웜홀 라우팅을 지원하는 스타그래프 네트워크에서의 전 포트 브로드캐스팅 알고리즘

(All-port Broadcasting Algorithms on Wormhole Routed Star Graph Networks)

김 차 영 † 이 상 규 †† 이 주 영 †††
(Cha-Young Kim) (Sang-Kyu Lee) (Ju-Young Lee)

요 약 병렬 처리 시스템의 상호연결 네트워크로써 스타 그래프 구조가 그간 널리 사용되어 왔던 하이퍼큐브에 비해 지름 및 차수 등의 특성에서 우수한 성능을 보임으로 인해 최근 많은 연구자들의 관심을 받고 있다. 스타 그래프 네트워크에서 여러 가지 통신 문제들이 연구되어지고있는데 그러한 통신 문제 중에 가장 기본이 될 수 있는 문제 중의 하나가 브로드캐스팅이다. 본 논문에서는 웜홀라우팅을 지원하는 스타 그래프 네트워크 시스템에서의 브로드캐스팅 문제를 다룬다. 웜홀라우팅을 사용하는 네트워크에서는 전송 노드간의 거리보다 전송 시 링크충돌을 최소화하는 것이 전체 통신 시간을 줄이는 중요한 요소가 되는데, 본 논문에서는 스타 그래프 네트워크에서의 해밀토니안 경로를 이용하여 링크 충돌 없이 n 차원 스타 네트워크에서 ($\lceil \log_n n! \rceil + 1$) 통신스텝에 전체 브로드캐스팅이 완료되는 알고리즘을 제시한다. 이는 이론적 하한값 $\lceil \log_n n! \rceil$ 에 근접한 결과로 기존의 n-1 통신 스텝이 걸리는 알고리즘 보다 향상된 결과이다.

키워드 : 브로드캐스팅, 스타그래프, 웜홀 라우팅

Abstract Recently, star graph networks are considered as attractive alternatives to the widely used hypercube for interconnection networks in parallel processing systems by many researchers. One of the fundamental communication problems on star graph networks is broadcasting. In this paper, we consider the broadcasting problems in star graph networks using wormhole routing. In wormhole routed system, minimizing link contention is more critical for the system performance than the distance between two communicating nodes. We use Hamiltonian paths in star graph to set up link-disjoint communication paths. We present a broadcast algorithm in n-dimensional star graph of $N(=n!)$ nodes such that the total completion time is no larger than $\lceil \log_n n! \rceil + 1$ steps, where $\lceil \log_n n! \rceil$ is the lower bound. This result is significant improvement over the previous n-1 step broadcasting algorithm.

Key words : Broadcasting, Star graph, Wormhole Routing

1. 서 론

스타 그래프(star graph)는 병렬처리 시스템을 위한 네트워크 구조로써 하이퍼큐브(hypercube)에 효과적으로 대처할 수 있는 네트워크 구조로 많은 학자들로부터 주목을 받고 있다[1][2]. n 차원 하이퍼큐브(n-cube)와 n 차원 스타그래프(n-star)를 비교해 보면, n-cube는 2^n 개의 노드들로 구성된 차수(degree)가 n이고 지름(diameter)이 n인 네트워크이며, n 차원 스타그래프(n-star)는 $n!$ 개의 노드들로 연결되어 있으며 차수는 $n-1$, 지름은 $\lfloor 3(n-1)/2 \rfloor$ 인 네트워크이다. 여기서 차수란 각 노드에 연결된 링크(link)의 개수이며, 지름은 네트워크 상에서 가장 멀리 떨어진 두 노드 사이의 거리이다. 비슷한 개수의 노드를 갖는 하이퍼큐브와 스타그래프 네트워크를 비교해 보면, 스타그래프가 하이퍼큐브 보다 더 낮은 차수와 작은 지름으로 구성됨을 표[1]에서 볼 수 있다. 낮은 차수와 작은 지름은 적은 비용을 뜻하므로, 병렬컴퓨터를 크게 대량적으로 구현하기에 스타그래프가 더 효과적이라고 볼 수 있다.

† 비 회 원 : 숙명여자대학교 컴퓨터과학과
chayoung@cs.sookmyung.ac.kr
†† 종신회원 : 숙명여자대학교 컴퓨터과학과 교수
sanglee@sookmyung.ac.kr
††† 종신회원 : 덕성여자대학교 전산학과 교수
jylee@namhae.duksung.ac.kr
논문접수 : 1999년 7월 7일
심사완료 : 2001년 10월 24일

표 3 *n-star* 와 *n-cube*의 비교

	Number of Vertices	Degree	Diameter	Average Distance
5-star	120	4	6	3.7
7-cube	128	7	7	3.5
7-star	5,040	6	9	5.9
12-cube	4,096	12	12	6
9-star	362,880	8	12	8.1
18-cube	262,144	18	18	9

현재 스타그래프 토폴로지 네트워크에 관련한 여러 가지 연구가 이루어지고 있는데, 그 주된 문제 중 하나가 여러 가지 통신 문제이고 그 중 가장 기본적인 것이 브로드캐스트(broadcast) 문제이다. 어떤 특정한 원시노드(source node)에서 네트워크 상에 있는 다른 모든 노드에게 메시지를 보내는 브로드캐스팅은 병렬컴퓨터 시스템에서 빈번하게 사용되어지는 통신형태로 브로드캐스트 오버헤드(overhead)를 감소시키기 위해서는 효과적인 브로드캐스트 알고리즘이 필요하다. 지금까지 여러 가지 형태의 상호 연결 망에서의 브로드캐스트 알고리즘[3,4,5]들이 개발되었는데, 본 논문에서는 전 포트(all- port) 통신을 지원하고 웜홀라우팅(wormhole routing)을 사용하는 스타 그래프 네트워크에서의 최적 통신시간에 근접한 알고리즘을 제안한다. 본 논문의 구성은 2장에서 시스템 모델 및 브로드캐스트 문제를 설명하고, 3장에서 브로드캐스트 알고리즘을 제시하고, 4장에서 결론을 맺는다.

2. 모델설명 및 문제

n 차원 스타그래프(n-*star*, 혹은 S_n)는 $n!$ 개의 노드와 $n!(n-1)/2$ 개의 링크(link)들로 이루어지며, 노드와 링크에 대해 대칭적 성질을 갖는다. S_n의 각 노드들은 n 개의 서로 다른 심볼들의 순열(permutation)로 표시되는 레이블(label, 혹은 주소)을 갖으며 $n-1$ 개의 서로 다른 노드들과 링크로 연결되어 있어서 $n-1$ 차수의 정규 그래프(regular graph)를 이룬다. 이 때 레이블이 $a_1a_2\ldots a_n$인 어떤 노드와 링크로 연결되는 노드들은 첫 번째 위치에 있는 심볼 a_1과 $i(2\le i\le n)$ 번째 위치에 있는 심볼 a_i를 서로 교환한 레이블을 갖는 $n-1$ 개의 노드들이며 각각을 생성자(generator) g_i로 연결되었다고 말한다. 예를 들어 4 차원 스타 그래프 S_4의 4 개의 서로 다른 심볼을 a, b, c, d라고 했을 때 S_4는 4 개의 심볼들의 순열을 레이블로 갖는 24 개의 노드들로 이루어져 있고 각각의 노드들은 g_2-g_4 생성자들로 연결된 3 개의 노드들과 이웃하게 된다. 그림 1은 4 차원 스타 그래프 S_4를 보여주는데, 여기에서 네 개의 육각형은 각각 마지막(4번째) 위치의 심볼을 a, b, c, d로 고정시킨 레이블을 갖는 3 차원 서브스타들로 배열로 되어 있음을 볼 수 있다.

