정도를 갖게 됨으로서, 기존 bisectional 네트워크에서 사용하는 경로배정 알고리즘을 사용할 수 없으며, 일반적인 하이퍼큐브 구조와는 노드 표기 방법과 링크 연결 방법이 다르기 때문에 일반적인 하이퍼큐브에서 사용하는 알고리즘도 사용할 수 없다. 따라서, 새로 적용된 하이퍼큐브 구조에 적합한 경로배정 알고리즘이 필요하게 된다.

본 논문에서 제안하는 하이퍼큐브에서의 경로배정 알고리즘은 bisectional 네트워크의 알고리즘에서 수행되는 경로배정을 그대로 수용한다. 그러나 하이퍼큐브로의 구현시 제외된 링크가 있기 때문에 최단 거리의 경로에 이 링크가 포함되어 있으면, 이 링크를 제외한 나머지 경로를 통해서 하이퍼큐브와 같은 경로배정 알고리즘을 갖게 된다. 이 알고리즘은, 먼저 근원지 노드와 목적지 노드 사이의 최단 경로를 찾는다. 그러나 조건을 검사하여 이 경로 중에 하이퍼큐브의 구현시 제외된 링크가 포함되어 있으면 경로배정을 바꾸어 일반적인 하이퍼큐브와 같은 방법으로 최단 경로를 찾게 된다.

또한 하이퍼큐브에서 방송 알고리즘은 모든 노드에게 방송 메시지를 보낼 때 메시지와 함께 'weight'를 전송하는데 weight에 의해서 메시지를 전달하는 노드는 어떻게 메시지를 다음 노드로 방송할 것인가

를 결정한다. 여기에서의 weight는 연결된 두 노드간의 틀린 비트 위치를 나타낸다. 본 논문에서 제안하는 방송 알고리즘은 일반 하이퍼큐브에서의 방송 알고리즘과 전체적으로 유사하게 수행된다. 그러나 일반 하이퍼큐브에서는 weight가 두 노드간의 틀린 비트 위치가 되지만 제안하는 방송 알고리즘에서는 weight를 주는데 있어서 두 노드간의 같은 비트 위치를 전달해 준다는 것이다. 물론, 하이퍼큐브 구현시 제외된 링크는 사용하지 않는다. 이렇게 제안된 방송 알고리즘은 일반적인 하이퍼큐브에서의 알고리즘과 같이 모든 노드에 d-1(차원)단계 내에 정확히 한번씩 메시지를 전달하게 된다.

본 논문에서는 Bisectional 네트워크 내에서 또다른 위상인 하이퍼큐브를 일정한 규칙으로 구현시키고 이에 해당하는 경로배정 알고리즘과 방송 알고리즘을 제시함으로써, bisectional 네트워크에서 하이퍼큐브 구조를 구현하게 된다.

Ⅱ. Bisectional 상호연결 네트워크

2.1 Bisectional 네트워크의 개요

Bisectional 네트워크는 네트워크를 구성하는 노드간의 거리가 작으며, 이의 경로배정 알고리즘은 자기 경로배정(self-routing)과 최대한의 오류 허용 능력(fault tolerant)을 갖는다.

위상을 구성하는 방법으로서, 각 노드를 나타내는 코드는 이진 짝수 비중이고 각 노드의 연결은 두 노드간의 코드를 비교하여 한 비트만이 같은 두 노드를 연결하여 각 노드의 정도는 홀수 d를 갖게 한다. 또한, 범용의 병렬처리 시스템을 지원하기 위한 특성으로서 여러 형태의 네트워크가 효과적으로 적용될 수 있다. Bisectional 네트워크가 허용하는 논리적인 구조에는 선형 배열, 완전 이진 트리, 메쉬 구조가 있으며 모든 메쉬 구조는 하이퍼큐브 구조로의 적용이 가능하다. 따라서, 본 논문에서 제안하는 하이퍼큐브 구조의 구현과 그에 해당하는 경로배정 알고리즘 및 방송 알고리즘의 설계를 가능하게 한다. Bisectional 네트워크의 시스템 위상은 아래와 같은 그래프로 표현될 수 있다.

G=(U, E)　　여기서, U:노드의 집합

E:간선의 집합

네트워크의 각 노드는 프로세서와 메시지 교환 요소의 조합으로 구성되며, 각 간선은 양방향 데이타 링크로 가정한다. 또한 각 노드의 정도는 각 노드에 연결되는 간선의 수를 나타낸다.

네트워크에서 중요한 성질은 메시지의 자기-경로배정 능력이다. 이것은 네트워크에 오류가 없는 경우 뿐만 아니라, 오류가 있는 경우에 대해서도 같이 적용된다. 오류가 없는 네트워크에 대해서는 경로배정 알고리즘이 순수히 하드웨어적인 논리(hardwired logic)로 구현되며, 이것은 오류가 있는 네트워크에 대해서도 쉽게 확장가능하다. 자기-경로배정 능력은 경로배정 테이블이나 기타 데이타베이스가 필요 없으며, 오히려 목적지 주소(destination address)의 가장 짧은 경로를 찾기 위해서는 메시지에 목적지 주소만이 사용된다. 네트워크 내에서 위와 같은 성질을 갖는 임의의 두 노드간의 최단 거리를 찾기 위해서는 다음의 【정의 1】을 사용한다[9].

【정의 1】 그래프의 임의의 두 노드 i, j 사이의 가장 짧은 거리를 최단 거리(minimum distance)라 하며, L_{ij}로 나타낸다. 네트워크에서 최단 거리를 갖는 경로를 최단 경로라고 한다.

2.2 Bisectional 네트워크의 위상

Bisectional 네트워크의 위상은 네트워크의 정도를 나타내는 변수 d(홀수)로 표현되며, 네트워크의 전체 노드 수는 2^{d-1}이다. Bisectional 네트워크의 위상 구성 방법은 아래와 같다.

■ Bisectional 네트워크의 위상 구성 방법

□ 홀수인 자릿수가 d인 이진 코드에 대해서 각 코드는 모두 짝수 비중을 갖도록 한다. 각 코드를 그래프의 노드로 표현한다.

□ Hamming 거리(두 노드의 코드를 비교해서 서로 다른 비트의 갯수)가 d-1인 두 개의 노드를 서로 연결한다.

위와 같은 방법으로 구성된 bisectional 네트워크의 정도는 홀수 d이고, 직경 K는 (d-1)/2 이며, 총 노드

수는 2^{d-1}이다. 이러한 네트워크를 bisectional 네트워크라 하며, 이 네트워크의 노드 수는 이진 d-큐브의 절반이 된다. 위와 같은 방법으로 구성된 bisectional 네트워크가 아래 【예 1】에 나타나 있다.

【예 1】 d = 5인 경우의 시스템 위상은 아래 〈그림 1〉과 같다. 전체 노드 수는 16이고 직경 K = 2이다.

