논문 01-26-7A-14 한국통신학회논문지 '01-7 Vol.26 No.7A

혼합 웜홀 컷-스루 라우팅 하에서 비균형 트래픽시 계층적 버퍼를 사용한 양방향 MIN의 성능평가

정회원 장 창 수*

Performance Evaluation of Bidirectional MIN Using Hierarchical Buffering Scheme on Non-Uniform Traffic Under HWCR

Chang Soo Jang* *Regular Member*

요 약

본 논문에서는 확장 가능한 병렬컴퓨터 시스템의 성능에 중요한 영향을 주는 MIN에서 통신 지연을 개선하기 위한 방법에 대해 고찰하였다. 네트워크의 트래픽이 매우 많거나 고르지 않게 분포 되었을 때 개선하기 위한 효율적인 스위칭 방안으로, 본 논문에서 제안된 HWCR 스위칭 방법은 양방향 MIN상에서 웜홀 라우팅을 수행시 라우팅 경로 상에 패킷들이 충돌이 발생하여 링크가 블럭되었을때 VCT로 전환하는 방법이다. 각 스위치에 계층적 버퍼를 설치하므로서 더욱 효율적인 라우팅을 할 수 있도록 설계하였다. 이는 VCT 라우팅이 일률적으로 각 스테이지마다 커다란 크기의 버퍼를 부여함에 반하여 매우 Cost-Effective 한 버퍼 할당 방법이다. 이를 정당화하고 버퍼 크기와 라우팅시 평균 통신 지연과의 관계를 적정화하기 위해 균일 트래픽 패턴과 여러 가지 비균일 트래픽 패턴 하에서 성능평가를 하였다. 성능결과는 부하가 0.4 일때 제안된 HWCR 방법이 VCT라우팅 보다 2-3배의 버퍼가 절약됨을 보여준다.

ABSTRACT

In this dissertation, we have proposed a new routing method Hybrid Wormhole and Virtual-Cut through Routing(HWCR), to improve a performance under wormhole routing. If confliction happens under wormhole routing at hot traffic, a performance of network decreases rapidly. When a blocking occurs, HWCR method transfers wormhole routing to Virtual-Cut through(VCT) so that it can provide a flow of flits freely. And if blocked link is removed, routing will return to wormhole routing. This scheme is a hybrid routing which is both wormhole and VCT routing that is presented for a heuristic method for multicasting under unicast free contention. Buffer in HWCR method adopted an hierarchical buffering scheme for improving the network performance and reducing the buffer cost in BMINs. This is a cost-effective shared buffering scheme. For optimum buffer size and communication latency, we compared and analyzed results based on the computer simulation at wormhole routing, VCT and proposed HWCR under uniform traffic pattern under various nonuniform traffic pattern. The performance results show that the proposed HWCR shceme is a cost-effective more two or three times than VCT routing in terms of buffer size when offered load is about 0.4.

Ⅰ. 서 론

현재 대다수의 중·대형 컴퓨터 시스템이 대칭형의 다중프로세서(Symmetric MultiProcessor : SAP) 구조를 기반으로 설계되어 CPU를 2개에서 많게는 10개까지 탑재하도록 되어 있다[8]. 그러나 이런 다

* 여수대학교 컴퓨터공학과 교수(csjang@ce.yosu.ac.kr)
논문번호 : K01015-0108, 접수일자 : 2001년 1월 8일
※ 본 연구는 여수대학교 1999년도 학술연구지원비를 지원 받아 수행되었습니다.

중프로세서 기종은 프로세서와 메모리가 단일 버스 구조 하에서 운용되기 때문에 전체 메모리를 쉽게 관리할 수 있는 이점은 있으나 지속적인 성능향상과 메모리 확장에는 한계가 있다. 이에 반해서 병렬처리 컴퓨터는 이러한 다중프로세서 기종의 문제를 보완해 개발, 모든 CPU와 메모리를 상호 병렬로 연결할 수 있도록 설계하여 지속적인 성능향상은 물론 거의 무한대로 메모리를 늘여나갈 수 있다. 이러한 병렬처리 시스템 내에는 다수의 프로세서와 다수의 모듈화 된 메모리들이 밀 결합 또는 약 결합 형태로 상호 연결되어 있다. 이들을 서로 연결해 주는 상호연결망은 시스템의 성능에 상당한 영향을 미치는 핵심구성 성분이다[2]. 즉, 상호연결의 대역폭에 따라 시스템의 확장성 및 대용량의 데이타를 처리하는데 있어서의 응답지연이 영향을 받게 된다. 일반적으로 시스템 내에서의 상호연결망은 연결방법에 따라 크게 다음과 같이 나눌 수 있다. 공유 버스 시스템(Shared-Bus System), 크로스바 스위치(Crossbar Switch), 단단계 상호연결망, 다단계 상호연결망(MIN)등으로 구분된다. MIN 구조의 상호연결망은 크게 2가지의 장단점이 존재하는데, 장점으로는 모듈러한 구성이 갖는 확장성의 용이성이 있고, 단점으로는 망 지연시간이 네트워크에서 단계수에 비례하여 증가한다는 문제가 존재한다. 대규모의 MIN구성이나 크로스바 망이 범용 컴퓨터 시스템에서 동적인 위상을 구성하는데 더 경제적이고 용이하게 될 것이다. 상호연결망 기술 추세는 다수개의 프로세서를 망 병목현상 없이 데이타 통신이 가능하고 망 대역폭을 확장하려는 방향으로 진행되고 있다. 특히 대형 병렬컴퓨터 시스템에서 상호연결망 구조는 전체 시스템의 성능에 중대한 영향을 미치는 요소이다. 특히 많은 수의 프로세서를 어떤 방식으로 어떻게 상호연결 할 것인가는 병렬처리 컴퓨터 분야에서 지속적으로 연구되어야할 분야이다.

MIN은 확장가능한 병렬 컴퓨터 시스템를 구성하기위해 스위치에 기반을 둔 네트워크 구조의 일반적인 종류이다. 이런 MIN에서의 통신 지연시간은 병렬 컴퓨터의 성능에 영향을 주는 중요한 요소이다. 통신 지연이란 패킷이 소스에서 목적지까지 경과된 시간을 말한다. 이런 지연은 첫째 소스에서 큐잉지연, 둘째 링크(혹은 버퍼) 충돌 셋째 출력에서 다중패킷 방출에 기인한 출력 충돌과 같은 세 가지 충돌 형태에 기인한다[3].

만약 네트워크의 트래픽이 매우 많거나, 고르지 않게 분포되면 통신지연은 높아진다. 이러한 통신지연은 스위칭 기술에 매우 많이 좌우된다[10]. 병렬컴퓨터 스위칭 방법으로는 회선스위칭(circuit switching), 패킷 스위칭이나 store-and-forward 스위칭, 웜홀 스위칭[3]과 같은 방법들이 있다. 낮은 통신 지연 시간과 작은 버퍼 공간을 제공하기 위해서 요즘 대부분의 병렬 처리 시스템에서 웜홀 스위칭 기법을 사용하고 있으며, 여기서도 오직 웜홀 스위칭 기법만을 고려하기로 한다. 스위치 기반 네트워크의 스위치는 소규모 크로스바 스위치이다. <그림 1>에서 볼 수 있듯이, $k \times k$ 스위치는 한 쪽에 k개의 입력 부와 다른 쪽에 k개의 출력부를 가진다. 그림 1의 (a)는 k개의 입력 부와 k개의 출력부를 가진 단방향 TMIN(Traditional MIN)을 보여 주는데, 각 입/출력부는 단방향 통신 채널을 가진다. (b)는 d-dilated ($d=2$) 단방향 DMIN(Dilated MIN) 을 보여주는 데, 각 입/출력부는 d 개의 단방향 채널을 가진다. 라우팅 알고리즘에 의해서, 나가는 패킷은 선택된 출력부에 있는 d 개의 채널들 중에 임의의 한 채널을 선택할 수 있다. 그래서 d개의 채널 때문에 d개의 서로 다른 패킷들이 하나의 출력부로 블록킹을 발생하지 않고 동시에 전송될 수 있다. (c)는 여러 개의 패킷들이 하나의 출력부를 공유하는 또 다른 방식인 가상 채널을 사용한 단방향 VMIN(Virtual MIN)을 보여 준다. 두 개의 가상 채널이 하나의 물리적 채널을 time multiplexing 방식으로 공유하고 있는 것을 나타낸다. 이상의 세 개의 단방향 스위치들과는 다르게, (d)는 각 입/출력부에서 각각 다른 방향의 쌍으로 되어 있는 양방향 BMIN을 나타낸다. 이것은 두 개의 패킷들이 서로 이웃한 스위치들 사이에서 동시에 서로 다른 방향으로 전송되는 것을 가능하게 해 준다.