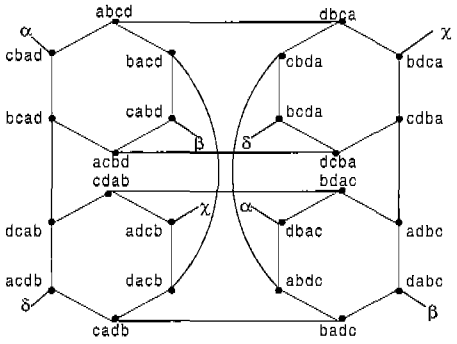

그림 1 4 차원 스타그래프 S_4

일반적으로 병렬컴퓨터는 통신포트가 1 개인 단일 포트(one-port) 방식이거나, TMC, CM-5, Intel/CMU, iWARP 등과 같은 전 포트(all-port)방식이다. 단일 포트 모델에서는 어느 하나의 노드에서 한 번에 보낼 수 있는(받을 수 있는) 메시지 패킷은 많아야 1 개인 반면, 다중포트 모델에서는 어느 하나의 노드에서 동시에 보낼 수 있는(받을 수 있는) 메시지 패킷은 포트의 수만큼 가능하다. 그리고, 어떤 스위칭(switching)방법을 사용하느냐에 따라서도 통신 알고리즘은 많은 영향을 받는데, 저장전달 방식의(store-and-forward) 라우팅에서는 한 개의 메시지를 여러 개의 패킷들로 나누어, 한 스텝(스텝은 전송노드에서 출발한 메시지가 수신노드에 도착하는데 소요되는 시간을 나타내는 단위 통신시간을 의미)에 이웃 노드에게 패킷을 보낼 수 있다. 그러므로, 브로드캐스팅에 걸리는 전체 통신시간은 경로의 길이에 비례한다. 그러나 최근에 가장 많이 사용하며 전송거리에 덜 민감하여 보다 빠른 속도의 전송을 보여주는 웜홀라우팅(wormhole-routing) 방법[6]은 메시지를 플릿(flit)이라는 작은 단위로 나누어, 데이타 플릿들을 보내기 전에 헤더플릿이 전송되며 지정해 놓은 중복되지 않는(link-disjoint) 경로들을 통해서 파이프라인(pipeline) 형태로 데이타 메시지를 전송한다. 이러한 시스템에서는 메시지 크기에 비해 플릿의 크기가 아주 작은 경우(대부분의 경우 이러한 관계를 갖는다.), 전달 경로상의 거리는 실제통신의 시작에서 끝날 때까지의 시간인 통신

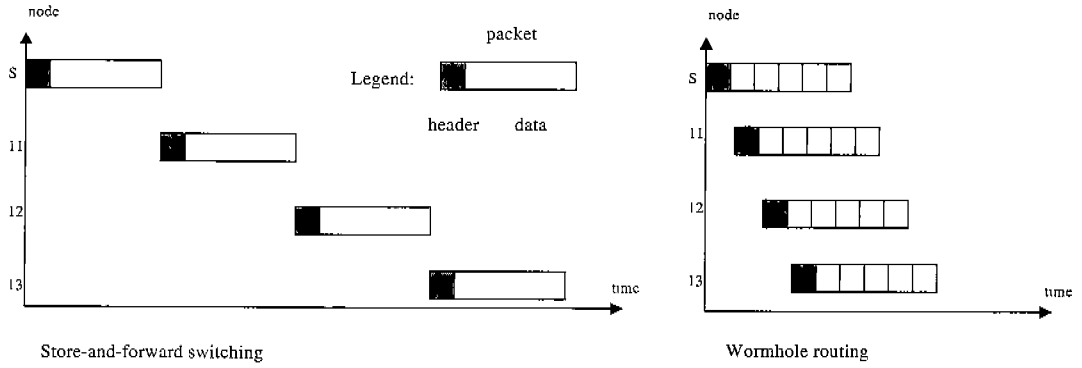

그림 2 스토어-앤-포워드 라우팅과 웜홀라우팅의 비교

레이턴시를 고려할 때 거의 무시할 수 있다. 즉, 웜홀라우팅을 사용하는 시스템에서의 통신문제에서는 통신을 주고받는 두 노드사이의 거리보다는 그 두 개의 노드사이의 통신경로에 같은 시간에 일어나는 다른 통신들의 경로 중복여부가 전체의 통신시간에 결정적 영향을 줄 수 있다. 중복이 일어나면 다른 하나는 큐에 기다렸다가 실행되기 때문이다. 따라서 이러한 시스템에서의 통신문제들은 경로의 길이보다는 같은 시간에 일어나는 통신경로들의 중복을 최소화하는데 그 초점을 맞추어야한다. 그림 2에서 두 가지 라우팅의 차이를 볼 수 있다[6]. S는 원시 노드를, I1, I2, I3은 중간 노드들을 나타낸다.

기존관련 연구로는 단일 포트 스타 네트워크 모델에서의 브로드캐스팅 알고리즘을 제안한 Latifi등[7]의 논문이 있고 본 논문이 고려하는 시스템과 같은 전 포트 방식의 모델에서의 브로드캐스팅 관련 연구로는 Akers 등[1]에 본 논문에서는 웜홀라우팅과 전 포트를 지원하는 스타그래프 네트워크 시스템에서의 브로드캐스팅 문제를 다룬다. 의해 제안된 통신시간이 $n-1$ 스텝이 걸리는 알고리즘이 있다. 이 알고리즘에서는 스타 그래프의 재귀적 구성(recursive construction) 방법을 브로드캐스팅에 그대로 적용하였는데, n 차원 스타 그래프에서 첫 번째 스텝에 원시노드가 자신의 $n-1$ 개의 링크를 이용해 자신이 속하지 않은 나머지 $n-1$ 개의 $n-1$ 차원 서브 스타에 있는 하나의 노드에 메시지를 보내고, 두 번째 스텝에는 메시지를 갖고 있는 각각의 노드가 자신의 $n-2$ 개의 링크를 사용해 자신이 속한 $n-1$ 서브 스타 안의 $n-2$ 개의 $n-2$ 서브 스타의 하나의 노드들에 동시에 메시지를 보낸다. 이와 같은 방법을 반복하여 마지막 $n-1$ 번째 스텝에서는 메시지를 가지고 있는 모든 노드들이 자신의 1 개의 링크를 사용하여 나머지 노드들에게 메시지를 보낸다. 이것은 전형적인 하이퍼큐브에서의 브로드캐스팅방법과 일치하는 방법이다. 하이퍼큐브에서의 기본적 브로드캐스팅 방법에서와 마찬가지로 메시지를 전달하는 노드들은 매번 자신이 갖고 있는 모든 링크를 사용하지 못하고 각 스텝이 지날 때마다 하나씩 줄여서 사용해야하기 때문에 스텝이 진행될수록 낭비되는 통신링크가 증가됨을 알 수 있다. 본 논문에서는 이점을 고려해 사용할 수 있는 링크를 매 스텝 최대로 사용하여 메시지를 전달함으로서 전체의 브로드캐스팅 통신시간을 최소화하는 알고리즘을 제안한다.