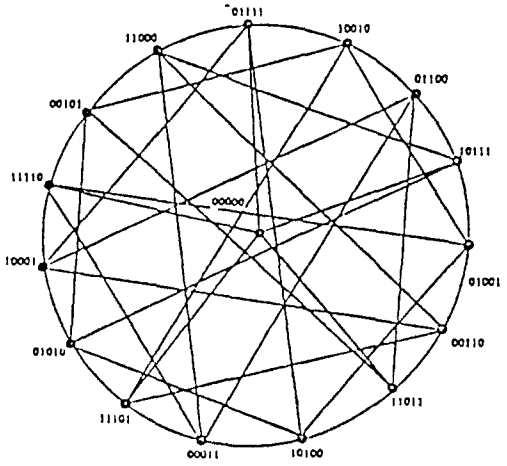


그림 1. Bisectional 네트워크의 위상 (d = 5)

Bisectional 네트워크의 정도 d는 전체 노드 수의 로그($\log_2$)와 같고, 직경 K는 전체 노드 수 n에 대해서 $(\log_2 n)/2$에 비례한다. 이 결과로 같은 크기일 경우, 이진 d-큐브의 직경이 전체 노드 수에 비례하기 때문에 bisectional 그래프가 더 밀도(density)가 높다고 할 수 있다. 위와 같은 이유로 트리, 링 구조(chordal ring)[21]나 일반화된 이진 큐브(generalized binary cube) 네트워크보다 활용성이 더 높다고 할 수 있다.

인접성 규칙(adjacency rule)에 따라 이웃 노드간의 코드는 hamming 거리 d−1을 갖는다. 이것은 이진 d-큐브와 그의 일반화된 구조의 인접성 규칙과는 다른 것이다. 결과적으로 제안된 네트워크는 더 높은 밀도(density), 각 논리적인 구조로의 최적화된 적용, hamming 거리와 최단 경로 거리(minimum path distance) 간의 일정한 관계등의 특성을 갖는다.

또한, 임의의 두 노드 x와 y사이의 최단 거리를 L_{xy}라 하고, H_{xy}를 hamming 거리라 할 때, $H^+_{xy} = d - H_{xy}$이면 $L_{xy} = \mathrm{Min}(H_{xy}, H^+_{xy})$이다[5]. 위의 식에 따라 임의의 두 노드 사이의 최단 거리 L_{xy}를 찾을 수 있으며, 실제 거리와 hamming 거리와의 관계가 [표 1]에 나타나 있다. 예를 들어, 네트워크 정도 d가 11인 경우, 두 노드의 $H_{xy} = 8$일 때, [표 1]에서 알 수 있듯이 최단 거리는 3 이다.

표 1. 실제 거리와 hamming 거리와의 분포도

정 도	d =	5	7	9	11	13	15	
	L_{xy}							
	1	4	6	8	10	12	14	
	2	2	2	2	2	2	2	hamming
실 제	3	-	4	6	8	10	12	거 리
거 리	4	-	-	4	4	4	4	
	5	-	-	-	6	8	10	
	6	-	-	-	-	6	6	
	7	-	-	-	-	-	8	

임의의 두 노드를 x, y라 할 때, 각 노드의 이진 표현에서 i번째 비트가 같은 값 i를 존재하는 링크로 나타낼 수 있다. 이에 따라 두 노드 사이의 경로는 이 값들의 순서(sequence)로 표현 가능하다. 즉, 〈그림 2〉에서 2, 1, 6과 같은 값이다.

Bisectional network에서 임의의 노드 x와 y사이에서 중복되는 노드 없이(node-disjoint) 갈 수 있는 경로의 수는 d 이며, 이러한 경로의 길이는 아래와 같다.

1) $L_{xy} = H_{xy} (\neq K$, 직경)이면, H_{xy} 길이를 갖는 H_{xy}개의 경로가 존재하며, $H_{xy}+2$의 길이를 갖는 H^+_{xy}개의 대체(alternate) 경로가 존재한다. 만약, $H_{xy} = K$ 이면 대체 경로의 길이는 K + 1 이다.
2) $L_{xy} = H^+_{xy} (\neq K)$이면, H_{xy} 길이를 갖는 H^+_{xy} 개의 경로가 존재하며, $H^+_{xy}+2$ 길이를 갖는 H^+_{xy} 개의 대체 경로가 존재한다. $H^+_{xy} = K$ 이면, K + 1 길이를 갖는 대체 경로를 구성할 수 있다.

이러한 중복이 없는 경로들은 모두 노드 x와 y 사이의 가능한 가장 짧은 경로를 갖게 되며, 두 노드간의 경로배정을 위해서 아래의 【정의 2】를 통해 두 노드간의 비트 위치의 조합을 알 수 있다[9]. 아래 【예

2]에는 두 노드간의 최단 경로와 각 비트 위치에 대한 집합을 보여 주고 있다[9].

【정의 2】 P^{xy}_{ij}: 임의의 두 노드를 x, y라 할 때, 이들의 이진 표현 값이 각각 i, j인 비트 위치(bit position)들의 집합. 단 i, j={0, 1}

【예 2】 정도 d가 7이며, 직경이 3인 네트워크에서 근원지 주소 x가 0001100이고 목적지 주소 y가 0110101 이라 하면, $H_{xy}=4$이고 $H^{+}_{xy}=d-H_{xy}=3$ 이다. 따라서, L_{xy}(최단 거리)=3 이다. 또한 P^{xy}_{ij}에 대한 각 집합은 아래와 같다.

x : 0 0 0 1 **1** 0 0
y : 0 1 1 0 **1** 0 1 $\Rightarrow P^{xy}_{11}=\{2\}$,

x : 0 **0** **0** 1 1 0 **0**
y : 0 **1** **1** 0 1 0 **1** $\Rightarrow P^{xy}_{01}=\{0, 4, 5\}$,

x : 0 0 0 **1** 1 0 0
y : 0 1 1 **0** 1 0 1 $\Rightarrow P^{xy}_{10}=\{3\}$,

x : **0** 0 0 1 1 **0** 0
y : **0** 1 1 0 1 **0** 1 $\Rightarrow P^{xy}_{00}=\{1, 6\}$,

다음 장에서 경로배정 알고리즘이 언급되겠지만, 위의 예에서 나타난 직접 경로(direct path)를 위한 i 값들을 먼저 살펴보면 2, 1, 6이 있다. 따라서 근원지 노드(0001100)에서 목적지 노드(0110101)로 가기 위해서는 근원지 노드 x에서 비트 위치 2가 같은 노드를 먼저 찾아 가면 1110111이 되며, 다음에는 비트 위치 1이 같은 노드를 찾게 되면 0001010이 된다. 마지막으로 비트 위치 6이 같은 노드를 찾으면 목적지 노드 0110101이 된다. 위의 경로배정 순서는 2, 1, 6을 따르고 있으며, 다른 조합으로 6, 2, 1과 1, 6, 2 등이 있다. 이에 대한 경로배정도 위와 같은 순서를 따르며, 이를 위한 경로의 조합이 아래 〈그림 2〉에 나타나 있다.

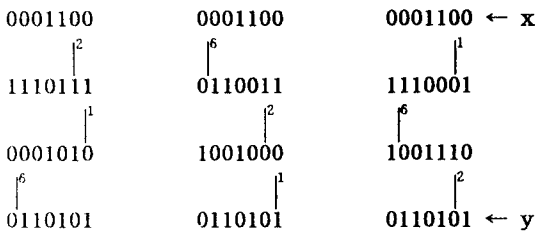


그림 2. 경로배정의 예(x : 0001100, y : 0110101)

2.3 Bisectional 네트워크에서의 경로배정 알고리즘

메시지를 받는 노드 w는 메시지를 전송하기 위해 〈*Algorithm 1*〉과 같은 방법으로 이웃 노드를 찾는다. 특히, w가 출발 노드인 근원지 노드일 경우에는 받을 메시지가 없기 때문에 메시지를 보내기 위한 경우이고, 이 w는 경로중의 노드를 나타내기 때문에 목적지 노드가 아닌 경로중의 노드를나타낸다. 아래 〈*Algorithm 1*〉은 오류가 없는 상태에서 임의의 두 노드간의 경로배정을 보여주며, 이에 대한 예가 아래 【예 3】에 나타나 있다.