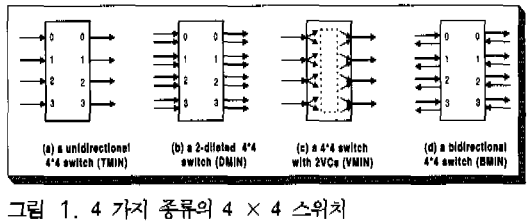

그림 1. 4 가지 종류의 4 × 4 스위치

양방향 스위치는 <그림 2>에서 볼 수 있듯이, 세 가지 종류의 연결을 제공한다. 순방향, 역방향, 선회 연결을 볼 수 있다. 이 때 주의할 점은 역방향으로부터 오는 선회 연결은 허락되지 않는다. 이 성질은 최단 경로 라우팅이 싸이클을 형성함으로써 데드록이 발생하는 것을 방지한다.

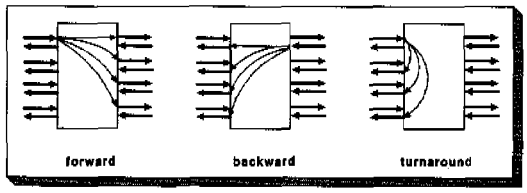

그림 2. 4 × 4 양방향 스위치에서 모든 가능한 연결 유형

병렬처리 시스템의 프로세서 모듈 구성의 특성으로 인해서 양방향 링크를 갖는 BMIN은 경로상의 다중패스가 기존의 TMIN보다 더 많은 수가 존재하므로, 채널(또는 link) 충돌에 의한 블록킹 확률은 상대적으로 적어진다. 따라서 BMIN에는 웜홀 라우팅의 라우팅 시간이 거리에 무관한 특성을 이용하여 네트워크 지연을 줄이려는 시도가 있었다[13].

따라서 본 연구에서는 이러한 문제를 상호 보완하여, 트랙픽이 많은 웜홀 스위칭 하에서 블럭킹이 발생하였을 때, 성능이 급격히 저하되는 것을 방지하면서 적당한 비용 대 성능을 얻을 수 있는 새로운 라우팅 기법을 제안한다. 이 방법을 Worm-hole and Virtual-Cut-through(Hybrid WC)라우팅이라 부른다. 이 라우팅의 성능평가를 하기 위해 여러 가지 비균형 트래픽 하에서 기존의 라우팅과 시뮬레이션을 할 것이다[7]. 제안한 라우팅 방법은 트래픽의 증가로 인해 많은 채널이 점유되어 링크에서 블럭킹이 발생한 상황에서 VCT 라우팅 기법의 장점을 혼합한 라우팅 방법이다. 그리고, 제안한 Hybrid WC 라우팅을 위한 BMIN의 버퍼는 다단계 상호연결망에서 각 레벨마다 링크의 사용 빈도 수가 다르다는 것에 착안하여 각 SE에 VCT 경우보다는 상대적으로 적은 양의 계층적 버퍼을 설계하여 사용한다. 이로써 기존의 웜홀 라우팅보다 더 우수한 성능을 성취할 수 있을 것이다. 이 기법으로 링크 자원을 더 효율적으로 사용하여 MPP 시스템의 네트워크 성능을 향상시킬 수 있을 것이다. 마지막으로 성능을 증명하기 위해 균일 트래픽 패턴과 비균일 트래픽 패턴 하에서 각 네트워크 입력에 들어온 패킷들이 각 스테이지를 거치면서 라우팅 하는 과정에서 각 라우팅 모델과 여러 트래픽 패턴을 가지고 중요한 factor가 되는 지연시간 및 사용한 버퍼 량을 분석, 시뮬레이션한 결과를 기반으로 결론을 내린다.

BMIN에서 사용되는 웜홀 라우팅은 네트워크 트래픽이 적은 경우에는 효율적이지만, 트래픽이 비균일한 패턴이거나, 과도한 트래픽 집중현상이 일어날 경우에는 상대적으로 많은 채널이 점유되어, 전체적으로 네트워크 처리율이 급격히 저하되는 단점을 갖는다[5,1,3]. 또한 일반적으로 VCT 라우팅 방법은 많은 양의 버퍼를 필요로 하므로 스위칭 소자 설계에 많은 비용이드는 단점이 있으나, 많은 트래픽 시에는 블럭킹이 발생하지 않으므로 웜홀 보다 우수한 성능을 보여준다[3].

본문은 다음과 같이 구성된다. Ⅱ장에서는 MIN에 대한 전반적인 구조와 특성을 정의 하며, BMIN의 Fat tree 위상에 따른 선회(TurnAround) 알고리즘을 기술한다. Ⅲ장에서는 Cut-Through 방식에서 웜홀과 VCT방식을 비교 연구하여 본 연구에서 제안한 새로운 라우팅 알고리즘을 설계하고, 기존의 라우팅 알고리즘의 장점을 수용하여 Dynamic한 BMIN에서 계층적 버퍼설계와 *SE* 구조를 설계 제안한다. Ⅳ장은 성능평가를 위해 균일트래픽 패턴과 여러 가지 비균일 트래픽 패턴에 대해 제안하고, Ⅴ이들 트래픽패턴 하에서 평균 패킷 지연과 버퍼 크기를 성능 평가 척도로 시뮬레이션하여 평가한다. Ⅵ장에서는 마지막으로 결론을 제시한다.

Ⅱ. 양방향 MIN과 스위칭 기법

1. BMIN 구조

양방향 통신을 허용하기 위해서 스위치의 각 포트는 이중 채널을 갖는다. 즉, 각 입/출력부에서 두 개의 채널을 가져야만 한다. 8-노드 버터플라이 양방향 MIN에서 프로세서 노드들이 네트워크의 왼쪽 편에 있는 시스템 구조가 <그림 3>에서 보여준다. 네트워크의 연결 구성은 최상위 레벨에서 180도 선회하여 접속하도록 한다. <그림 3>에 8-노드 버터플라이 BMIN이 나타나 있는데, G2는 없어도 선회 연결이 되지만, TMIN과 DMIN의 경우와 위상적으로 동일하게 하기 위해서 첨가하기로 한다. 웜홀 라우팅 기법과 선회 연결을 포함하는 BMIN을 사용하는 상업적인 SPC들이 많이 있다. 예를 들면, TMC CM-5, Meiko CS-2, 그리고 IBM SP-1/2 등이 있는데, CM-5에서는 처음 두 스테이지에서는 이중 입력 통신이 가능한 4×2 스위치들을 사용한다. 다음은 BMIN의 위상에 연관된 정의이다[13][6]. 다음 정의에 따라 BMIN을 구성할 수 있으며, 이러한 정의는 라우팅 시에도 사용된다. <그림 3> 은 이러한 정의를 사용하여 만든 8×8 BMIN이다.

【정의 1】 BMIN는 $N \times N$ 스위치로 이루어지며, 한 스위칭 소자 *SE*의 크기는 $k \times k$로 나타낸다. 노드의 수는 $N = k^n$ 이고, 한 스테이지의 수 (n)는 $\log_k N$개이며, 한 스테이지의 *SE* 수는 N/k 개이다.

【정의 2】 하나의 $k\times k$ SE_i는 양방향 링크를 가지며, 왼쪽에서 오른쪽 방향으로 경로 연결시(즉, 순방향 연결 시) 입력 노드는 위에서부터 차례로 $l_0, l_1, \ldots, l_{k-1}$로, 출력 노드는 $r_0, r_1, \ldots, r_{k-1}$로 인덱스 된다. 오른쪽에서 왼쪽으로 연결시(즉, 역방향 연결 시) 입력 노드는 위에서부터 차례로 $l_0, l_1, \ldots, l_{k-1}$로, 출력 노드는 $r_0, r_1, \ldots, r_{k-1}$로 인덱스 된다.

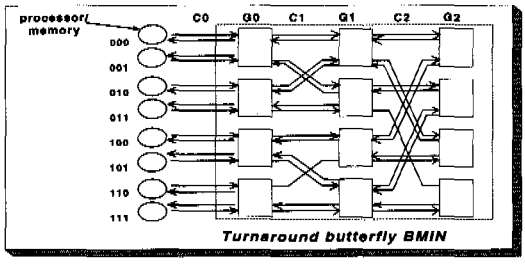

그림 3. 8-노드 양방향 butterfly BMIN

【정의 3】 임의의 스테이지는 S_i로 정의하는데, i는 스테이지 번호로 $0\le i\le n-1$이며, 여기서 $n=\log_k N$ 이다. 한 스테이지내의 SE들은 위에서부터 $SE_{i,0}, SE_{i,1}, \ldots, SE_{i,N/k-1}$ 로 인덱스 된다.