3. 브로드캐스팅 알고리즘

본 장에서는 웜홀라우팅을 지원하는 n 차원 스타그래프 S_n에서 해밀토니안경로를 사용하여 최적+1의 통신스텝 안에 임의의 원시 노드로부터 다른 모든 노드로 브로드캐스팅을 할 수 있는 알고리즘을 제안한다. 웜홀라우팅을 지원하는 네트워크에서는 링크 충돌(link contention)을 최소화하는 것이 중요한데, 해밀토니안 경로를 스타그래프에 임베딩(embedding)하여 각각의 노드마다 그 경로의 할당된 구간만을 사용하도록 하여, 어떠한 통신도 각기 다른 경로를 사용함으로써 링크 충돌을 없앰으로 해서 링크 정체가 일어나지 않게 하는 방법을 본 논문에서는 사용하였다. 스타그래프에서의 해밀토니안 경로를 찾는 방법은 이미 널이 알려져 있다[2][8]. 본 장에서는 먼저 [8]에서 제시한 스타그래프에서 해밀토니안 경로를 찾는 방법을 간략히 소개하고, 본 논문이 고려하는 네트워크 모델에서의 브로드캐스트 최적 통신시간을 정리하고 최적에 거의 근접한 브로드케스트 알고리즘을 제시하고자 한다.

3.1 스타그래프에서의 해밀토니안 사이클

[8]에서 $HP(i,s')$는 노드 s'에서 시작해서 $g_i(s')$ $(2\le i\le n)$의 노드로 끝나는 스타그래프의 해밀토니안 경로, $I(HP(i,s'),k,l)$은 $HP(i,s')$에 의해 생기는 노드 시퀀스의 각각의 노드들에 $(k-1)$ 번째 심볼 다음에 심볼 l을 넣어 생기는 노드 시퀀스, $THP(k)$는 심볼집합 $\{a_2,a_3,\ldots,a_k\}$을 가진 노드 $a_2a_4a_5\ldots a_ka_3$에서 시작해서, 노드 $a_ka_2a_3a_4\ldots a_{k-1}$로 끝나는 스타그래프의 해밀토니안 경로로 정의되어 있다.

<해밀토니안 경로 만드는 방법>

procedure $THP(k+1)$

begin

심볼 a_{k+1} , $HP(i,s'')(2\le i\le|s''|=k-1)$ 로써,

$p_1 = I(HP(2,a_2a_5a_6\ldots a_{k-1}a_3),2,a_4)$
$= a_2a_4a_5a_6\ldots a_{k+1}a_3,\ldots,a_5a_4a_2a_6\ldots a_{k+1}a_3$

$p_2 = I(HP(3,a_4a_2a_6\ldots a_{k+1}a_3),2,a_5)$
$= a_4a_5a_2a_6\ldots a_{k+1}a_3,\ldots,a_6a_5a_2a_4\ldots a_{k+1}a_3\ldots$

$p_{k-2} = I(HP(k-1,a_ka_2a_3a_5\ldots a_{k-1}),2,a_{k+1})$
$= a_ka_{k+1}a_2a_4a_5\ldots a_3,\ldots,a_3a_{k-1}a_2a_4a_5\ldots a_k$

$p_{k-1} = I(HP(2,a_{k+1}a_2a_4a_5\ldots a_k),2,a_3)$
$= a_{k+1}a_3a_2a_4a_5\ldots a_k,\ldots,a_2a_3a_{k+1}a_4\ldots a_k$

$p_k = I(HP(2,a_3a_{k+1}a_4a_5\ldots a_k),2,a_2)$
$= a_3a_2a_{k+1}a_4\ldots a_k,\ldots,a_{k+1}a_2a_3a_4\ldots a_k$

end

procedure $HP(k,s)$

begin

심볼 a_k , $HP(k-1,s')$, $THP(k)$ 로써,

$p_1 = I(HP(k-1,a_1a_2a_3\ldots a_{k-1}),k,a_k)$
$= a_1a_2a_3\ldots a_k,\ldots,a_{k-1}a_2a_3\ldots a_{k-2}a_1a_k$

$p_2 = I(HP(k-2,a_ka_2a_3\ldots a_{k-2}a_1),k,a_{k-1})$
$= a_ka_2a_3\ldots a_{k-2}a_1a_{k-1},\ldots,a_{k-2}a_2a_3\ldots a_ka_1a_{k-1}\ldots$

$p_{k-2} = I(HP(2,a_4a_2a_5a_6\ldots a_ka_1),k,a_3)$
$= a_4a_2a_5\ldots a_ka_1a_3,\ldots,a_2a_4a_5\ldots a_ka_1a_3$

$p_{k-1} = I(HP(k-1,a_3a_4a_5\ldots a_ka_1),k,a_2)$
$= a_3a_4a_5\ldots a_ka_1a_2,\ldots,a_1a_4a_5\ldots a_ka_3a_2$

$p_k = I(THP(k),k,a_1)$
$= a_2a_4a_5\ldots a_ka_3a_1,\ldots,a_ka_2a_3a_4\ldots a_{k-1}a_1$

end

begin

$s' = a_1a_2a_3$

$HP(3,s')=$

$a_1a_2a_3, a_2a_1a_3, a_3a_1a_2, a_1a_3a_2, a_2a_3a_1, a_3a_2a_1$

$THP(4)$ =

$a_2a_4a_3, a_3a_4a_2, a_4a_3a_2, a_2a_3a_4, a_3a_2a_4, a_4a_2a_3$

for $k=4$ to n

$HP(k,s)$를 만든다.

$s' = p_1,p_2,\ldots,p_k$

$THP(k+1)$를 만든다.

end

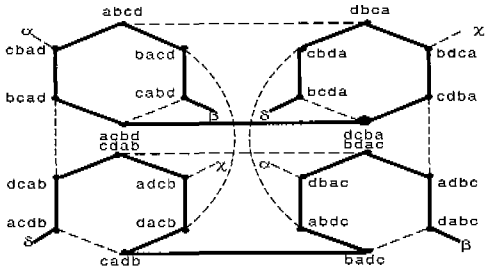

그림 3 노드 dcba에서 시작해서 dcba로 끝나는 S_4의 해밀토니안 경로

위의 알고리즘을 사용하여 S_4에서 해밀토니안 경로를 만드는 방법을 보면, S_3의 해밀토니안 경로가 사용됨을 볼 수 있다. 그림 3에서, 원시 노드 $s='dcba'$일 때, 심볼 dcb만을 취해, $HP(3,dcb)$= dcb, cdb, bdc, dbc, cbd, bcd와 $THP(4)$= cab, bac, abc, cba, bca, acb를 만든다. 그리고 심볼 a, $HP(3,dcb)$와 $THP(4)$로 $HP(4,dcba)$를 만들면, p_1 = $I(HP(3,dcb),4,a)$ = dcba, cdba, bdca, dbca, cbda, bcda, p_2 = $I(HP(2,acd),4,b)$ = acdb, dcab, cdab, adcb, dacb, cadb, p_3 = $I(HP(3,bad),4,c)$ = badc, abdc, dbac, bdac, adbc, dabc, p_4 = $I(THP(4),4,d)$ = cabd, bacd, abcd, cbad, bcad, acbd이다. p_1의 첫 번째 노드 dcba의 $g_4(dcba)=acbd$ 임으로, 정의한 바와 같이 p_4 의 마지막 노드이다. p_1,p_2,p_3,p_4를 모두 연결하면, S_4의 해밀토니안 경로가 된다. 그리고, p_4의 마지막 노드 acbd와 원시노드 dcba는 생성자 g_4로 연결되어 있으므로, 이 둘을 연결하면 해밀토니안 사이클이 된다. 본 논문에서는 해밀토니안 사이클 중에서 전송하려는 메시지 가지고 있는 노드의 이웃 노드를 시작점으로 한 해밀토니안 경로를 고려하도록 한다.