〈*Algorithm 1*〉
Step1 : Find L_{wy}.
Step2 : Find sets P^{wy}_{00}, P^{wy}_{01}, P^{wy}_{10}, P^{wy}_{11}.
Step3 : If L_{wy} = even, then goto Step4
else $i \in P^{wy}_{00} \cup P^{wy}_{11}$ 인 i를 찾아서 이웃 노드 n_i를 선택. Stop.
Step4 : $i \in P^{wy}_{10} \cup P^{wy}_{01}$ 인 i를 찾아서 이웃 노드 n_i를 선택. Stop.

〈*Algorithm 1*〉의 수행 과정을 살펴보면, 먼저 Step1에서 $\mathrm{Min}(H_{xy}, H^{+}_{xy})$에 따라 L_{wy}를 찾게 된다. Step2에서는 근원지 노드 w와 목적지 노드 y 사이의 두 노드간의 비트 위치를 나타내는 집합인 P^{xy}_{ij}를 구한다. Step3에서 L_{wy}가 홀수인 경우, $P^{wy}_{00} \cup P^{wy}_{11}$중에서 하나의 원소를 선택하여 그 비트 위치가 같은 이웃 노드로 가게 된다. 짝수에 대해서도 위와 같은 수행 과정을 따르는데 Step2까지는 같으며, 짝수이기 때문에 Step4로 가서 $P^{wy}_{10} \cup P^{wy}_{01}$중의 원소를 하나 선택하여 그 비트 위치가 같은 이웃 노드로 가게 된다. 〈*Algorithm 1*〉의 수행 과정을 보여 주기 위한 예가 아래 【예 3】에 나타나 있다.

【예 3】 정도 d가 9이고 즉, 전체 노드 수 n이 256인 네

트워크에서, 근원 지 노드 x를 001101010, 목적지 노드 y를 001010000이라 할 때, 경로배정은 위의 알고리즘에 따라 아래와 같은 절차를 거쳐 이루어진다.

Step1 : $L_{xy}=4=H^{+}_{xy}$
Step2 : $P^{xy}_{00}=\{0, 2, 7, 8\}$
$P^{xy}_{11}=\{6\}$
$P^{xy}_{10}=\{1, 3, 5\}$
$P^{xy}_{01}=\{4\}$
Step3 : L_{xy}가 짝수이므로 goto Step4.
Step4 : $P^{xy}_{10}\cup P^{xy}_{01}=\{1, 3, 4, 5\}$
여기에서 1을 선택하여 이웃 노드 n_1으로 메시지를 전송함.

아래의 〈그림 3〉은 위의 예에서 나타난 x와 y 사이의 가장 짧은 경로 중의 하나를 보여준다. 먼저, 근원지 노드 x(001101010)에서 비트 위치 1이 같은 노드인 110010111로 간다. 다음에는 비트 위치 3이 같은 00110000로 가게 되며, 다음으로 비트 위치 4가 같은 110001111로 간다. 마지막으로, 비트 위치 5가 같은 001010000으로 가면 목적지 노드 y가 되어 경로배정을 마치게 된다.

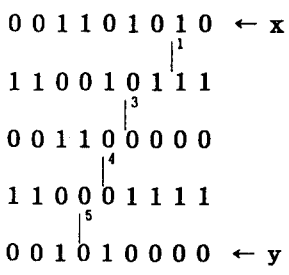


그림 3. 가장 짧은 경로의 예(x : 001101010, y : 001010000)

Ⅲ. Bisectional 네트워크에서 하이퍼큐브의 구현

홀수 정도 d를 갖는 bisectional 네트워크는 선형 배열, 완전 이진 트리, 메쉬의 세가지 중요한 위상으로 최적화되어 적용될 수 있다. 특히 모든 메쉬 구조는 항상 하이퍼큐브로의 적용이 가능하기 때문에, bisectional 네트워크의 모든 정도 d에 대해서 d−1 차원의 하이퍼큐브가 항상 존재하게 된다. 즉, d가 3인 경우 2차원, d가 5인 경우 4차원, d가 7인 경우 6차원 등의 하이퍼큐브가 존재한다. 각각의 d에 대해서 적용된 메쉬 구조를 기초로 하이퍼큐브를 구현할 때 아래 설명과 같은 조건 하에 메쉬에서 하이퍼큐브를 구현한다. Bisectional 네트워크에서 하이퍼큐브로의 적용시, 각 d에 대해서 일정 조건의 같은 비트 위치를 갖는 링크를 없애고 하이퍼큐브를 적용한다.

3.1 구현 과정

Bisectional 네트워크는, $1\le s\le d-1$인 F^s라고 하는 S-차원의 공간이 주어지면, F^s 공간 내에서 정도 d의 bisectional 네트워크는 $w_1\times w_2\times \dots \times w_s$ 차원의 메쉬로 적용될 수 있다. 여기서, $w_i=2^{bi}$이고 $\sum b_i=d-1$이다[9]. 위와 같은 성질에 따라 모든 bisectional 네트워크는 임의의 차원의 메쉬로 구현 가능하다.

또한, d-차원 R^d에서 임의의 $m_1\times m_2\times \dots \times m_d$ 메쉬($m_i=2^{pi}$)는 n-큐브로 매핑될 수 있으며($n=P_1+P_2+\dots+P_d$). 여기에서 각 점에 대한 번호매김(numbering)은 그레이 순서(gray sequence)를 따른다[19]. 이에 따라 임의의 메쉬는 n-큐브로의 구현이 가능하다.

위와 같이 bisectional 네트워크는 임의의 메쉬로 구현가능하며, 임의의 메쉬는 n-큐브로의 구현이 가능하기 때문에 bisectional 네트워크에서 하이퍼큐브의 구현이 항상 가능하다. 이와같이 구현된 그래프 G=(V, E)가 n-큐브일 조건은 아래와 같다.

1) V는 2^n 개의 노드이다.
2) 모든 노드는 정도 n을 갖는다.
3) G는 연결되어 있다.
4) 임의의 이웃한 두 노드 A와 B는 일대일(one-to-one) 대응을 이룬다.

위와 같은 과정을 거쳐 bisectional 네트워크에서 생성된 각 간선에 둘러싸여 연결된(wraparound) 4×4 메쉬(격자)는 4-큐브로 생성될 수 있다.

3.2 하이퍼큐브 구현 규칙과 적용예

위와 같은 설명을 기초로 bisectional 네트워크에서 하이퍼큐브의 구현이 가능하며 하이퍼큐브 구현시 나타나는 없는 링크의 비트 유형은 [표 2]와 같다. 예

를 들어, d가 7인 경우에는 bisectional 네트워크에서 링크를 연결하는 두 노드간의 비트 위치가 '3'(LSB부터 4번째)인 링크는 없애고 하이퍼큐브를 구현한다. 이를 요약하면 아래와 같다.