【정의 4】 임의의 채널(또는 링크) 스테이지는 C_i로 정의하는데, i는 채널 번호로 $0\le i\le n-1$ 이다. 여기서 $n=\log_k N$ 이다.

【정의 5】 소스 주소는 $S=s_{n-1}\ldots s_1 s_0$로 나타내고, 목적지 주소는 $D=d_{n-1}\ldots d_1 d_0$로 나타낸다. 여기서 $n=\log_2 N$ 이다.

BMIN과 같이 선회 지점을 갖는 네트워크 구조에서 *FirstDifference*(S,D)의미는 매우 중요하다. 이 값은 라우팅 시 선회 지점을 결정하는 값이 된다. $K\times K$스위치를 가진 순회 버터플라이 BMIN에서 근원지 주소가 $S=s_{n-1}\ldots s_1 s_0$이고, 목적지 주소가 $D=d_{n-1}\ldots d_1 d_0$ 일 때, *FirstDiff-erence*(S,D)는 $s_{n-1}\ldots s_1 s_0$와 $d_{n-1}\ldots d_1 d_0$ 사이에 처음으로 다르게 나타나는 최상위 비트를 나타내는 함수이다. 예를 들어, 이 값이 2일 경우는 라우팅 시에 스테이지 S_2에서 선회하며, 1인 경우에는 S1 스테이지에서 선회하게 된다. 따라서 *FirstDifference*$(S,D)=i$ $(i<n-1)$인 경우는 전체 네트워크를 순회할 필요 없이 Si 스테이지에서 선회하면 된다. 즉, 최상위 레벨까지 올라갔다가 다시 내려올 필요가 없다. 다음은 채널 충돌이 없는 일반적인 경우에서의 라우팅 알고리즘이다. [알고리즘 1]은 충돌이 없는 경우로 다음절에서 충돌을 해결하는 알고리즘으로 확장될 것이다. 이 알고리즘은 먼저 *FirstDifference*(S,D)을 구한 다음, 순방향 경로에서는 임의의 자유로운 링크($r_i(0\le i\le k-1)$)를 선택하여 랜덤 하게 라우팅 한다. 선회 지점은 *FirstDifference*(S,D) 값을 갖는 스테이지번호, *FirstDifference*(S,D) 값이 t 일 경우 St 스테이지가 선회하는 지점이 된다.

Algorithm1 : Turnaround routing in each switch at stage j
Input: Source address S : s_{n-1} ..... $s_1 s_0$
Destination address D : d_{n-1} ..... $d_1 d_0$
Procedure:
1. t : = *FirstDifference*(S, D) ($j\le t$ is always true)
2. If $j=t$, then take a turnaround connection to port l_{di}.
3. If $j<t$ and the message comes from an input port l_m, then take a forward connection to any available port r_i $(0\le m,\ i\le k-1)$.
4. If $j>t$ and the message comes from an input port r_m, then take a backward connection to port l_{di} $(0\le m\le k-1)$.

알고리즘 1] 선회 버터플라이 라우팅 알고리즘

이 지점 이후의 라우팅은 역방향 경로를 거치게 되는데, 이 경로 상에서는 일반적인 MIN에서의 라우팅 기법인 목적지 태그를 이용한 분산 라우팅을 수행하게 된다. 다시 말하면, 라우팅 할 출력포트가 목적지 주소 값, $D=d_{n-1}\ldots d_1 d_0$ 에 의해 결정된다. 2×2 SE인 경우 $d_i(0\le i\le n-1)$값이 0인 경우는 l_0 출력포트로 라우팅하며, 1인 경우는 l_1 출력포트로 라우팅 한다(이 경우 l_0 가 상위 출력 링크, 그리고 l_1이 하위 출력 링크가 된다). 스위치로 설계된 버터플라이 BMIN에서는, 소스 어드레스 S와 목적지 어드레스 D는 각각 $k-ary$ $k\times k$ 수들 $s_{n-1}\ldots s_1 s_0$ 와 $d_{n-1}\ldots d_1 d_0$에 의해 표현되어진다. *FirstDifference*(S,D) 함수는 t 에서 선회한다. 이 값은 $s_{n-1}\ldots s_1 s_0$ 와 $d_{n-1}\ldots d_1 d_0$ 사이에 왼쪽에서 첫 번 째 다른 디지트가 나타나는 곳의 위치이다. 다음 【정의 6】에서 정의한다.

【정의6】 $t<j<n$ 에서 $s_t\neq d_t$ 이고 $s_j=d_j$가 되는 t 값을 *FirstDifference* $(S,D)=t$ 라고 한다. 이 t

값은 소스와 목적지 주소 사이에 왼쪽 첫 번째 다른 디지트 값을 갖는 비트 위치를 말한다[12][14]. *FirstDifference* $(S,D)=t$, 이 값은 소스에서 목적지까지 라우팅 시에 선회하는 스테이지를 결정하는 값이 된다. 예를 들어, 이 값이 2 일 경우(예 : $011(s_2s_1s_0) \rightarrow 111(d_2d_1d_0)$)는 라우팅 시에 스테이지 S_2에서 선회하며, 1 인 경우(예 : 001→010)에는 S_1 스테이지에서 선회하게 된다.

【정의 7】 채널 $c_i^{r,j}$는 순방향연결에서, 출력포트(또는 출력 링크)가 $r_j(0\le j\le k-1)$인 i $(0\le i\le \log_k N-1)$번째 채널 스테이지에 있는 채널을 말한다. 그리고, 채널 $c_i^{l,j}$는 역방향 연결에서, 출력포트가 l_j인 i번째 링크 스테이지에 있는 채널을 말한다. 소스와 목적지 사이에 선회 라우팅 경로는 아래 【정의 8】 에서 정의한다.

【정의 8】 선회경로는 소스노드에서 목적지노드까지 라우트된다. 경로는 다음 조건을 만족해야한다.

- ◆ 경로는 전 방향 채널들과 역방향 채널들 그리고 하나의 선회 연결자 로 구성된다.
- ◆ 순방향 채널의 수는 역방향 채널의 수와 같다.
- ◆ 경로 상의 어떤 순방향 채널들과 역방향 채널들 쌍도 같은 입/출력 부의 채널 쌍이 될 수 없다.

마지막 조건은 중복하여 통신이 발생하는 것을 막아준다. 소스에서 목적지까지 메세지를 보낼 때, 처음에 메세지는 스테이지 G_t로 순방향으로 보내고, 이 때 스테이지 G_t로 순방향으로 보내질 때 어떤 출력 채널을 선택해야 할 지는, 스테이지 G_t에서 전 방향으로 움직임에 따라, 메세지는 취할 수 있는 전 방향 출력채널로서 다중선택이 있을 수 있다. 그 결정은 다른 메시지에 의해서 사용되지 않은, 블럭킹 되지 않는 임의의 출력 채널을 선택할 수 있다. 일단 스테이지 G_t의 스위치에 도착하게 되면, 역방향으로 그 목적지까지 유일경로를 선택하게 된다. 그래서 역방향 라우팅은 목적지 태그 라우팅 방법으로 결정된다. 다음은 선회 버터플라이 라우팅 알고리즘을 나타내었다.

FirstDifference(000, 101)=2이고 경로 A→B→C는 임의로 선택된다. 또한 대체 경로로서 A→E→F가 선택될 수도 있다. 메시지가 일단 C까지 순방향으로 전송된 다음에는 C에서 선회하여 역방향으로 D로 전송된다. 이 때, C→G→H→D 로 경로는 유일한 경로가 된다. 이 경로는 목적지 주소 $d_2d_1d_0=101$ 에 의해서 C의 l_1, G의 l_0, H의 l_1의 출력 채널이 각각 선택되어 진다. 그리고 $S1\rightarrow D1$의 라우팅과 $S2\rightarrow D2$의 라우팅에서 블럭킹 예를 나타낸다. <그림 4>에는 2×2 스위치를 가진 8-포트 선회 버터플라이 BMIN의 라우팅 예와 블럭킹 예를 나타내었다. 일반적으로 8×8 BMIN에서 *FirstDifference*(*S*,*D*)가 2인 경우는 4개의 중복경로가 존재한다. *FirstDifference*$(S,D)=1$ 인 경우는 2개의 중복 경로가 가능하다.