3.2 스타그래프에서의 브로드캐스트 통신 하한값

본 논문이 고려하는 모델에서의 브로드캐스팅 통신 하한값과 최적 통신시간은 다음과 같다.

보조정리 1. n 차원 스타그래프 S_n에서 브로드캐스팅 통신은 최소 $\lceil \log_n n! \rceil$ 통신 스텝이 걸린다.

증명 : 스타 그래프의 원시노드가 첫 번째 브로드캐스팅 스텝에 메시지를 보낼 수 있는 노드의 최대 수는 노드가 가지고 있는 링크의 수인 $n-1$이다. 그래서, 브로드캐스팅 한 스텝이 끝난 후 메시지를 가지고 있는 노드들은 원시노드와 원시노드로부터 첫 번째 스텝에 메시지를 받은 $n-1$ 개의 노드를 합쳐 $1*(n-1)+1=n$ 개가된다. 그 다음 스텝에 이 n 개의 노드들이 각기 자신이 줄

수 있는 최대 수인 $n-1$ 개의 다른 노드에 메시지를 전달 할 수 있다면 두 번째 스텝이 끝난 후 메시지를 가질 수 있는 최대 노드 수는 원래 메시지를 갖고 있던 n 개의 노드를 포함해서 $n*(n-1)+n=n^2$이 된다. 이런 계속 전개해보면 t 스텝 후에 메시지를 갖게 되는 노드들은 최대 n^t가 됨을 알 수 있고, 브로드캐스팅에 필요한 통신 스텝 t는 전체 노드에 전해지는데 필요한 조건 $n^t \geq n!$을 만족해야 하고, 따라서 브로드캐스팅에 필요한 통신시간의 스텝 수 t는 $t \geq \lceil \log_n n! \rceil$이 된다. ■

논문에서는 S_n의 최적 브로드캐스팅 스텝을 T_{opt}라하고, $T_{opt} = \lceil \log_n n! \rceil$라 하겠다.

3.3 최적 + 1 브로드캐스팅 알고리즘

그림 1의 스타그래프 노드의 배열에서 원시 노드를 *dcba*라 할 때 각 노드의 레이블에서 $a_1(=d)$의 위치에 따라 구분하여 정리하면 그림 4와 같이 나타낼 수 있다. 이 때 S_n은 그림 4와 같이 $n-1(=3)$ 개의 $S_{n-1}(=S_3)$들과 원시 노드를 포함하여 $(n-1)!(=3!)$ 개의 노드를 갖는 독립 집합(independent set)($\simeq I_s$)으로 구성할 수 있다. 여기에서 I_s는 심볼 $a_1(=d)$이 레이블의 첫 번째 위치에 오는 노드들의 집합이다. 이 때, I_s의 각 노드는 $n-1(=3)$ 개의 $S_{n-1}(=S_3)$ 서브스타의 정확히 하나의 노드들과 서로 연결되어 있으며 I_s 노드의 이웃 노드들의 집합은 I_s를 제외한 나머지 $n-1(=3)$ 개의 $S_{n-1}(=S_3)$ 서브스타 전체가 됨을 알 수 있다. 이러한 재배열을 이용하여 본 논문에서 제시하는 알고리즘은 만약 어떤 t 스텝 안에 I_s의 모든 노드가 메시지를 다 받으면, 바로 다음의 한 번의 스텝 후에 나머지 다른 모든 노드들도 I_s의 노드들로부터 모든 링크를 전부 사용하여 링크 충돌 없이 메시지를 받을 수 있다는데 착안을 하였다. 본 논문에서는 최적 ($\lceil \log_n n! \rceil$) 스텝 안에 I_s의 노드들에게 메시지를 전하고 있다.

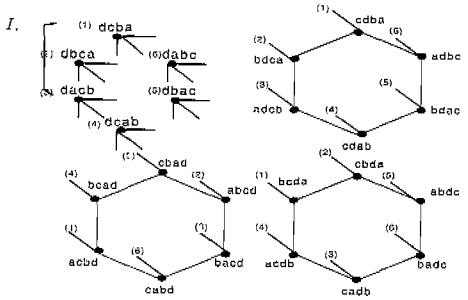

그림 4 Independent Set으로 표현된 S_4

그리고, 본 논문의 알고리즘이 필요로 하는 해밀토니안 경로는 그림 4에서처럼 I_s로 구성된 S_n에서 만들어야 한다. 왜냐하면, I_s의 어떤 한 노드에서 I_s 안의 다른 노드들로 메시지를 보낼 때에 서로 연결된 링크가 없어서 $n-1$ 개의 S_{n-1} 서브스타들의 링크들을 통해 메시지를 충돌 없이 보내야 하기 때문이다. 그래서, 그림 4의 S_4에서 원시 노드를 dcba로 하여 본 논문이 필요로 하는 해밀토니안 경로를 만들면, $HP(3, cba)$ = cba, bca, acb, cab, bac, abc이다. 그리고 첫 번째 심볼인 $a_1=d$를 그 노드들의 레이블의 두 번 째 위치에 놓으면, cdba, bdca, adcb, cdab, bdac, adbc가 된다. 그림 4의 심볼 d가 두 번 째에 오는 노드들의 집합과 같다. 이 노드들이 I_s의 각각의 노드에서 링크 2를 통해서 갈 수 있는 $n-1(=3)$ 차원 서브스타 노드 모임이다. 여기에서 주의를 둘 것은 $n-1$ 차원 스타그래프의 해밀토니안 경로를 만드는 방법이 각각의 서브스타 마다 똑같이 적용될 수 있다는 점이다. 따라서, 심볼 d를 세 번째, ..., n 번째에 각각 놓으면, 각각의 노드 시퀀스들은 I_s의 각각의 노드에서 링크 3을 통해서, ..., 링크 n을 통해서 갈 수 있는 노드들의 모임이 된다.

이러한 구성에서 $n-1$ 개의 S_{n-1}들과 I_s의 이웃노드들의 합집합이 같으며 I_s의 각각의 노드는 한 개의 링크도 중복되는 것 없이 각각의 S_{n-1}의 노드들에 연결되어 있다. 따라서, I_s의 모든 노드들이 어떤 시간 안에 메시지를 모두 받을 수 있다면 바로 다음의 한번의 스텝이 지나면 나머지 다른 모든 노드들도 메시지를 받을 수 있다. 본 장에서는 이러한 특성과 각 서브스타의 해밀토니안 경로를 이용하여 충돌 없는(contention free) 브로드캐스팅 알고리즘을 제시한다.