표 2. 적용된 하이퍼큐브에서 없는 링크의 비트 유형

정 도 (d)	차 원 (d-1)	없는 링크의 비트 유형 (두 노드간의 비트 위치 비교)
3	2	'0' 비트 위치 같음
5	4	'1' 〃
7	6	'3' 〃
9	8	'5' 〃
11	10	'7' 〃
⋮	⋮	⋮

[표 2]를 기초로 네트워크의 정도 d와 두 노드간의 같은 비트 위치 i는 아래 [표 3]과 같은 관계를 가지며 규칙 ①, ②를 기초로 하이퍼큐브를 구현한다.

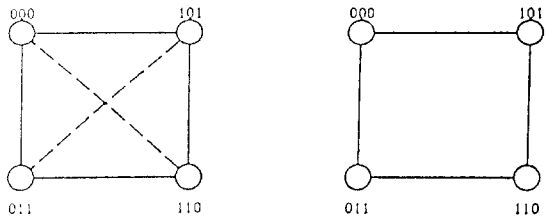
표 3. 네트워크 정도 d와 비트 위치 i와의 관계

d	3	5	7	9	11	13	…
i	0	1	3	5	7	9	…

(d : 네트워크의 정도, i : 두 노드간의 같은 비트 위치)

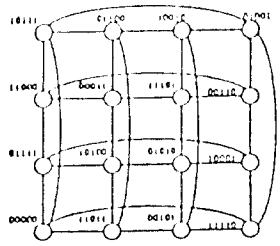
규 칙 : $i = \begin{cases} 0, & \text{for } d < 5 \quad \cdots\cdots ① \\ d-4, & \text{for } d \geq 5 \quad \cdots\cdots ② \end{cases}$

[표 3]의 규칙 ①은 d가 1과 3인 경우를 말하며 이 때는 i가 0인 링크를 없애고 큐브를 구현하며, 규칙 ②에서는 d가 5보다 크거나 같은 홀수를 말하며, 이 때는 i가 d보다 4가 작은 링크는 없애고 하이퍼큐브를 구현한다. 위와 같은 관계를 기초로 아래 〈그림 4〉과 〈그림 5〉에서는 d가 3과 5인 경우에 대해서 메쉬 구조를 기초로 하이퍼큐브가 구현되는 과정을 보여준다. 〈그림 4〉의 (a)는 d가 3인 bisectional 네트워크에서 메쉬로 바뀌는 과정을 보이는데 점선까지 포함한 부분이 원래의 bisectional 네트워크를 나타낸다. 여기에서 000과 110, 011과 101을 연결하는 점선은 규칙 ①에 따라 d가 3이기 때문에 비트 위치 '0'으로 같은 두 노드간의 링크를 제외하게 된다. 따라서 점선을 제외한 000, 011, 110, 101을 연결하는 메쉬가 생성되며, 이는 곧 (b)의 2차원 하이퍼큐브가 된다. 또한 〈그림 5〉의 (a)는 d가 5인 bisectional 네트워크에서 규칙 ②에 따라 두 노드간의 비트 위치가 i=d-4인 즉, 비트 위치 '1'로 같은 두 노드간의 링크를 제외시키고 생성된 메쉬를 보여 주고 있다. 그림에서 보듯이 각 노드는 링크가 하나 감소한 정도 4를 보이고 있으며, 이는 곧 (b)의 4차원 하이퍼큐브로 구현된다.

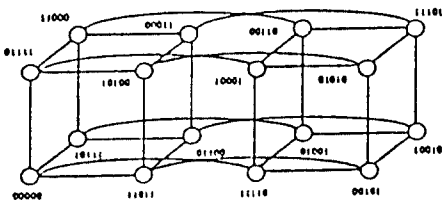


(a) 메쉬 (b) 하이퍼큐브

그림 4. d = 3인 경우, 메쉬와 하이퍼큐브의 구현



(a) 메쉬



(b) 하이퍼큐브

그림 5. d = 5인 경우, 메쉬와 하이퍼큐브의 구현

Ⅳ. 경로배정 및 방송 알고리즘

Bisectional 네트워크에서 적용된 하이퍼큐브는 기

존의 하이퍼큐브 노드와는 다른 표기를 갖는다. 즉 기존의 하이퍼큐브는 각 노드 번호의 비트 수와 그 구조의 차원이 같지만(3차원=3 비트), 적용된 하이퍼큐브는 차원 수보다 더 많은 비트를 갖는다. 이와 같은 구성상의 차이점 때문에 다른 노드 표기를 가지고 하이퍼큐브의 경로배정을 이룰 수 있어야 한다. 따라서 이에 해당하는 경로배정 알고리즘을 제안하며, 방송 알고리즘은 기존의 방법[20]과 유사하지만 메시지 전송시 전달해 주는 weight값에 있어서 다른 값을 갖고 방송을 한다. 특히, 방송 알고리즘은 weight 값만을 바꾸어 전송하기 때문에 기존의 하이퍼큐브 방송 알고리즘의 성질인, 모든 노드에게 방송 메시지를 정확히 한번씩 d−1 단계내에 보낼 수 있다.

4.1 경로배정 알고리즘

Bisectional 네트워크에서 구현된 하이퍼큐브는 [표 2]와 [표 3]을 기초로 하고 있으며, 본 장에서 제안하는 경로배정 알고리즘은 〈*Algorithm 1*〉을 기초로 하고 있다. 즉, 적용된 하이퍼큐브는 [표 3]의 규칙 ①, ②를 기초로 구성되었기 때문에 〈*Algorithm 1*〉에서 수행되는 모든 링크가 포함되어 있지 않다. 물론 제외된 링크는 규칙 ①, ②의 링크 i이다. 따라서 최단 경로의 설정시 규칙 ①, ②와 같은 링크가 존재하면 경로를 바꾸어야 한다.

〈*Algorithm 1*〉에서 Step1과 Step2에 따라 L_{wy}와 P^{wy}_{00}, P^{wy}_{01}, P^{wy}_{10}, P^{wy}_{11}을 찾은 후, L_{wy}의 홀수 짝수 여부에 따라 $P^{wy}_{00} \cup P^{wy}_{11}$ 집합과 $P^{wy}_{01} \cup P^{wy}_{10}$ 집합을 구한다. 그러나 $P^{wy}_{00} \cup P^{wy}_{11}$ 집합과 $P^{wy}_{01} \cup P^{wy}_{10}$ 집합에 규칙 ① 혹은 ②를 만족하는 링크 i가 존재하면 하이퍼큐브의 구현시 없어진 링크이기 때문에 다른 경로 즉, 하이퍼큐브의 경로배정 방식을 따라가야 한다. 물론, $P^{wy}_{00} \cup P^{wy}_{11}$ 집합이나 $P^{wy}_{01} \cup P^{wy}_{10}$ 집합에 규칙 ①, ②를 만족하는 링크 i가 없으면 기존 알고리즘의 방법을 따르며, 하이퍼큐브와 같이 경로배정이 이루어진다.