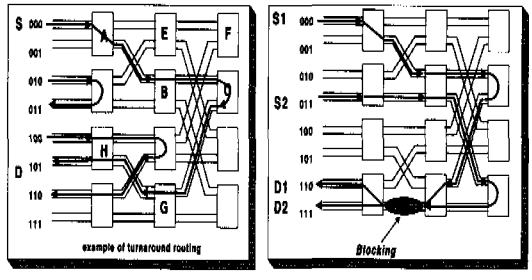

그림 4. 선회(Turnaround) 라우팅과 블럭킹의 예

따라서 $N\times N$BMIN 네트워크에서 가장 최단 거리의 중복 경로의 수(*RD path*)는 다음과 같이 결정할 수 있다[15].

$$RD\ \ path=k^t$$
$$(t=FirstDifference(S,D),0\le t\le \log_k N-1)$$

다음 〈그림 5〉에서 16−*node* 선회 라우팅 시 중복 경로를 나타냈다.

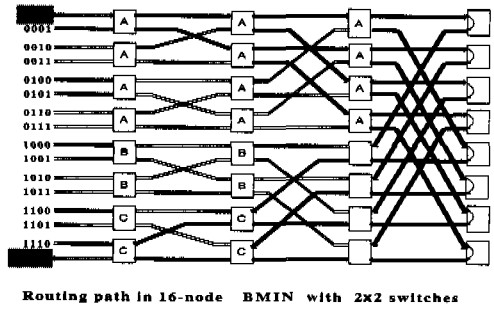

그림 5. 2×2 스위치에서 16-노드 BMIN의 라우팅경로

기존의 BMIN에서 중복경로가 존재할 때, 원홀 라우팅을 사용할 경우, 하나의 경로에 우선 순위를 두어, 이 경로를 먼저 선택한 후 전송을 하고, 이 경로상의 전송이 종료된 후에 블럭된 나머지 경로도 전송을 재개하거나, 아니면 블럭된 경로를 연결하여 다시 재 전송하는 방법을 택할 수 있다. 그러나 재 전송하는 경우 블럭된 시점까지 전송된 패킷

(또는 플릿)들을 모두 폐기해야하는 문제가 있고, 우선순위가 높은 경로가 먼저 전송을 완료할 때까지 대기하는 방안은 현재 블럭된 지점이전까지 경로상을 점유하고 있는 플릿들로 인해 다른 경로의 진행이 또 다시 블럭되는 문제가 있다. 이런 문제는 블럭된 링크가 풀릴 때까지 경로 상에 있는 링크를 계속 점유상태로 존재하기 때문에 심각한 자원(link) 부족 문제 야기하여 네트워크의 성능을 급격히 저하 시킨다. 현재 블럭된 지점 이전까지 경로 상을 점유하고 있는 플릿들로 인해 또 다른 경로의 진행을 방해하기 때문에 다시 블럭되는 문제가 발생된다. 이런 상황은 비균일한 트래픽 패턴일 경우는 더욱 심각하다. 다음 <그림 6>은 중복 경로 때 충돌이 존재하는 경우의 라우팅 예이다.

$S_1=001\rightarrow D_1=110$, $S_2=011\rightarrow D_2=111$로의 2개의 경로가 동시에 존재하는 라우팅이다. <그림 6>에서 $S_2\rightarrow D_2$ 경로에 우선 순위를 두면, $S_1\rightarrow D_1$ 경로 상에 있는 SE들은 블럭된 링크 C_1^0가 풀릴 때까지 계속 점유상태가 된다.

중복경로가 존재할 때, 기존의 BMIN에서 웜홀 라우팅을 사용할 경우, 하나의 경로를 먼저 선택하여 전송을 한 후 블럭된 나머지 경로도 재전송을 해야 하지만 그러나 블럭된 시점까지 전송된 패킷(또는 플릿)들을 모두 폐기해야 하는 문제가 발생한다. 현재 블럭된 지점 이전까지 경로 상을 점유하고 있는 플릿들로 인해 또 다른 경로의 진행을 방해하기 때문에 다시 블럭되는 문제가 발생된다. 이런 상황은 비균일한 트래픽 패턴일 경우는 더욱 심각하다. <그림 6>에서 $S_2\rightarrow D_2$ 경로에 우선 순위를 두면, $S_1\rightarrow D_1$ 경로 상에 있는 SE들은 블럭된 링크 C_1^0가 풀릴 때까지 계속 점유된다. BMIN에서 두 개의 라우팅 경로 P와 Q가 존재한다고 가정할 때, 이는

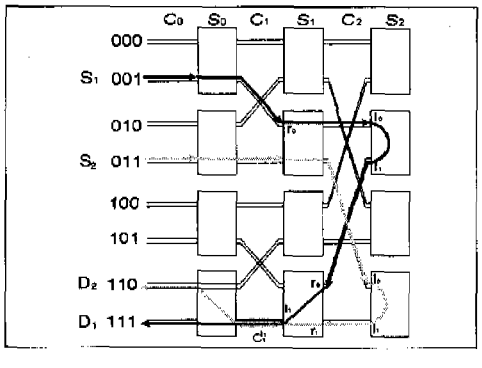

그림 6. 8 × 8 BMIN에서 블록킹

순방향 라우팅 시에는 여러 개의 대체 경로로 라우팅이 가능하기 때문에 충돌을 피할 수가 있다. 충돌은 선회 지점까지는 피할 수 있지만, 역방향 라우팅이 시작되는 이후부터 단 하나의 경로만이 존재하기 때문에 충돌 가능성이 항상 존재한다. 충돌이 발생하는 경우를 다음과 같이 정의할 수 있다.

【정의 9】 두 경로 P와 Q가 스테이지 S_i $(0\le i\le n-2)$의 한 SE_j $(0\le j\le N/2-1)$에서 동일한 출력 링크 상에서 충돌이 발생할 경우는 다음의 경우이다.

P와 Q의 목적지 주소를 각 각 $d_{n-1}\ldots d_1d_0$ 라고 가정한다. P와 Q 가 S_i의 SE 내에 한 입력 링크로 각각 r_m $(0\le m\le k-1)$과 r_n $(n\neq m, 0\le n\le k-1)$를 사용하고, P의 출력 링크 l_{d_i} $(0\le i\le n-2)$와 Q의 출력 링크 $l_{d_i'}$가 $d_i=d_i'$ 인 관계가 성립할 경우에 출력 링크에서 충돌이 발생한다.

2. 웜홀에서 블럭킹 vs. VCT에서 블럭킹.

스윗칭은 입력 채널로 들어오는 데이터를 출력 채널로 내보내는 실제적인 메커니즘이며, 통신 지연은 사용되는 스윗칭 기법에 따라서 좌우된다[5][18]. 병렬 컴퓨터를 위한 직접 연결망에서 사용되는 메시지 통과하는 방법으로는 크게 store-and-forward, circuit 스위칭, VCT, 웜홀 라우팅이 있다. 직접 연결망의 성능을 평가하기 위한 척도로 쓰이는 통신 지연은 출발 지연, 네트워크 지연, 블럭킹 시간 등을 합한 시간으로 계산한다.

store-and-forward 방식의 문제점으로 지적된 프로세서들간의 통신 속도가 느린 점을 극복하기 위해서 프로세서들간의 통신 속도가 빠른 cut-through 스위칭 방법이 개발되었다. 이 방법에는 프로세서의 간섭 없이 흐름을 조절하기 위한 라우터가 각 노드마다 필요하다. 만약 다음 노드로써 채널이 사용가능하면, 중간 노드에 도착한 메세지는 경로 상에 있는 다음 노드에 아무것도 저장하지 않고 즉시 진행한다. 이 때, cut-through는 플릿 버퍼에 있는 메세지의 헤더 플릿에 대해서만 행해진다. 만일 cut-through가 모든 중간 노드에 대해서 행해진다면 circuit이 형성되고 circuit 스위칭과 유사하게 된다. 그리고, 메세지 헤더가 블럭킹 될때, 어떻게 처리하느냐에 따라서 Cut-through 방법이 가상 cut-through와 웜홀 라우팅 두 가지로 나뉜다. 〈그림 7〉 에서 볼 수 있듯이, 웜홀 라우팅에서 패킷 1 의 헤더 플릿이 다음 노드로 진행하려고 할 때, 이미 다른 패

킷 2가 그 채널을 점유하고 있다면 블럭킹이 발생한다. 이 때, 웜홀 라우팅에서는 헤더 플릿(H)이 진행하려는 채널이 사용가능해 질 때까지 기다리는데, 그 헤더 플릿(H)의 뒤를 따르던 모든 데이터 플릿(D1,D2)들은 통신상에서 채널을 점유한 채로 블록킹 된다. 반면에, 〈그림 8〉에서 볼 수 있듯이, virtual cut-through 에서 패킷 1의 헤더 플릿이 다음 노드로 진행하려고 할 때, 이미 다른 패킷 2가 그 채널을 점유하고 있다면 블럭킹이 발생한다. 이 때, virtual cut-through에서는 헤더 플릿이 진행하려는 채널이 사용 가능해질 때까지, 패킷 1의 헤더 플릿(H)과 그 뒤를 따르던 데이터 플릿(D1,D2), 그리고 아직 전송되지 않은 모든 데이터 플릿(D3,...., D7)들을 그 자신의 패킷 버퍼에 버퍼링 한다. 본 연구에서는 네트워크에서 충돌을 해결할 수 있는 혼합된 라우팅 방법으로서 HWCR 전략과 버퍼설계를 제안한다.