알고리즘에서 쓰이는 정의들은 다음과 같다. T_{opt}는 $\lceil \log_n n! \rceil$, $g_i(r)$은 임의의 노드 r에서 첫 번째에 있는 심볼과 l $(2 \leq l \leq n)$번째에 있는 심볼을 바꿨을 때 만들어지는 노드, 노드 r의 link i는 노드 r과 노드 $g_i(r)$을 연결하는 링크, SS^l_{n-1}는 I_s의 각각의 노드 r에서 $g_l(r)$의 노드들로만 만들어지는 S_{n-1} 서브스타, target_node(t)는 t 번째 브로드캐스팅 스텝에 새로 메시지를 받을 I_s 안의 노드들, send_node(t)는 t 번째 브로드캐스팅 스텝이 시작될 때 이미 메시지를 갖고 있는 I_s 안의 노드들이라 정의한다. 따라서, 매 브로드캐스팅 스텝은 send_node(t)의 각각의 노드들이 아직 메시지를 갖고 있지 않은 I_s 안의 어느 노드들에게 중복되지 않는 경로를 통하여 메시지를 전달하는가에 대한 결정으로 볼 수 있고 이때 새로 메시지를 받게되는 I_s 안의 노드들이 target_node(t)를 이룬다. send_node(t)

는 매 스텝이 지날 때마다 이전의 send_node(t)에 속해 있던 노드들과 바로 이전 스텝에서의 target_node (t)에 속한 노드의 합집합을 이루며 축적되고 target_ node(t)는 매 스텝마다 새로이 구해지게 된다. t 번째 스텝에서의 target_node(t)는 n-1 개의 서브스타를 순차적으로 거치면서 다시 send_node(t)의 노드 하나 하나에 대하여 순차적으로 구해간다. 하나의 send_node(t) 안의 노드 r이 고려될 때 r로부터 메시지를 받게되는 target_node(t)의 노드들의 선택은 2부터 n까지의 각 링크 l을 통해 전달할 수 있는 노드들이 존재하는가를 순차적으로 검사하여 전달 가능한 노드를 포함하게 되는데, 이때 각 링크 l을 통한 전달이란 $g_l(r)$의 노드들을 통한 경로로 이루어지며 이때 모든 send_node(t)안의 노드들의 전달에 중복이 없어야 한다.

이를 보장하기 위해 본 논문에서는 해밀토니안 경로를 사용한다. 각각의 서브스타의 하밀토니안 경로는 다시 send_node(t)구간들과 각각의 send_node(t) 노드 r에 대한 target_node(t)구간들로 나누어지는데 두 가지 구간에 대한 정의는 다음과 같다. 각각의 서브스타 SS^l_{n-1} $(2 \le l \le n)$들의 해밀토니안 경로들을 send_node(t)에 속한 노드들의 이웃 노드들을 경계로 나누어진 구간을 서브스타 SS^l_{n-1}의 send_node(t)구간이라 한다. 그림 5에 send_node(t)에 5개의 노드 $r_1, \cdots, r_5$이 있을 때 그로 인해 만들어지는 5개의 구간 $c_1, \cdots, c_5$들이 설명되고 있다. 이 경우 5개의 send node r_i들이 t 번째 브로드캐스팅 스텝에 서브스타 SS^l_{n-1}을 이용하여 아직 메시지를 받지 않은 I_s의 노드 중에 하나를 선택하여 메시지를 전달할 때 각각 c_i 구간만을 이용하여 메시지를 전달한다면 서브스타 SS^l_{n-1}에서 t 번째 스텝에 일어나는 모든 통신경로에 절대로 중복이 일어날 수 없다는 것을 쉽게 확인할 수 있다. send_node(t)안의 각각의 노드들은 링크 l에 따라 서브스타 SS^l_{n-1} $(2 \le l \le n)$들에서

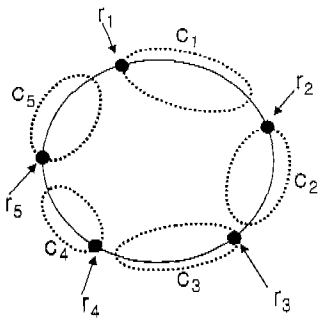

그림 5 해밀토니안 경로상의 send_node(t)구간

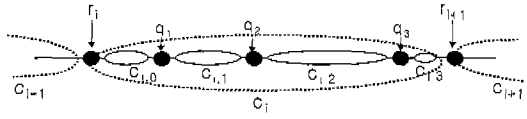

그림 6 send_node(t)구간 Ci와 target_node(t)구간들

순차적으로 자신이 메시지를 전달할 target_node(t)에 포함될 노드를 구하게 되는데, 한번 메시지를 받은 노드는 어느 한 스텝에만 다른 메시지를 받지 않은 노드에게 메시지를 보내는 것이 아니고 가능한 한 많은 스텝에서 계속 메시지를 전달해야 전체 브로드캐스팅 시간을 단축할 수 있음으로 노드의 선택 시 이를 고려해야 한다. 전체 브로드캐스팅 스텝의 단축을 고려하기 위해 노드 r_i가 각각의 서브스타에서 스텝 t에 메시지를 보낼 노드를 선택할 때 자신에 할당된 구간 c_i를 다시 이번 스텝에 새로이 추가되고 있는 target_node(t) 노드들의 이웃 노드들 중 c_i 구간에 위치하는 노드들이 $q_1, q_2, \cdots, q_k$라 할 때 이 노드들을 경계로 나누어 이 구간들을 $c_{i,1}, c_{i,2}, \cdots, c_{i,k}$라하고 이 구간을 노드 r_i의 서브스타 SS^l_{n-1}에서의 target_node(t)구간이라 하자. 그림 6에서 어느 서브스타에서의 노드 r_i의 target_node(t)구간들이 설명되고 있다. 그림 6에서 $q_1 - q_3$는 현재까지 구해진 send_node(t) 안의 노드들의 이웃노드들 중 구간 c_i에 있는 노드들을 나타내고 $c_{i,0} - c_{i,3}$들은 $q_1 - q_3$를 경계로 만들어진 target_node(t)구간 들이다. 그 구간들 중에서 가장 긴 구간을 SS^l_{n-1}에서 r_i에 대한 max_interval (r_i)라고 하자. 현재 send_node(t)에 포함되는 $q_1 - q_3$노드들은 다음 번 스텝에서는 메시지를 전달할 수 있는 노드들이 되고 다음번 스텝에 자신에 해당하는 $c_{i,0} - c_{i,3}$의 구간을 이용하여 메시지를 전달할 것이다($c_{i,0}$구간은 노드 r_i가 다음 스텝에 이용하는 구간이 된다.). 따라서 전체적인 브로드캐스팅 스텝을 최소화하기 위해서는 매 스텝 max_interval 구간들의 크기를 최소화 해야하고 그러기 위해서는 r_i노드가 각각의 서브스타에서 target node로 선택하야 하는 노드들은 max_interval(r_i)구간의 1/2 지점($\lceil \frac{\text{첫노드색인} + \text{마지막노드색인}}{2} \rceil$)에 있는 노드를 선택하는 것이 될 것이다. 다음은 이를 정리한 알고리즘이다.