최단 경로에 규칙 ①, ②를 만족하는 링크 i가 존재하면 최단 경로 L_{wy}를 $\mathrm{Max}(H_{wy}, H^{+}_{wy})$로 바꾼다. L_{wy}를 H_{wy}와 H^{+}_{wy}중에서 큰 값으로 바꾸게 되면 $P^{wy}_{00} \cup P^{wy}_{11}$ 집합과 $P^{wy}_{01} \cup P^{wy}_{10}$ 집합은 서로 상호 배제되는 집합이므로 새로운 L_{wy}에는 링크 i가 존재하지 않는다. 또한 새로운 L_{wy}는 링크 i가 포함된 경로를 제외한 다른 경로 중 최단 경로를 갖게 된다. 따라서 $\mathrm{Max}(H_{wy}, H^{+}_{wy})$에 의한 L_{wy}에 규칙 ①, ②의 i가 포함되지 않을 경우에는 기존의 〈*Algorithm 1*〉과 같이 경로배정이 이루어지며, 위 규칙의 링크 i가 있어 L_{wy}가 $\mathrm{Max}(H_{wy}, H^{+}_{wy})$로 바뀌었을 때는 링크 i를 제외한 다른 경로 중 최단 경로로 경로배정이 이루어진다.

아래 〈*Algorithm 2*〉는 적용된 하이퍼큐브에서 수행되는 경로배정 알고리즘을 보여 준다. Step1에서는 두 노드간의 비트 위치를 나타내는 P^{wy}_{00}, P^{wy}_{01}, P^{wy}_{10}, P^{wy}_{11} 집합을 구하고 규칙 ①, ②의 링크 i를 검사하기 위해 ODD와 EVEN이라는 집합을 정의한다. Step2에서는 H_{wy}와 H^{+}_{wy}를 기초로 최단 경로 L_{wy}를 $\mathrm{Min}(H_{wy}, H^{+}_{wy})$로 구한다. Step3에서는 규칙 ①, ②를 검사한다. 먼저, 정도 d가 5보다 큰지 작은지의 여부를 판단한다. 그리고 난 후, d가 5보다 작을 경우에는 각 집합(ODD, EVEN)에 대해서 비트 위치 0가 포함되어 있는지 판단하여 0이 포함되어 있을 경우에는 다른 경로 즉, L_{wy}를 $\mathrm{Max}(H_{wy}, H^{+}_{wy})$로 바꾼 다음 다시 Step3을 수행한다. 만약 0이 포함되어 있지 않을 경우에는 그 집합의 원소에 대해서 임의의 i를 선택하여 이웃 노드 n_i로 간 후 멈추게 된다. 위와 같은 방법으로 d가 5보다 크거나 같은 경우에 대해서도 알고리즘의 수행이 이루어지며, 이 경우의 비트 위치 i는 d−4인 링크이다. 〈*Algorithm 1*〉과 마찬가지로 메시지를 받는 노드 w는 메시지를 전송하기 위해 아래 알고리즘을 수행하며, w가 근원지 노드 x와 같은 경우는 메시지를 보내기 위해서이다.

〈*Algorithm 2*〉

Step1 : Find Sets P^{wy}_{00}, P^{wy}_{11}, P^{wy}_{01}, P^{wy}_{10}.
Let Set ODD = {i} = $P^{wy}_{00} \cup P^{wy}_{11}$,
EVEN = {i} = $P^{wy}_{01} \cup P^{wy}_{10}$.

Step2 : Find $L_{wy} = \mathrm{Min}(H_{wy}, H^{+}_{wy})$

Step3 : Case (d, L_{wy}, ODD, EVEN) of
: (d < 5) & (L_{wy} = even) & (0 ∈ EVEN) then goto Step4;
: (d < 5) & (L_{wy} = even) & (0 ∉ EVEN) then Choose an element i, with i ∈ EVEN. Select neighbor node n_i, Stop;
: (d < 5) & (L_{wy} = odd) & (0 ∈ ODD) then goto Step4;

:($d<5$) & (L_{wy} = odd) & (0 ∉ ODD) then Choose an element i, with i ∈ ODD. Select neighbor node n_i, Stop;

:($d \geq 5$) & (L_{wy} = even) & ($d-4=i \in$ EVEN) then goto Step4;

:($d \geq 5$) & (L_{wy} = even) & ($d-4=i \notin$ EVEN) then Choose an element i, with i ∈ EVEN. Select neighbor node n_i, Stop;

:($d \geq 5$) & (L_{wy} = odd) & ($d-4=i \in$ ODD) then goto Step4;

:($d \geq 5$) & (L_{wy} = odd) & ($d-4=i \notin$ ODD) then Choose an element i, with i ∈ ODD. Select neighbor node n_i, Stop;

Step4: Replace $L_{wy} = \mathrm{Max}(H_{wy}, H^+_{wy})$, goto Step3;

【정리 1】 위의 〈*Algorithm 2*〉는 항상 노드 w와 y 사이의 가장 짧은 경로를 갖는다.

(증 명) L_{zy}가 가장 짧은 경로를 갖는다는 것을 증명함으로써 L_{wy}가 가장 짧은 경로를 가짐을 증명한다. 여기에서 z는 제안된 알고리즘이 사용하는 w에 의해 선택된 w의 이웃 노드이며, L_{zy}는 w와 목적지 노드 y 사이의 거리인 L_{wy}보다 하나 작은 값을 갖는다. 이를 통해 순환적으로 증명할 수 있다. 아래의 각 경우는 정도를 나타내는 d의 조건과 최단 경로 L_{wy}가 짝수인지 홀수인지, 그리고 각 EVEN 이나 ODD 집합 내의 d에 따라 규칙 ①, ②의 i 값이 들어 있는지에 따라 나누어지기 때문에 d, Lwy, EVEN(ODD)에 따라 8가지 경우가 생긴다. 따라서 이의 각 경우에 대해서 증명한다.

Case 1: ($d<5$) & (L_{wy} = even) & (0 ∈ EVEN)

d가 5보다 작은 경우에는 L_{wy}가 짝수로서 EVEN내에 '0'이 포함되는 경우는 발생하지 않게 된다.

Case 2: ($d<5$) & (L_{wy} = even) & (0 ∉ EVEN)

L_{wy}가 짝수라면 $L_{wy} = H_{wy} < H^+_{wy}$ 이다. 그리고 $i \in P^{wy}_{01} \cup P^{wy}_{10}$와 인접성 요구에 따라 통해 이웃 노드를 하나 선택하면 $H^+_{zy} = H_{wy} - 1$이 되며 $H_{zy} = H_{wy} + 2$가 된다. 이를 통해 H_{zy}와 H^+_{zy}를 비교해 보면 $H^+_{zy} < H_{zy}$가 된다. 따라서 $L_{zy} = H^+_{zy}$이다. 위에서 $L_{wy} = H_{wy}$이기 때문에 $H^+_{zy} = H_{wy} - 1$에 각각 L_{zy}와 L_{wy}를 대입하면 $L_{zy} = L_{wy} - 1$이 된다. 따라서 w에 의해 선택된 이웃 노드인 z에서의 L_{zy}는 L_{wy}보다 하나 작은 값을 갖게 된다.