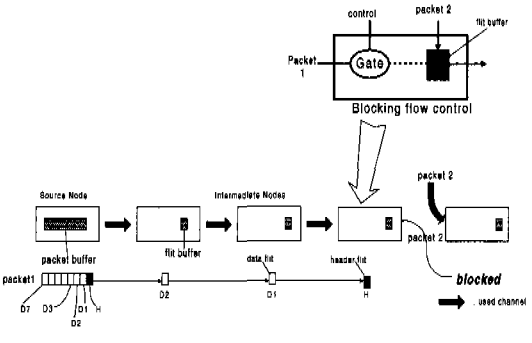

그림 7. 웜홀 안에서 블록킹

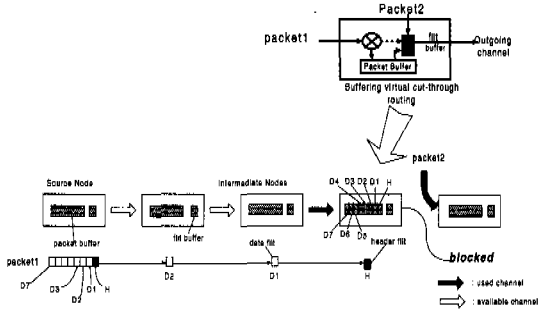

그림 8. Virtual Cut-Through에서 블럭킹

Ⅲ. 성능 향상을 위해 제안한 HWCR 기법

1. HWCR 알고리즘

네트워크에서 충돌을 해결할 수 있는 HWCR 알고리즘을 설계한다. 이 알고리즘은 기존의 BMIN에서 사용하는 선회 웜홀 라우팅 알고리즘의 단점을 보완하였다. HWCR 알고리즘은 경로 상에서 진행하는 두 경로가 어떤 링크에서 충돌할 때에 VCT로 전환하여, 진행하려는 경로 상에 있는 *SE* 안의 버퍼에 블럭된 플릿들을 모두 모아 저장하므로, 링크가 블럭되기 전에 점유하고 있던 경로 상의 링크를 풀어 주게 되어 다른 경로들이 이 링크를 사용할 수 있게 한다. 이후에 다시 블럭된 링크가 해제되면, 다시 이전의 웜홀 라우팅으로 목적지까지 라우팅을 계속한다. 다음은 HWCR 알고리즘2이다. 개선된 HWCR 방법을 사용한 라우팅 예를 다음 〈그림 9〉 에서 보여 주고 있다. 각 SE에는 기존의 작은 크기의 플릿 버퍼와, 그리고 특정 위치에는 큰 사이즈의 HWC 버퍼를 더 확보한다. 버퍼 크기와 위치는 다음의 HWCR 버퍼링 방법과 버퍼 모델에서 결정한다.

Algorithm2 : Conflict-free HWC turnaround routing algorithm
Input : Source address S : $s_{n-1}\ldots s_1 s_0$
Destination address D : $d_{n-1}\ldots d_1 d_0$
Procedure :
1. $t := FirstDifference\ (S, D)$;
2. IF $j = t$, 포트 l_{d_i}으로 turnaround connection을 택한다.
3. IF $j < t$ 이고, 입력포트 l_m으로부터 메시지가 오면, /* forward path */ 임의의 이용 가능한 포트 r_i로의 순방향 연결을택한다. $(0 \le m, i \le k-1)$.
4. IF $j < t$ 이고, 입력포트 r_m으로부터 메시지가 오면, /* backward path */
   ① IF 출력 포트 l_{d_i}에 연결된 링크 $C_i^{l_{d_i}}(0 \le i \le \log_k N-1)$가 블럭되어 있으면, 현재 SE의 HWC 버퍼에 현재 경로의 모든 플릿을 모은 후, $C_i^{l_{d_i}}$ 링크가 freed일 때 까지 대기한다.
   ② IF 진행할 링크 $C_i^{l_{d_i}}$ 가 블럭되지 않은 링크이면, l_{d_i}로의 역방향연결을 택한다.
   ③ IF 출력 포트(l_{d_i})에 연결된 링크 $C_i^{l_{d_i}}(0 \le i \le \log_k N-1)$가 블럭 됐었지만, 현재 링크가 free이면, 플릿 단위로 다시 l_{d_i}로 역방향연결을 택한다.
END.

[알고리즘2] 충돌 해결 HWCR 알고리즘

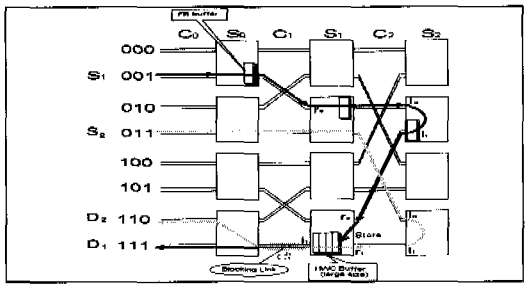

그림 9. 8 x 8 BMIN에서 HWC 라우팅

〈그림 9〉 에서 $S_2 \rightarrow D_2$ 경로에 우선 순위가 있다고 하면(혹은 두 경로 중에 한 경로가 무 순위로 먼저 선택되어지면), 블럭된 링크에서 이 경로를 먼저 진행시키고 다른 경로($S_1 \rightarrow D_1$)는 보류 상태가 된다. 이때 이전 경로 상에 있던 *SEs*의 플릿 버퍼 안에 저장된 플릿들 중 헤더 플릿 이후의 데이터 플릿은 모두 현재 블록된 *SEs*의 HWC 버퍼에 모아 둔다. 따라서 이전의 경로가 점유한 링크들은 모두 자유로운 상태가 된다. 이후에 우선순위경로의 전송이 완료된 후, 블럭되어 대기 중이던 경로($S_1 \rightarrow D_1$)를 다시 진행시킬 수 있다.

2. BMIN 에서 SE 구조 설계

BMIN에서, 각 *SE*들의 양방향 입출력 포트들에는 하나의 플릿을 저장할 수 있는 플릿 버퍼와 개선된 라우팅 알고리즘을 지원하기 위한 유한 개의 패킷을 저장할 수 있는 패킷 버퍼로 구성된다. 다음 <그림 10>에서는 스위칭 소자의 구조를 설계하였다. 이 유한 개의 패킷 버퍼는 진행하려는 링크가 블럭될 경우, 하나의 패킷에 해당되는 여러 개의 플릿들을 모아 저장하기 위한 용도로 사용된다.

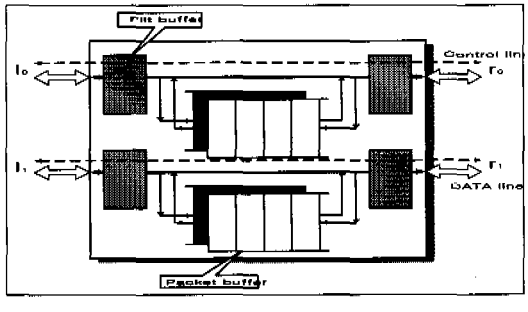

그림 10. SE의 구조

물론 플릿이 다시 조합되어야 하는 오버 헤드가 더 요구된다. 보통의 경우에는 웜홀 라우팅을 사용하기 때문에 이 패킷 버퍼는 사용되지 않다가, 현재 경로를 따라 진행하는 플릿들이 진행 링크가 블럭될 경우에만 이 패킷 버퍼를 사용한다. 그러므로 이 경로 중에 여러 플릿들이 점유하고 있는 헤더 플릿 이후의 블럭된 링크를 자유롭게 함으로서, 링크 자원을 효율적으로 사용할 수 있게 해 준다. *SE*에서 데이터는 입출력 포트를 통해 양방향으로 전송이 가능하며, 제어 라인을 통해 진행하려는 링크의 블럭 상태를 감지할 수 있다. 그리고 *SE* 내에서 데이터의 이동 경로는 두 경우가 존재한다. 하나의 경우는 일반적인 웜홀 라우팅을 사용할 때에 사용하는 플릿 버퍼를 통한 경로이고, 다른 하나의 경로는 진행하려는 링크에서 충돌이 발생할 때 패킷 버퍼를 통하여 라우팅 하는 경로이다. 따라서 제어 라인을 통해 진행 링크의 블럭된 상황이 감지되면, 라우팅 하던 모든 플릿들은 각 입출력 포트에 연결된 이 패킷 버퍼에 저장된 후 VCT 라우팅으로 전환한다.