본 논문의 브로드캐스팅 알고리즘은 최적 통신시간 (T_{opt}) 동안 각 스텝마다 각각의 링크 $l(2 \le l \le n)$을 순차적으로(sequentially) 거치면서 목적지 노드들을 한 개 씩 계산해 낸다. 그래서, 전 단계의 통신시간에 목

<알고리즘 1>

input: Star Graph S_n , source node s
output: Broadcast_Tree (node, edge, weight)
// edge는 어떤 send_node r에서 어떤 target_node q들로 메시지를 보내는가에 대한 정보를 갖고 있고, weight는 그 메시지가 어느 스텝에 보내어 지는가에 대한 정보를 갖고 있다. 따라서, 프로그램이 끝나고 나면, Broadcast_Tree에는 '어떤 node r에서 어떤 node q들로 언제 메시지를 보내는가'에 대한 브로드캐스팅 스케쥴(schedule)이 저장 된다. //
Step 0. Broadcast_Tree의 노드로 원시노드를 포함시킨다.
Step 1. S_n의 hamiltonian cycle을 만든다.
Step 2. for $t=1$ to T_{opt}
for $l=2$ to n
for $i=1$ to $|send\_node(t)|$ // send_node(t) 노드들에 인덱스가 순차적으로 부여했다고 가정 한다
SS'_{n-1}에서 send_node(t)를 고려하여 send_node(t)구간 중에 r_i에 할당되는 구간 c_i를 구하고 이 구간에서 다시 target_node(t)에 있는 노드들을 고려하여 target_node(t)구간들을 구하여 max_interval(r_i)를 계산한다. 그 구간의 ½위치에 해당하는 노드 h의 $g_l(h)=q$를 구하여 노드 q가 target_node(t)에 포함되어 있지 않으면 Broadcast_Tree의 *node*에 q를, *edge*에 (r_i, q)를 포함시키고, *weight*(r_i, q) = t로 한다. q를 target_node(t)에 저장한다.
end //for i
end //for l
end //for t
Step 3. Broadcast_Tree의 *node*에 있는 모든 노드 r에 대해, $g_l(r)$ $(2 \le l \le n)$을 구하여 *node*에 $g_l(r)$을 *edge*에 포함시키고 *weight*$(r, g_l(r)) = T_{opt}+1$로 한다.

적지 노드들로 계산된 노드들은 현재의 통신시간에 메시지를 보낼 수 있는 노드들이 되고, 같은 스텝에서 전 단계 링크에서 선택되어진 목적지 노드들은 현재 링크에서 목적지 노드들을 선택하는 각각의 구간을 세분화한다. 따라서, 목적지들을 선택할 수 있는 구간은 스텝이 지날수록 점점 줄어든다.

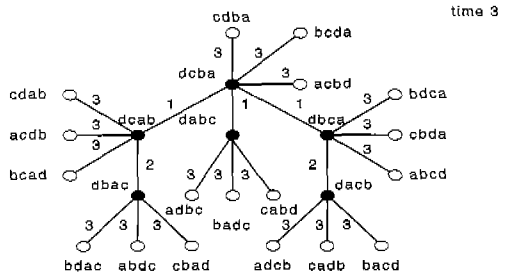

그림 7 S_4의 Broadcast_Tree

4 차원 스타에서 원시 노드 S가 *dcba*일 때, 알고리즘 1을 적용해 만들어지는 Broadcast_Tree가 그림 7에 나타내져있다. 여기서 검은색 노드들이 독립집합에 속하는 노드들이고 링크에 있는 정수 값들은 통신이 일어나는 브로드캐스팅 스텝을 이야기한다. 따라서 어느 통신 스텝에 누가 누구에게 메시지를 전해야하는 정보를 이 Tree가 갖고 있다. 그림 7의 경우 3번의 브로드캐스팅 스텝에 모든 노드들이 메시지를 전달받을 수 있는 방법을 보여준다.

다음은 제시한 알고리즘이 항상 최적+1 스텝에 브로케스팅 할 수 있는 Broadcast_Tree를 생성함에 대한 증명이다. 먼저 보조정리 1에서 마지막 스텝의 마지막 서브스타에 나타나는 target_node($\lceil \log_n n! \rceil$)구간이 1을 넘지 않음을 보이고 이를 이용해 최적+1 스텝 브로드캐스팅이 항상 존재함을 보이겠다.

보조정리 1. 알고리즘 1을 적용하여 만들어지는 마지막 브로드캐스팅 스텝에서 Send_node($\lceil \log_n n! \rceil$)의 마지막 노드의 마지막 서브스타에서의 Target_node ($\lceil \log_n n! \rceil$)구간은 1을 넘지 않는다.

증명: 먼저 보조정리 1을 위배하는 가장 작은 경우를 고려하고 그 경우도 위배상황이 발생될 수 없음을 보임으로서 모든 경우에 위의 보조정리가 만족됨을 보이겠다. 보조정리 1의 내용을 위배하는 가장 작은 경우는 마지막 target node를 결정하는 노드의 마지막 서브스타에서의 Target_node 구간은 한 개이고 그 크기가 2인 경우이다. 그러한 경우가 그림 8에 나타나 있다. 여기서 r 노드들은 send_node(T_{opt})에 속해있는 노드(마지막 스텝이 진행될 때 메시지를 갖고 있는 노드)들과 link n으로 연결된 노드들이고 y 노드들은 브로드케스트 마지막 스텝에 메시지를 갖고 있지 않은 노드들 중 아직

send_node(T_{opt})에 포함되지 않은 노드들과 link n으로 연결된 노드들이다. x 노드들과 z 노드들은 브로드케스트 마지막 스텝에 메시지를 갖고 있지 않은 노드들 중 send_node(T_{opt})에 이미 포함된 노드들과 link n으로 연결된 노드들이다. 이 때 x 노드들과 z 노드들은 존재 할 수도 있고 또 존재하지 않을 수 있다. 만약 x_1이 존재하지 않으면 $x_1 = r_i$로 생각하고 만약 x_2가 존재하지 않으면 $x_2 = r_{i+1}$이라 생각하면 된다. 마지막 $\lceil \log_n n! \rceil$ 번째의 스텝의 마지막 서브스타 SS^n_{n-1}에서 이와 같은 노드의 배열이 존재하기 위해, $\lceil \log_n n! \rceil$ 번째 스텝의 서브스타 SS^{n-1}_{n-1}에서 생길 수 있는 최소의 경우로 다음과 같은 세 경우를 고려해 볼 수 있다. 여기서 최소의 경우란 필요한 구간 크기의 최소를 의미한다. 첫 번째 경우는 구간의 크기가 2인 경우로 이때 y_1, y_2는 SS^{n-1}_{n-1}에서 각기 다른 크기가 2인 구간에 존재해야 한다. 그림 9에서, x_3, y_1, y_3, x_4와 x_5, y_2, y_4, x_6가 그런 경우를 보여주고 있다. 두 번째 경우는 y_1, y_2가 SS^{n-1}_{n-1}에서 크기가 3인 구간에 존재하는 경우이다. 그림 11(b)에서 x_3, y_1, y_3, y_2, x_4의 노드 배열이 그런 경우를 보여준다. 마지막 경우는 y_1, y_2가 SS^{n-1}_{n-1}에서 크기가 4인 구간에 존재하는 경우로 이 경우, 그림 10의 (a)-(c)와 같이 $x_3, y_1, y_2, y_3, y_4, x_4$, $x_3y_1y_4y_3y_2x_4$, 혹은 $x_3y_4y_1y_3y_2x_4$의 배열이 존재하는 경우이다. (여기서 y_1과 y_2의 순서는 상관이 없다. 앞에 오는 메시지를 받지 않은 독립 그룹의 노드와 연결된 노드를 y_1이라 하자.) 다시 말하면 그림 8과 같은 최소 위배 경우가 생기기 위한 최소한의 구간 크기는 SS^{n-1}_{n-1}에서 크기가 2인 구간이 2개 존재하던지 아니면 크기가 3인 구간 하나거나 크기가 4인 구간 하나가 존재하는 것이다. 먼저 크기가 3인 구간 하나가 존재하는 경우를 고려해보자.