Case 3: ($d<5$) & (L_{wy} = odd) & (0 ∈ EVEN)

L_{wy}가 홀수라면 $L_{wy} = H^+_{wy} < H_{wy}$이다. 집합 EVEN 내에 규칙 ①의 '0'이 포함되어 있기 때문에 Step4로 가서 L_{wy}를 $\mathrm{Max}(H_{wy}, H^+_{wy})$로 대치하여 $L_{wy} = H_{wy}$가 되어 L_{wy}는 짝수가 된다. 다시 Step3로 가서 조건을 검사해 보면, ($d<5$) & (L_{wy} = even) & (0 ∉ EVEN)가 된다. 여기에서의 L_{wy} 값은 H_{wy}와 H^+_{wy}값 중에서 큰 값을 가진 값이고, $i \in P^{wy}_{01} \cup P^{wy}_{10}$와 인접성 요구에 따라 이웃 노드를 하나 선택하면 $H^+_{zy} = H_{wy} - 1$이 되며, $H_{zy} = H_{wy}$가 된다. 따라서 $L_{zy} = H^+_{zy}$이고 $L_{wy} = H_{wy}$가 되기 때문에 $L_{zy} = L_{wy} - 1$이 된다. 따라서 w에 의해 선택된 이웃 노드인 z에서의 L_{zy}는 L_{wy}보다 하나 작은 값을 갖게 된다.

Case 4: ($d<5$) & (L_{wy} = odd) & (0 ∉ ODD)

L_{wy}가 홀수라면 $L_{wy} = H^+_{wy} < H_{wy}$ 이다. 그리고 $i \in P^{wy}_{00} \cup P^{wy}_{11}$와 인접성 요구에 따라 이웃 노드를 하나 선택하면 $H_{zy} = H^+_{wy} - 1$이 되며 $H^+_{zy} = H^+_{wy} + 2$가 된다. 이를 통해 H_{zy}와 H^+_{zy}를 비교해 보면 $H_{zy} < H^+_{zy}$가 된다. 따라서 $L_{zy} = H_{zy}$이다. 위에서 $L_{wy} = H^+_{wy}$이기 때문에 $H_{zy} = H^+_{wy} - 1$에 각각 L_{zy}와 L_{wy}를 대입하면 $L_{zy} = L_{wy} - 1$이 된다. 따라서 w에 의해 선택된 이웃 노드인 z에서의 L_{zy}는 L_{wy}보다 하나 작은 값을 갖게 된다.

Case 5: ($d \geq 5$) & (L_{wy} = even) & ($d-4=i \in$ EVEN)

L_{wy}가 짝수라면 $L_{wy} = H_{wy} < H^+_{wy}$이다. 규칙 ②의 $i = d-4$인 i가 집합 EVEN내에 포함되어 있기 때문에 Step 4로 가서 L_{wy}를 $\mathrm{Max}(H_{wy}, H^+_{wy})$로 대치하여 L_{wy}는 H^+_{wy}가 되어 홀수가 된다. 다시 Step3로 가서 조건을 검사하면 ($d \geq 5$) & (L_{wy} = odd) & ($d-4=i \notin$ ODD)가 된다. $i \in P^{wy}_{00} \cup P^{wy}_{11}$와 인접성 요구에 따라 이웃 노드를 하나 선택하면 $H_{zy} = H^+_{wy} - 1$이 된다. 여기에서 $H_{zy} < H^+_{zy}$와 $H_{zy} > H^+_{zy}$의 두가지 경우가 있을 수 있는데, $H_{zy} > H^+_{zy}$인 경우에는 H^+_{zy} 내에 $d-4$인 i가 경로 속에 포함되어 다시 Step4로 가서 $L_{zy} = H_{zy}$가 된

다. 따라서 항상 L_{zy}는 H_{zy}가 된다. 결과적으로 $H_{zy}=H^{+}_{wy}-1$에 L_{zy}와 L_{wy}를 각각 대입하면 $L_{zy}=L_{wy}-1$이 된다.

Case 6: $(d \geq 5)$ & $(L_{wy}=\text{even})$ & $(d-4=i \notin EVEN)$
Case 2와 동일한 방법으로 증명될 수 있다.

Case 7: $(d \geq 5)$ & $(L_{wy}=\text{odd})$ & $(d-4=i \in ODD)$
L_{wy}가 홀수라면 $L_{wy}=H^{+}_{wy}<H_{wy}$이다. 규칙 ②의 $i=d-4$인 i가 집합 ODD내에 포함되어 있기 때문에 Step 4로 가서 L_{wy}를 $Max(H_{wy}, H^{+}_{wy})$로 대치하여 L_{wy}는 H_{wy}가 되어 짝수가 된다. 다시 Step3로 가서 조건을 검사하면 $(d \geq 5)$ & $(L_{wy}=\text{even})$ & $(d-4=i \notin EVEN)$가 된다. $i \in P^{wy}{}_{01} \cup P^{wy}{}_{10}$와 인접성 요구에 따라 이웃 노드를 하나 선택하면 $H^{+}_{zy}=H_{wy}-1$이 된다. 여기에서 $H_{zy}<H^{+}_{zy}$와 $H_{zy}>H^{+}_{zy}$의 두가지 경우가 있을 수 있는데, $H_{zy}<H^{+}_{zy}$인 경우에는 H_{zy} 내에 $d-4$인 i가 경로 속에 포함되어 다시 Step4로 가서 $L_{zy}=H^{+}_{zy}$가 된다. 따라서 항상 L_{zy}는 H^{+}_{zy}가 된다. 결과적으로 $H^{+}_{zy}=H_{wy}-1$에 L_{zy}와 L_{wy}를 각각 대입하면 $L_{zy}=L_{wy}-1$ 이 된다.

Case 8: $(d \geq 5)$ & $(L_{wy}=\text{odd})$ & $(d-4=i \notin ODD)$
Case 4와 동일한 방법으로 증명될 수 있다.

위와 같은 8가지 경우에 대해서 모두 L_{zy}는 L_{wy}보다 하나 작은 값을 항상 갖게 됨으로서 〈*Algorithm 2*〉는 항상 임의의 두 노드 w와 y사이의 가장 짧은 거리를 갖는다. Q.E.D

〈*Algorithm 2*〉의 수행과정을 보여 주기 위한 예가 아래 【예 4】에 나타나 있다.

【예 4】 정도 d가 5인 bisectional 네트워크에서 구현된 4차원 하이퍼큐브는 아래 〈그림 6〉과 같다. 여기서 근원지 노드 x를 11000, 목적지 노드 y를 00101이라 할 때, x에서 y로 메시지를 보낸다고 하자. 이에 따라 〈*Algorithm 2*〉를 수행하면, 먼저 Step1에서 $P^{xy}{}_{00}$, $P^{xy}{}_{01}$, $P^{xy}{}_{10}$, $P^{xy}{}_{11}$를 구하게 되며, 이는 아래와 같다.

$P^{xy}{}_{00}=\{1\}$,
$P^{xy}{}_{01}=\{0, 2\}$,
$P^{xy}{}_{10}=\{3, 4\}$,
$P^{xy}{}_{11}=\varphi=\{\}$

그리고 $P^{xy}{}_{00} \cup P^{xy}{}_{11}=\{1\}$을 ODD 집합이라 하고 $P^{xy}{}_{01} \cup P^{xy}{}_{10}=\{0, 2, 3, 4\}$를 EVEN 집합이라 하자. Step 2에서는 $Min(H_{xy}, H^{+}_{xy})$에 따라 L_{xy}를 구하면 1이 된다($H_{xy}=4$, $H^{+}_{xy}=1$). Step3에서 d, L_{xy}, ODD, EVEN을 기초로 조건을 검사해 보면 $(d \geq 5)$ & $(L_{xy}=\text{odd})$ & $(d-4=i \in ODD)$이다. 즉, d는 5이고 L_{xy}는 홀수 1이며, ODD 집합 {1} 안에 $i=d-4$인 i 값 1이 존재한다. 따라서 Step4로 가서 L_{xy}를 $Max(H_{xy}, H^{+}_{xy})$로 바꾼다. Step4에 의해 L_{xy}는 4로 바뀌고 다시 Step3으로 간다.