3. HWCR에서 버퍼링 방법 및 모델 설계

BMIN의 특성은 최상위레벨에서 선회하는 구조이므로 팻-트리[15]와 함수적으로 동형으로 볼 수 있다. 따라서 single-end 구조를 갖는 BMIN의 트래픽은 상위 레벨에 가까운 쪽에 많은 패킷 충돌이 발생한다[16]. 스위치 크기가 k×k 일 경우, t = *First Difference*(S,D) 값에 따라 블록킹 횟수는 k^t 가 된다. 이것은 t-스테이지에서 경로의 수를 계산함으로써 쉽게 구할 수 있다. 순방향 경로에 있는 스테이지들로부터 선회하는 스테이지까지는 여러 개의 중복경로가 존재하기 때문에 최악의 상태에서도 충돌을 피할 수 있으므로 고려 할 필요가 없지만, 역방향 경로에 있는 *SE*들의 버퍼 크기는 스테이지가 감소하는 레벨 크기 순서로 버퍼 크기를 할당한다.

N=8인 경우를 예를 들면, *FirstDifference*(S,D)=2 임으로 두 번째 스테이지에서 선회한다. 그러므로 여러 중복경로를 사용하여 라우팅이 가능하기 때문에 순방향 경로에서 스테이지 2 까지는 충돌을 피할 수 있다. 그러므로 스테이지 1, 스테이지 0순으로 버퍼 크기를 할당하면 된다.

성능분석을 통해 최적의 버퍼 크기 결정하여 각 레벨의 버퍼 크기를 최적화 한다. MIN에서 버퍼 크기가 4 이상 일 경우는 처리율이 0.6 보다 더 향상되지 않기 때문에 큰 크기의 버퍼를 사용 할 의미가 없어진다. 오히려 네트워크 지연 시간만 길어지게 할 뿐이다[15][16].

버퍼링은 하위 레벨부터 상위 레벨로 점차로 감소하는 크기의 계층적 버퍼 크기를 할당한다. 이는

상위레벨에서의 채널 충돌 가능성이 증가하기 때문이다. BMIN의 각 *SE*들은 양방향 입출력 라인을 가지므로, 입출력 쪽에 모두 플릿 버퍼가 필요하고, 각 링크 당 한 쌍의 패킷 버퍼가 필요하다. 이 패킷 버퍼는 패킷 충돌을 효율적으로 관리하기 위해 사용하며, 보통 때에는 플릿 버퍼만 사용된다. 그러나 진행 링크에서 블럭된 상황이 검출될 때는 자동적으로 패킷 버퍼 경로로 패스가 스위치 된다[15].

본 연구에서는 기본적으로 BMIN이 팻-트리 구조를 가지고 있으므로, 계층적 버퍼할당 전략을 사용하였다. 이것은 팻-트리의 특성상 상위 레벨에서 패킷 충돌이 많이 발생한다[7]. <그림 11>은 N=16 일 경우에 계층적 패킷 버퍼 할당전략에 따른 패킷 버퍼 모델을 도식화한 것이다. 물론 각 SE 당 양방향의 입출력포트에 한 쌍씩의 플릿 버퍼가 존재한다. 버퍼의 크기는 $b_i=2^{i+1}$ $(0 \le i \le log_k N-1)$의 비율로 할당한다. 그러나 버퍼의 증가는 처음 2~3 스테이지까지만 증가하고 그 이상은 증가시키지 않는다. 이것은 버퍼 크기가 4 이상 일 때는 더 이상 성능 향상에 이득이 없다는 결과에 근거한다[15][16].

i-번째 스테이지의 버퍼 크기 할당은 다음과 같다.

$$0 \le i < 2 \ : \ b_i=2^{i+1}$$
$$i > 2 \ : \ b_i=b_{i-1}$$
$$i=log_k N-1 \ : \ b^l=1$$

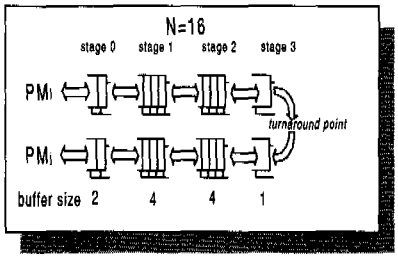

그림 11. 버퍼 모델

Ⅳ. 성능평가를 위한 트래픽 패턴 및 모의 실험

BMIN의 성능을 분석하기 위해서는 기존의 원홀 라우팅을 사용하는 BMIN과 제안된 HWCR 알고리즘을 사용하는 BMIN의 성능을 네트워크 통신 지연과 버퍼 측면에서 분석해 본다. 버퍼크기는 각 레벨에 따라 계층적으로 증가되었으므로, 통신지연을 증가시키지 않고 일정한 수준의 네트워크 처리율을 유지할 수 있는 최적의 버퍼 크기를 결정하기 위해 여러 가지 트래픽 패턴 하에서 분석한다. 시뮬레이션에 있어서 스위치 내부에 존재하는 버퍼의 크기는 네트워크의 성능을 유지하는 중요한 요소이므로 성능평가를 통해 본 연구에서 사용한 버퍼 할당 전략이 적절함을 증명 할 것이다. 입력 측에 새롭게 도착된 패킷은 동일한 확률로 N개의 입력 중에서 하나의 입력을 선택하며, MIN에서 대응하는 출력을 선택하는 방법은 선택된 트랙픽 유형에 의존한다. 일반적으로 트랙픽은 다음과 같이 분류할 수 있다.

1. 균일 트래픽 패턴

모든 패킷은 모든 소스 포트에 같은 비율로 들어가고, 예정된 모든 목적지 포트에 대해 패킷들이 같은 확률 $(1/N)$을 갖는 트래픽 패턴이다.[4] 균일 트래픽 패턴 (Uniform-traffic pattern)에서 모든 출력은 N개의 출력 중에서 독립적이며 균일한 확률로 선택된다.

2. 비균일 트래픽 패턴

(1) SSSD 트래픽 패턴

이 패턴에서 각 입력 소스는 그의 모든 패킷을 하나의 출력 목적지에 보낸다. 따라서 SSSD(Single Source to Single Destination)패턴에서는 베니안 네트워크를 비 균일 트래픽 패턴(Non-uniform traffic pattern)으로 만들게 된다[1]. 베니안 네트워크는 그 구조상 스테이지 사이의 링크가 다른 트래픽 경로에 의해 공유되므로[1], 더 적은 스위칭 소자가 요구된다. 따라서 SSSD트래픽은 다른 트랙픽에 의해 공유되는 공통 링크 상에 극도의 부하를 가져오게 한다. 커다란 데이터나 영상과 같은 광대역 트랙픽은 SSSD트래픽에 의해 모델 링 될 수 있다. 다음 행렬은 최대의 충돌 트래픽 패턴을 보여주는 행렬이다.

$$\begin{bmatrix} \lambda_{00} & 0 & 0 & \dots & 0 \\ 0 & \lambda_{11} & 0 & \dots & 0 \\ 0 & 0 & \lambda_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \lambda_{nn} \end{bmatrix}$$

다른 중요한 트랙픽 패턴은 균일 트래픽 패턴에 내포된 하나의 SSSD 경로와 연관된다. 이 트래픽 패턴의 부하 행렬, L 은 한 요소 eij 를 갖는데, i, j 는 각각 단일 소스, 단일 목적지 트래픽의 소스 포트와 목적지 포트이다. 또한 ein 과 emj 는 0이다. 여기서 n n과 m은 각각 i, j가 아니다. 행렬 요소의 나머지는 모두 균일 분포를 갖는다. 이러한 트래픽 패턴을 혼합 트래픽 패턴이라고 한다. 이러한 트래픽은 하나의 전용된 비디오 채널과 많은 음성 및 데이타 채널을 갖는 네트워크의 트래픽 유형을

반영한다.

$$\begin{bmatrix} \lambda & 0 & \lambda & \cdots & \lambda \\ 0 & \lambda_{SSSD} & 0 & \cdots & 0 \\ \lambda & 0 & \lambda & \cdots & \lambda \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \lambda & 0 & \lambda & \cdots & \lambda \end{bmatrix}$$

다음 <그림 12 >에서는 SSSD 트래픽 모델 형태를 나타냈다. 여기서, 소스의 각 노드들은 목적지 노드 중 어떤 한 노드를 선택한다.