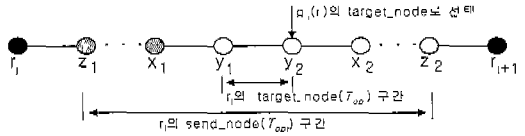

그림 8 보조정리 1의 최소 위배 경우

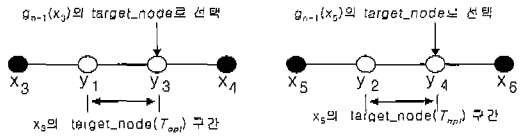

그림 9 크기가 2인 구간 2개가 필요한 경우

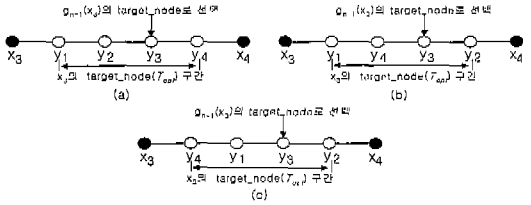

그림 10 크기가 4인 구간 하나가 필요한 경우

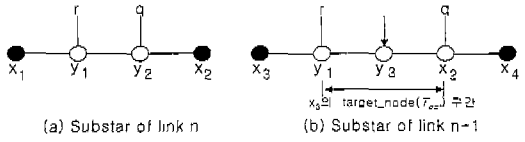

그림 11 크기가 3인 구간 하나가 필요한 경우

$\lceil \log_n n! \rceil$ 번째 스텝의 서브스타 SS^n_{n-1}에서 I_s의 노드들 중 y_1, y_2와 연결된 노드들을 각각 $r = a_1a_2a_3\ldots a_n$과 $q = b_1b_2b_3\ldots b_n$라 할 때, 경로 $r \rightarrow y_1 \rightarrow y_2 \rightarrow q$를 보면 (그림 11(a)), 노드 r에서 노드 y_1로 가는 것은 심볼 a_i $(2 \le i \le n)$ 중에서, 어떤 한 심볼이 첫 번째 위치로 나오는 것이다. 심볼 a_2가 앞으로 나온다고 하자. 그러면, 노드 y_1은 $a_2a_1a_3\ldots a_n$이 된다. 노드 y_1에서 노드 y_2로 갈 때 심볼 a_1은 앞으로 나올 수 없으므로(I_s의 노드가 아니기 때문에), 심볼 a_i $(3 \le i \le n)$ 중에서 하나가 첫 번째 위치로 나와야 한다. 심볼 a_3이 앞으로 나온다 하면, $y_2 = a_3a_1a_2\ldots a_n$이 된다. 노드 y_2에서 노드 q로 갈 때, 심볼 a_1이 제일 앞으로 나와야 하므로, 노드 q는 $a_1a_3a_2\ldots a_n$이 된다. 이 때 서브스타 SS^n_{n-1}에서 경로 $r \rightarrow y_1 \rightarrow y_2 \rightarrow q$는 노드 r에서 노드 q로 가는 최소길이경로이다. 그러면, 이제 $\lceil \log_n n! \rceil$ 번째 스텝의 SS^{n-1}_{n-1}의 경로 $r \rightarrow y_1 \rightarrow y_3 \rightarrow y_2 \rightarrow q$을 보자(그림 11(b)). 이 경로에서 노드 y_3을 거치는 것은, 위에서 선택한 임의의 심볼 2개 외에 또 다른 심볼 하나를 첫 번째 위치에 두는 것이다. 그래서, 위에서 예를 든 심볼 a_2, a_3을 제외한 a_i $(4 \le i \le n)$ 중 하나의 심볼을 임의로 선택하여 첫 번째 위치에 놓으면, 편의상 a_4라 하자, $y_3 = a_4a_1a_3a_2\ldots a_n$이 될 것이다. 이 때 $y_3 = a_4a_1a_3a_2a_5\ldots a_n$와 $y_2 = a_3a_1a_2a_4a_5\ldots a_n$노드의 최소 거리는 2임을 알 수 있다. 따라서 어느 한 노드에 이웃한 두 노드가 서로 이웃할 수 없음을 알 수 있고, 따라서, $\lceil \log_n n! \rceil$ 번째 스텝에 서브스타 SS^{n-1}_{n-1}에서 x_3, y_1, y_3, y_2, x_4의 경로 는 존재하

지 않는다. 그렇다면, 서브스타 SS^{n-1}_{n-1}에서 존재 할 수 있는 최소 경우는 그림 9와 그림 10의 4경우와 같이 구간 크기가 2인 두 개의 구간이 생기는 경우와 구간 크가가 4인 한 개의 구간이 생기는 경우가 된다. 위의 모든 경우 4개의 메시지를 받지 않은 노드들이 서브스타 SS^{n-1}_{n-1}에 존재해야 하고 마찬가지로 서브스타 SS^{n-1}_{n-1}에 4개의 메시지를 받지 않은 노드들이 존재하기 위해서는 서브스타 SS^{n-2}_{n-1}에 8개의 메시지를 받지 않은 노드들이 존재해야 한다. 이를 정리하면 마지막에 2개의 메시지를 갖지 않은 구간이 존재하려면 각각의 브로드캐스팅 스텝의 각각의 서브 스타 SS^{l}_{n-1} $(2 \le l \le n)$를 거치는 루프의 순환을 거슬러 가며 매번 2배의 메시지를 받지 않은 노드들이 존재해야하고 따라서 $\lceil \log_n n! \rceil$의 스텝마다 $n-1$개의 서브스타를 거슬러 가서 최종에는 $2^{(n-1)\lceil \log_n n! \rceil}$개의 메시지를 받지 않은 노드들이 독립 집합 I_s에 존재해야 한다. 독립집합의 크기는 $n-1$ 서브스타의 크기와 같으므로 마지막에 크기가 2가되는 구간이 존재하기 위해서는 다름과 같은 조건을 만족해야 한다.

$$2^{(n-1)\lceil \log_n n! \rceil} \le (n-1)! \quad \cdots \quad (1)$$

그러나, 식 (1)의 양변에 로그를 취하고 그 대소관계를 비교해보면, $\frac{(n-1)}{\log_2 n} \cdot \log_2 n! > \log_2 (n-1)!$가 되어 독립집합의 개수보다 더 많은 수의 메시지를 받지 않은 노드가 최초에 존재해야만 마지막에 2개의 크기를 갖는 구간이 생기게 된다는 모순이 생기고 결과적으로 이러한 경우는 불가능함을 알 수 있다. 따라서, x_1, y_1, y_2, x_2의 배열은 $T_{opt}(=\lceil \log_n n! \rceil)$ 브로드캐스팅 통신스텝 마지막 루프의 SS^{n}_{n-1}위에 존재하지 않으며 이는 모든 더 큰 구간은 더욱이 존재할 수 없다는 것을 보여준다. 따라서, 알고리즘 1을 적용하여 만들어지는 마지막 브로드캐스팅 스텝에서 Send_node($\lceil \log_n n! \rceil$)의 마지막 노드의 마지막 서브스타에서의 Target_node($\lceil \log_n n! \rceil$)구간은 1을 넘지 않는다. ■

정리 1. n 차원 스타그래프 S_n과 임의의 원시 노드 $s = a_1 a_2 \ldots a_n$이 주어 졌을 때, s로부터 다른 모든 노드로의 브로드캐스팅은 웜홀라우팅을 사용하여 최적+1 ($\lceil \log_n n! \rceil + 1$)의 통신시간 안에 완료될 수 있다.