Step3에서 다시 조건을 검사해 보면 $(d \geq 5)$ & $(L_{xy}=\text{even})$ & $(d-4=i \notin EVEN)$이다. 즉 d는 5이고 L_{xy}는 짝수 4이며, EVEN 집합 {0, 2, 3, 4} 안에는 $i=d-4$인 i 값이 존재하지 않는다. 따라서 EVEN 집합 {0, 2, 3, 4}내의 임의의 원소 i를 선택하여 이웃 노드로 간다. 즉 먼저 0을 선택하여 노드 00110로 가고, 그 다음 2를 선택하여 11101로 간 후, 3을 선택하여 노드 01010으로 간다. 다음으로 4를 선택하여 최종 노드 00101로 가게 된다. 이의 경로배정은 기존 하이퍼큐브에서의 경로배정과 같이 수행된다. 〈그림 6〉에서 11000과 00101 사이의 점선은 원래의 bisectional 네트워크에는 존재하는 링크이다. 따라서 〈*Algorithm 1*〉에서는 $L_{xy}=1$에 따라 {1}의 집합으로 최종 목적지 00101로 가게

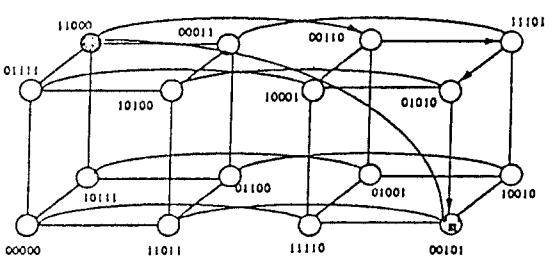


그림 6. d = 5인 경우 하이퍼큐브 경로배정의 예

된다. 그러나 하이퍼큐브 구현시 이 링크는 규칙 ①, ②에 따라 사용하지 않는다. 그러므로 위와 같은 방법으로 경로배정이 이루어진다.

4.2 방송 알고리즘

정도 d에 대해서 적용된 하이퍼큐를 위해서 사용되는 방송 알고리즘은 다음과 같다. 이 알고리즘은 임의의 노드에서 시작하며, d < 5인 경우에는 i = 0, d ≥ 5인 경우에는 i = d-4인 i 의 링크는 사용하지 않는다. 왜냐하면 하이퍼큐브의 구현시 이 링크는 사용하지 않기 때문이다. 방송이 시작되면 임의의 노드에서 메세지와 함께 모든 이웃 노드에 weight로서 같은 비트 위치를 전달한다. 각 노드는 같은 비트 위치로 메세지를 전달하며, 이 때 메시지를 받은 노드는 그 weight보다 작은 링크만을 사용하여 다음 노드로 방송을 한다. 각 노드는 받은 비트 위치 즉 weight가 '0' 일 경우, 더 이상 메세지를 전달하지 않는다. 물론, 앞에서 언급한 규칙 ①, ②에 해당하는 링크 i는 사용하지 않는다. 본 논문에서 제안하는 방송 알고리즘은 일반 하이퍼큐브 방송 알고리즘의 성질인 각 노드에 정확히 한번씩 d-1 단계내에 모든 방송을 마치게 된다.

〈*Algorithm 3*〉

Step1 : Start the algorithm at any node with weight set to d-1

Step2 : Initialize i, weight;

where, i : bit position between two nodes

weight : same bit position between two nodes,

Step3 : For each link i from this node with i less than weight

Send the message on link i with a weight of i;

where, if d < 5 then, Do not send the message and weight, for i = 0

else, Do not send the message and weight, for i=d-4

위의 〈*Algorithm 3*〉는 기존 하이퍼큐브의 방송 알고리즘과 유사하지만 단지 링크에 weight를 전해주는 데 있어서, 기존 하이퍼큐브는 두 노드간의 틀린 비트 위치를 전송하는 반면, 본 〈*Algorithm 3*〉에서는 두 노드간의 같은 비트 위치를 전달한다. 이와 같이 기존 알고리즘에서 weight 값만을 변화시켜 주기 때문에 기존 하이퍼큐브 방송 알고리즘의 성질을 그대로 만족한다. 적용된 하이퍼큐브에서 〈*Algorithm 3*〉를 수행한 방송 알고리즘의 예가 아래 【예 5】에 나타나 있다.

【예 5】 정도 d가 5인 bisectional 네트워크에서 적용된 하이퍼큐브의 방송 알고리즘을 생각해 보자. 〈그림 7〉에서와 같이 노드 *x* 10100에서 방송을 시작하는 경우, 첫번째 단계에서 초기 weight 4를 가지고 이웃 노드 01010, 01111, 00011, 11011로 방송 메시지와 함께 weight를 전달한다. 여기서 weight는 각각 0, 2, 3, 4 이다. d가 5이기 때문에 i = d-4인 i 값 1은 사용하지 않는다. 두번째 단계에서 노드 01010은 weight가 0이기 때문에 더이상 방송을 하지 않고, 노드 01111은 weight 2를 갖고 있기 때문에 이보다 작은(1을 제외한) 링크 0으로 노드 11101로 방송을 한다. 마찬가지로 노드 00011은 weight 0, 2에 해당하는 노드 11101, 11000으로 방송을 하며, 노드 11011은 weight 0, 2, 3에 해당하는 노드 00101, 00000, 01100으로 방송을 한다.

세번째 단계와 네번째 단계도 위와 같은 방법으로 수행되며 이에 대한 그림이 아래 〈그림 7〉에 나타나 있다. 알고리즘의 수행과정을 보여 주기 위한 방송 트리를 〈그림 8〉에 나타내며, 〈그림 8〉의 각 링크에는 그 노드에 주어지는 weight가 나타나 있다.

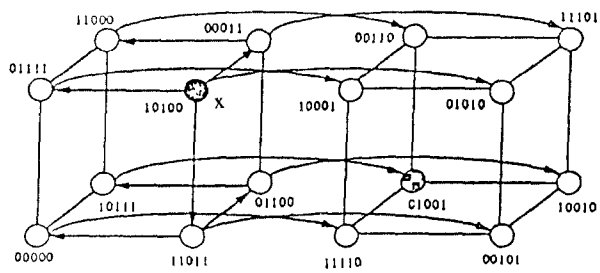


그림 7. d = 5인 경우 방송 알고리즘의 예

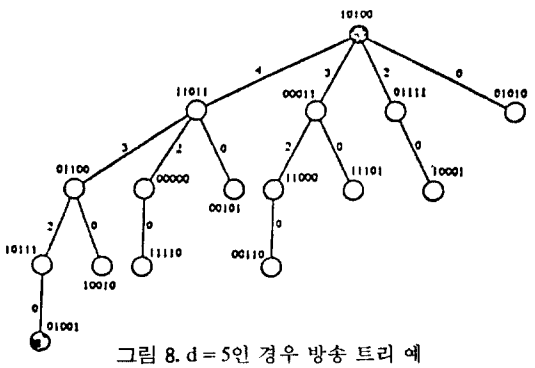


그림 8. d=5인 경우 방송 트리 예

V. 결 론

본 논문은 응용이 매우 자유로운 bisectional 네트워크에서 하이퍼큐브를 구현하는 규칙과 이에 따라 구현된 하이퍼큐브를 위해 경로배정 및 방송 알고리즘을 제안하였다. Bisectional 네트워크가 제공하는 기본적인 위상에는 선형 배열, 완전 이진 트리, 메쉬 구조가 있으며, 이와 함께 하이퍼큐브를 구현시키고 이에 해당하는 경로배정 알고리즘을 제안함으로써 bisectional 네트워크가 다중 컴퓨터의 네트워크 구조로 활용될 수 있게 하였다.