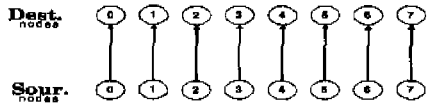

그림 12. SSSD 트래픽 모델

(2) Hot-Spot 트래픽 패턴

여러 개의 패킷이 하나의 목적지로 향하는 트래픽 패턴이다[7]. 출력에서 버퍼링 기능을 갖는 스위치는 이 스위치가 출력 버퍼에서 여러 개의 공간적 엑세스 포인트(spatial access point)를 가지고 있지 않다면 성능이 상당히 저하되는 결과를 초래한다[7]. 출력 0으로 향하는 부하가 다른 출력으로 향하는 부하보다 많다고 가정하자. 또한 각 시간 슬롯에서, 각 셀 이 동일한 확률,ρ 0 로 도착한다고 가정하자. 즉, offered load, ρ 0 는 ρ 0 = hρ 0+(1 - h)ρ 0 로 표현된다. 여기서 매개변수 h 는 출력0으로 향하는 부하의 비율이다.

임의의 x에 대하여, ρ 0x(j,n) (0≤j≤N-1) 는 다음과 같이 주어진다.

$$\rho_{0x}(0,n) = h\rho_0 + (1-h)\frac{\rho_0}{N},$$

$$\rho_{0x}(j,n) = (1-h)\frac{\rho_0}{N}, \quad (1 \le j \le N-1)$$

다음의 행렬은 hot-spot 트래픽의 분산 행렬을 나타낸다.

$$\begin{bmatrix} h+\frac{(1-h)}{N} & \frac{1-h}{N} & \frac{1-h}{N} & \cdots & \frac{1-h}{N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h+\frac{1-h}{N} & \frac{1-h}{N} & \frac{1-h}{N} & \cdots & \frac{1-h}{N} \end{bmatrix}$$

〈그림 13〉 은 Hot-Spot 트래픽 패턴의 형태를 예시하였다.

(3) 수평 트래픽 패턴

수평 트래픽 패턴((Horizontal Traffic Pattern)은

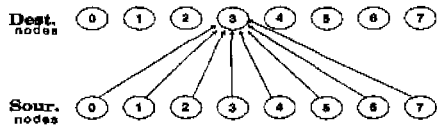

그림 13. Hot-Spot 트래픽 패턴

입력 i (i=1,2,3,...,N-2)에 도착된 요청 (i-1, i, i+1)에 대해 대응하는 출력을 갖고, 동일한 확률로 선택된다. 첫 번째 i=0일 때 도착된 요청은 (N-1,0,1)이며 마지막 i=N-1 일 때 도착된 요청은 (N-2,N-1,0)로 각각 동일한 확률로 선택되어 출력된다. 이 패턴은 균등 분포로 얻어진 것 보다 다소 넓게 스위치 내에서 트래픽 분산을 가져온다[11].

〈그림 14 〉는 수평 트래픽 패턴을 보여준다

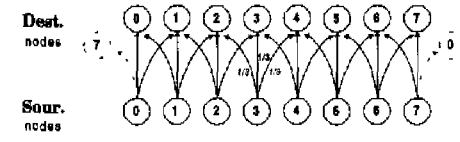

그림 14. 수평 트래픽 패턴

(4) 대각선 트래픽 패턴

대각선트래픽(Diagonal Traffic Pattern)의 입력 i(i=0,1,2,....,$\frac{2}{N}$-1, $\frac{2}{N}$, $\frac{2}{N}$+1,...,N-1)에 도착된 요청은 중앙을 중심으로 대각선으로 동일한 확률로 선택되고, 대응하는 출력을 갖는다. i(i=0,1,2,...,$\frac{N}{2}$-1)일 때는 $\frac{N}{2}+i$에 대응하는 출력을 갖고, i(i=$\frac{N}{2}$, $\frac{N}{2}$+1,...,N-1)일 때는 $\frac{N}{2}-i$에 대응하는 출력을 갖는다.

〈그림15〉 의 트래픽 패턴에 의해 얻어진 스위치 내에서의 트래픽 분포는 위 두 트래픽 패턴으로부터 얻어진 것 보다 더 넓게 분산된다.

그림 15. 대각선 트래픽 패턴

(5) CR 트래픽 패턴

CR(Consecutive Requests Traffic Pattern) 트래픽 패턴은 요청의 순차가 각 소스 노드에서 생성되

고, 연속적으로 다음 목적지 노드(sink node)를 요청하는 형태의 트래픽 패턴이다[5][12]. 한 소스에서의 요청의 순차는 중간에 다른 요청의 개입 없이 연속적으로 계속된다. 입력 i가 메모리 모듈 MM_3로 향한다면 다음 요청은 MM_4, MM_5, MM_6...순으로 연속적인 요청 패턴을 이루게 된다. 이러한 패턴은 벡터 엑세싱에서 주로 발생한다.

〈그림 16 〉은 CR 트래픽 패턴 모델이다.

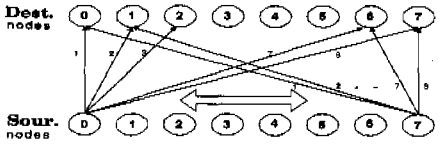

그림 16. CR 트래픽 패턴 모델

즉 커다란 수치 배열과 배열 엑세스와 연관된 많은 양의 반복 계산은 공학이나, 과학적인 응용에서 주로 사용된다. 또한 캐쉬 블록페치나 프로그램 로딩에서도 발생한다.

V. 성능 분석

1. 성능척도

본 논문에서 사용하는 성능척도는 평균 패킷 지연시간과 사용된 버퍼 양이다. 먼저 평균 패킷 지연시간은 근원지 프로세서 모듈에서 전체 메시지의 패킷이 생성된 후 목적지에 마지막 플릿이 도착할 때까지의 평균 시간을 나타낸다. 이 값을 측정하기 위해서는 패킷 헤더(플릿 헤더)에 있는 패킷 생성 시간과 목적지에 도착한 시간차로부터 측정된다. 이 값을 균일 및 비균일 트래픽 패턴에 따라 기존의 웜홀 라우팅과 VCT 그리고 제안된 HWCR 라우팅과 비교한다. 다음으로 도입하는 척도는 평균 버퍼 사용량이다. 버퍼 요구 정도를 측정하기 위해서 각 SE에서 사용된 평균 패킷 버퍼의 크기를 척도로 사용한다. 이 값은 데이터 전송 동안, 각 SE에 큐된 패킷의 평균수를 구함으로써 측정할 수 있는데, 이 값을 네트워크 크기로 나눈 값이다. 이 성능척도도 균일 및 비균일 트래픽 분포에 따라 분석한다[5].

BMIN 시뮬레이션을 위한 가정들은 다음과 같다;

① BMIN의 스위칭 소자는 동기 적으로 동작한다.

② 스테이지 사이클은 10 클럭 주기로 한다.

③ N개의 프로세서 모듈로부터 BMIN에 도착하는 패킷은 포아송 분포를 가지며, 그리고 모든 목적지는 동일한 확률($1/N$)로 참조된다[4][12].

④ 각 스위칭 소자는 유한개의 버퍼를 갖는다.

⑤ 두 개의 플릿이 동시에 같은 출력포트로 전송을 요구하는 경우, 임의의 하나를 선택하고, 나머지는 버퍼에 저장한 후 다음 사이클에 전송한다.

⑥ 하나의 패킷은 20개의 플릿으로 나누어지며, 각 플릿의 크기는 8 바이트로 가정한다.

시뮬레이션은 SLAM II를 사용하여 이산사건 모델로 모델링 되어 수행되었다, 시뮬레이션에 있어서 스위치 내부에 존재하는 버퍼의 크기는 네트워크의 성능을 유지하는 중요한 요소로 사용한다.

2. 균일 트래픽 패턴 하에서의 성능

(1) 평균 패킷 지연시간

제안된 HWCR 라우팅 기법은 VCT보다 적은 양의 패킷 버퍼를 요구한다. 그러면서도 지연 시간은 거의 VCT에 가까운 성능을 보여준다. 이 값은 기존의 BMIN에서 사용하는 웜홀 라우팅에서보다는 현저히 적은 값이다. 부하가 0.4일 경우 웜홀(40%)은 HWCR(25%)의 약 2배의 지연을 보인다. 이 결과는, HWCR 기법에서는 웜홀 라우팅 기법보다 라우팅 중에 플릿들이 링크를 점유하여 링크가 블록된 것이 현저히 적음으로 해서, 다른 메시지의 전송시에 대기시간이 거의 없기 때문이다. 다음 <그림 17> 그래프는 균일 분포에서 N=256, 20 flits/packet 인 경우에 부하 대 평균 패킷 지연시간을 보여주고 있다.