증명: 먼저 <알고리즘1>의 각 시간에 따른 메시지 전송 때 통신에 사용되는 모든 링크에서 링크 충돌(conflict)이 일어나지 않음을 쉽게 볼 수 있다. <알고리즘1>의 모든 통신은 미리 정해진 각 서브스타에서 헤밀토니안 경로의 할당된 구간만을 사용하게끔 제한을 두고 있기 때문에 브로드캐스팅 전과정에서 어떠한 두 개의 다른 통신도 같은 링크를 사용하는 경우는 존재하지 않으며 따라서 링크의 중복 지정으로 인한 통신 지연은 발생하지 않는다.

다음은 알고리즘 자체의 통신시간이 $\lceil \log_n n! \rceil + 1$을 넘지 않음을 보이기로 하겠다. <알고리즘1>에서 $\lceil \log_n n! \rceil$ 통신시간 안에 I_s의 모든 노드들이 메시지를 전달받을 수 있다면 다른 모든 노드로의 브로드캐스팅은 $\lceil \log_n n! \rceil + 1$의 통신시간 안에 끝날 수 있음을 알고리즘 설명 때에 보았다. 그러면 I_s안의 모든 노드들이 원시 노드 s로부터 시작해서 $\lceil \log_n n! \rceil$의 통신시간 안에 브로드캐스팅 메시지를 전달받을 수 있음을 보이겠다.

n 차원 스타그래프와 원시노드 s가 주어졌을 때, I_s안의 모든 노드들이 원시 노드 s로부터 시작해서 $\lceil \log_n n! \rceil$의 통신스텝 안에 브로드캐스팅 메시지를 전달받을 수 없다고 가정해보자. 그렇게 되기 위해서는, 가장 마지막 스텝의 ($\lceil \log_r n! \rceil$) 가장 마지막에 고려되는 Send_node (T_{opt})안의 노드가 마지막 서브스타 (SS^{n}_{n-1})에서 Target_node (T_{opt})에 포함될 노드를 정할 때 그 마지막 Send_node (T_{opt})안의 노드가 고려하는 구간의 크기가 2 이상 이여야 한다. 구간의 크기가 2 이상이라면 그 구간에 있는 노드들에 연결된 독립집합의 노드들은 아직 메시지를 받지 않았다는 소리이고 구간에 있는 노드 중 오직 한 노드만이 마지막 최적 스텝에 메시지를 받을 수 있으므로 나머지는 $\lceil \log_n n! \rceil$스텝 안에 메시지를 받을 수 없게 된다. 그러한 마지막에 나타나는 구간 중 가장 작은 경우인 구간의 크기가 2인 경우도 생길 수 없다는 것을 보조정리 1에서 증명했다. 결국, 마지막 스텝의 마지막 스타에서 생기는 구간은 하나보다 작거나 같고 구간이 있다면 그 크기가 1임을 알 수 있다. 따라서, I_s의 모든 노드들은 $\lceil \log_n n! \rceil$ 브로드캐스팅 스텝시간 안에 메시지를 받을 수 있고 전체 브로드캐스팅에 걸리는 스텝 수는 $\lceil \log_n n! \rceil + 1$ 을 넘지 않게 된다. ■

4. 결 론

본 논문에서는 전포트 사용과 웜홀라우팅을 지원하는 스타 그래프 네트워크 시스템에서 효율적인 브로드캐스팅 알고리즘을 제시하였다. 스타 그래프는 짧은지름과 적은 차수로 기존에 널리 사용되고 있던 하이퍼큐브에 비교되며 많은 연구가들의 관심의 대상이 되어왔다. 그 중에서도 브로드캐스팅은 네크워크 구성의 기초가 되는

중요한 문제로써 고려되어야 할 근본적인 통신 문제이다. 그런데 기존에 알려진 스타그래프에서의 브로드캐스팅 알고리즘[1]이 이론적인 경계(lower bound), 즉, 최적 통신시간에 비해 많은 차이를 갖고 있었다. 본 논문에서는 이러한 통신시간을 줄이고자 하는 것을 목적으로, 해밀토니안 경로를 이용하여 링크-충돌 문제를 해결하여 낭비되는 통신 링크를 최소화시킴으로 해서, 모든 노드가 메시지를 전달받는 데에 최적에 가까운 최적+1 ($\lceil \log_n n! \rceil + 1$)의 브로드캐스팅 스텝이 걸리는 알고리즘을 제안하였다. 이는 기존의 브로드캐스팅 알고리즘의 통신시간인 n-1 브로드캐스팅 스텝에 비해 향상된 결과이다.

참 고 문 헌

[1] S. B. Akers, D. Harel, and B. Krishnamurty, "The Star graph: An Attractive Alternate to the n-cube", *Proc, Int'l Conf. on Parallel Processing*, pp. 393-400, 1987.

[2] S. Latifi and N. Bagherzadeh, "The Clustered_Star Graph: A New Topoloty for Large Interconnection Networks", *Proceedings of the 7th International Parallel Processing Symposium*, pp. 514-518, *IEEE Computer Society Press*, April 1993.

[3] J-Y. Lee(Park) and H.-A. Choi, "Circuit-Switched Broadcasting in Torus and Mesh Networks", *IEEE Trans. on Parallel & Distributed Systems*, vol. 7, No. 2, pp. 184-190, Feb. 1996.

[4] S.-K. Lee and J.-Y. Lee, "Optimal Broadcast in α-port Wormhole-Routed Mesh Networks", *1997 International Conference on Parallel and Distributed Systems*, pp. 109-114, 1997.

[5] V. Mendia and D. Sarkar, "Optimal Broadcasting on the Star Graph", *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 4, pp. 389-396, July 1992.

[6] L.M. Ni and P.K. Makinley, "A Survey of Routing Techniques in Wormhole Networks," Technical Report MSU-CPS-ACS-46, October 17, 1991.

[7] S. Latifi and P.K. Srimani, "Wormhole Broadcast in Star Graph Networks," Parallel Computing, vol. 24, no. 8, pp. 1263-1276. August 1998.

[8] M. Nigam, S. Sahni, and B. Krishnamurty, "Embedding Hamiltonians and Hypercubes in Star Interconnection Graphs", *Proc. of 1990 Int'l Conf. Parallel Processing*, pp. 340-343, 1990.

김 차 영

1998년 숙명여자대학교 전산학 석사. 1996년 숙명여자대학교 전산학 학사. 관심분야는 알고리즘, 병렬처리, 그래프이론

이 상 규

1989년 University of Southern Califonia Dept. of Computer Science (공학사). 1991년 George Washington University Dept. of Electrical Engineering and Computer Science(공학석사). 1995년 George Washington University Dept. of Electrical Engineering and Computer Science(공학박사). 1995년 ~ 1996년 George Washington University Dept. of Electrical Engineering and Computer Science 박사후 과정. 1997년 ~ 현재 숙명여자대학교 전산학과 교수. 관심분야는 병렬/분산 처리 시스템, 컴퓨터이론, 컴퓨터 통신, 인터넷 프로그래밍.

이 주 영

1984년 이화여자대학교 수학과 졸업(학사). 1991년 The George Washington Univ. 전산학과 졸업(석사). 1996년 The George Washington Univ. 전산학과 졸업(박사). 1996년 ~ 현재 덕성여자대학교 전산학과 조교수. 관심분야는 알고리즘, 병렬처리, 그래프이론