또한 bisectional 네트워크에서 메쉬 구조를 기초로 항상 하이퍼큐브의 구현이 가능함을 보였으며, 이에 대한 규칙을 기술하였다. 이러한 규칙을 기초로 생성된 하이퍼큐브를 위해 제안된 경로배정 및 방송 알고리즘은 기존의 하이퍼큐브에서 수행되는 각 알고리즘의 성질을 그대로 따른다. 경로배정 알고리즘은 근원지 노드와 목적지 노드 사이의 가장 짧은 경로로 일반적인 하이퍼큐브와 같이 경로배정이 이루어지며, 방송 알고리즘은 메시지를 하이퍼큐브의 차원인 d−1 단계내에 모든 노드에 정확히 한번씩 방송한다.

향후 연구 과제로는 구현된 하이퍼큐브에서 오류를 허용할 수 있는 경로배정 및 방송 알고리즘에 대한 연구가 있어야 하며, 기존의 하이퍼큐브에서 사용하는 여러 알고리즘의 모든 성질을 그대로 적용시킬 수 있는 방안에 대한 연구가 필요하다.

참 고 문 헌

1. D. K. Pradhan and S. M. Reddy, "A fault-tolerant communication architecture for distributed systems," IEEE Trans. Comput., vol.C-31, pp.863-870. Sept., 1982.
2. G. B. Adams III and H. J. Siegel, "The Extra Stage Cube : A Fault-Tolerant Interconnection Network for Supersystems," IEEE Trans. Comput., pp.443-454, May, 1982.
3. R. J. Mcmillen and H. J. Siegel, "Routing Schemes for the Augmented Data Manipulator Network in an MIMD Systems," IEEE Trans. Comput., pp. 184-196, Dec., 1982.
4. D. K. Pradhan, "Fault-tolerant multiprocessor link and bus network architectures," IEEE Trans. Comput., vol.34, pp.33-45, Jan., 1985.
5. Kai Hwang and Faye A. Briggs, Computer Architecture and Parallel Processing, Mcgraw Hill, 1984.
6. L. N. Bhuyan and D. P. Agrawal, "Design and performance of a general class of interconnection networks," IEEE Trans. Comput., vol.C-30, pp. 587-590, Aug., 1981.
7. L. D. Wittie, "Communication structures for large networks of microcomputers," IEEE Trans. Comput., vol.C-30, pp.264-273, Apr., 1981.
8. L. W. Hawkes, "A regular fault-tolerant architecture for interconnection networks," IEEE Trans Comput., vol.C-34, pp.677-680, July, 1985.
9. Arif Ghafoor, Theodore R. Bashkow, Imran Ghafoor, "Bisectional Fault-Tolerant Communication Architecture for Supercomputer Systems," IEEE Trans. Comput., vol.38, pp.1425-1446, NO.10, Oct., 1989.
10. E. Horowitz and A. Zorat, "The binary tree as an interconnection network : Applications to multiprocessor systems and VLSI," IEEE Trans. Comput., vol.C-30, pp.247-253, Apr., 1981.
11. Q. F. Stout, "Mesh connected computers with broadcast," IEEE Trans. Comput., vol.C-30 pp. 291-295, Apr., 1981.
12. J. R. Armstrong and F. G. Gray, "Fault diagnosis in a Boolean n-cube array of microprocessors," IEEE Trans. Comput., vol.C-30, pp.581-590,

Aug., 1981.

13. L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," IEEE Trans. Comput., vol.C-33, pp. 323-333, 1984.
14. J. P. Hayes, T. N. Mudge, and Q. F. Stout, "Architecture of a hypercube supercomputer," Int. Conf. Parallel Processing, pp.653-660, 1986.
15. John P. Hayes, Trevor Mudge, and Quentin F. Stout, "A Microprocessor based Hypercube Supercomputer," IEEE Micro, pp.6-17, Oct., 1986.
16. Abdol-Hossein Esfahnian, "On Enhancing Hypercube Multiprocessors," Int. Conf. Parallel Processing, pp.86-89, May, 1988.
17. Youcef Saad and Martin H. Schultz, "Data communication in Hypercubes," Journal of Parallel and Distributed Computing 6, pp.115-135, 1989.
18. Douglas M. Blough, Nader Bagherzadeh, and Rajeev Sehgal, "A New Fault-Tolerant Routing Algorithm for Hypercube system," Proceedings of the 3rd Annual Parallel Processing Symposium, pp.130-137, Mar., 1989.
19. Youcef Saad and Martin H. Schultz, "Topological Properties of Hypercubes," IEEE Trans. Comput., vol.37, pp.867-872, NO.7, July, 1988.
20. Howard P. Katseff, "Incomplete Hypercube," IEEE Trans. Comput., vol.37, NO.5, May, 1988.
21. B. W. Arden and H. Lee, "Analysis of chordal ring network," IEEE Trans. Comput., vol.C-30, pp.291-295, Apr., 1981.

최 창 훈(Chang Hoon Choi)정회원
1988년 2월 : 명지대학교 전자계산학과 졸업
1990년 2월 : 서강대학교 대학원 전자계산학과(공학석사) 졸업
1990년 1월~9월 : 대우통신 기술개발부 근무
1992년~현재 : 서강대학교 대학원 전자계산학과 박사과정 재학중
※주관심분야 : Computer Architecture, parallel processing system

정 영 호(Young-Ho Jeong)정회원
1990년 2월 : 서강대학교 전자계산학과 졸업
1992년 2월 : 서강대학교 전자계산학과 대학원 졸업
1992년 2월~현재 : 삼성전자(주) 기술총괄 근무

김 성 천(Sung-Chun Kim) 정회원
1975년 : 서울대학교 공과대학 공업교육학(전기전공)학사
1976년~1977년 : 동아컴퓨터(주) Sys. Eng.
1977년~1978년 : 스페리 유니백 Sales Rep.
1978년 : Wayne State Univ. 컴퓨터공학 석사
1982년 : Wayne State Univ. 컴퓨터공학 박사
1982년~1984년 : 캘리포니아주립대 조교수
1984년~1985년 : 금성반도체(주) 책임연구원
1986년~1989년 : 서강대학교 공과대학 전자계산소 부소장
1989년~1991년 : 서강대학교 공과대학 전자계산학과 학과장
1985년~현재 : 서강대학교 공과대학 전자계산학과 조교수(1985. 8~1987. 8), 부교수(1987. 9~1992. 8), 교수(1992. 9~현재)
1989년~현재 : 한국정보과학회 병렬처리시스템 연구부 부위원장(1989~1993), 위원장(1994~현재), 대한전자공학회 및 한국통신학회 논문지 편집위원(1991~현재, 1993~현재)
※주관심분야 : 병렬처리시스템(Parallel Computer Architecture, Interconnection Network), Computer Network