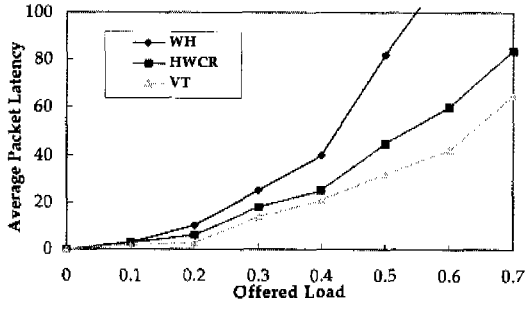

그림 17. 균일 분포 하에서 평균 패킷 지연시간

(2) 평균 버퍼 크기

〈그림 18〉 은 평균 버퍼 크기를 나타낸다. 이 그래프에 따르면, 제안된 HWCR 기법이 VCT보다 훨씬 적은 양의 버퍼를 필요로 함을 알 수 있다.

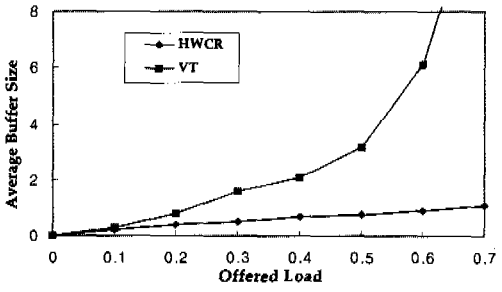

그림 18. 균일 분포 하에서의 평균 버퍼 크기

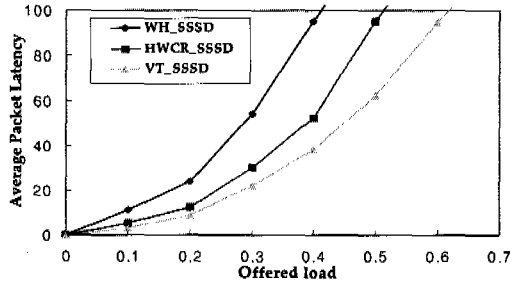

그림 19. SSSD 트래픽하에서 평균 패킷 지연시간

부하가 0.6 일 경우 VCT의 버퍼 크기(6.2)는 HWCR 기법의 버퍼 크기(0.8)의 약 6배가 요구된다. 그리고 작은 입력 부하(ρ =0.1) 하에서는 실제 설계된 버퍼 양 보다 상당히 적은 양만이 사용됨을 알 수 있다. 이 결과로 보아 각 *SE*의 버퍼에 저장된 패킷이 평균적으로 적음을 알 수 있다.

3. 비균일 트래픽 패턴 하에서의 성능

(1) 평균 패킷 지연

이 절에서는 비 균일한 트래픽 패턴 하에서 제안된 라우팅 기법들의 성능을 비교한다. 먼저 SSSD 트래픽은 베니안 유형의 네트워크로 내부적으로 공유되는 링크가 많으므로 가장 많은 충돌이 있는 트래픽 유형이며, Hot-spot 트래픽은 하나의 출력으로 모든 입력 측의 요청이 집중되는 패턴이다. 그리고 수평 및 대각선 트래픽 패턴, 그리고 연속적인 요청 패턴을 만드는 CR 트래픽 패턴에 대한 분석을 기술한다. 이러한 트래픽 환경 하에서 패킷 지연시간 분석을 통하여 제안된 라우팅 기법의 효율성을 평가한다.

a. SSSD 트래픽 패턴

〈그림 19〉는 균일 트래픽 분포가 아닌 비균일 트래픽 패턴의 일종인 SSSD 트래픽 패턴을 적용하였을 때, 제안된 HWRC 기법 및 기존의 기법들간의 평균 패킷 지연시간을 보여주고 있다. 그래프에 나타난 것처럼 부하가 0.4 일 때 VCT의 패킷 지연(38)과 HWCR의 패킷 지연(53)은 큰 차이가 나지 않는다. 그러나 웜홀의 패킷 지연(95)은 거의 2배의 지연을 보이고 있다. 제안된 HWCR 기법이 SSSD 분포에서도 우수한 성능을 나타냄을 알 수 있다.

b. Hot-Spot 트래픽 패턴

<그림 20>는 hot-spot 트래픽하에서 각 기법간의 성능을 보여준다. 이 경우에 h=0.01을 가정하였다. h는 전체 발생되는 패킷 중에서 hot-spot 트래픽 부분을 나타낸다. 세 개의 그래프 기울기가 크므로 이 트래픽 하에서의 성능은 여러 비균형 트래픽 중에서도 최저의 성능을 나타낸다. 예를 들어 Hot-Spot 시 부하가 0.2 일 때 HWCR의 지연은 약 35 이지만, SSSD 는 10 이다. 그리고 다음 그래프에서도 HWCR 기법과 VCT는 ρ =0.2 때 평균 패킷 지연은 약 26~36 이지만 웜홀 기법은 약 70으로 2배 이상의 지연을 보여 주고 있다. 그러므로 Hot 트래픽 하에서도 우수함을 보여주고 있다.

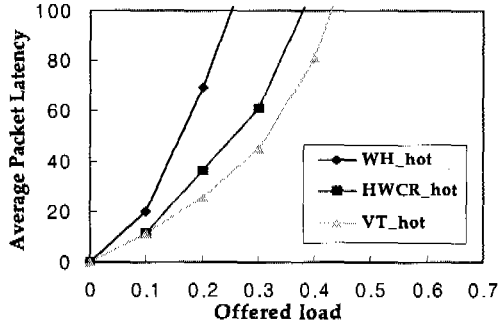

그림 20. Hot-spot 트래픽 하 에서 평균 패킷 지연시간

c. 수평 및 대각선 트래픽 패턴

<그림 21>은 수평 트래픽 및 대각선 트래픽 패턴에 대한 그래프이다. 비 균일 트래픽 분포 중에서 가장 균일 트래픽 분포에 근사하는 성능을 보여주고 있다. 그러나 대각선 트래픽 패턴(DT)의 경우는 수평 트래픽(HT) 보다 네트워크 내에서 더 넓게 분산된다. 이 그래프에서, 대각선이나 수평 트래픽 유형 하에서 패킷 지연은 웜홀 라우팅이 가장 많은 영향을 받음을 알 수 있다. HWCR과 VT의 패킷 지연은 거의 근사한 결과를 보여준다. ρ =0.4 일 때 HWCR과 VCT의 평균 패킷 지연은 약 24~27인 반면 웜홀은 43~56이므로 약 2배의 지연을 보인다.

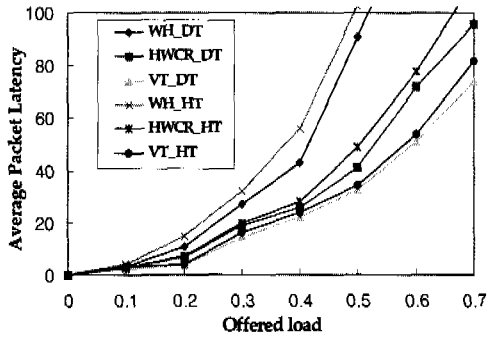

그림 21. 수평 및 대각선 트래픽 하에서 평균 패킷 지연 시간

d. CR 트래픽 패턴

<그림 22>는 목적지에 대한 요청이 연속적으로 만들어지는 CR패턴에 대한 결과이다. 이 경우에는 요청이 항상 일정한 순서로 만들어진다. 따라서 BMIN 내부에서 링크 충돌로 인한 부하가 많을 때는 매우 증가하게 된다. 부하가 0.1~0.4 사이에서 HWCR과 VCT의 패킷 지연(38~40)은 거의 같고, 웜홀과의 관계는 약 2배(95)가 넘는다. 따라서 〈그림 22〉에서 나타난 이러한 트래픽 형태에서는, 특히 HWCR을 포함하여 VT 라우팅 유형의 성능이 우수함을 알 수 있다.

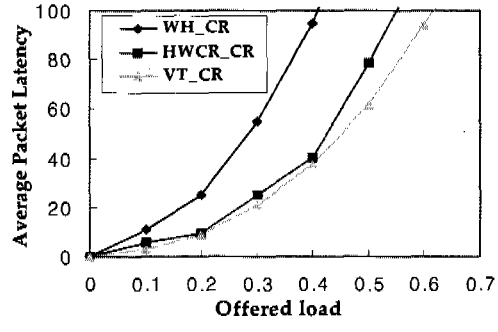

그림 22. CR 트래픽하에서 평균 패킷 지연시간

(2) 평균 버퍼 사용량

다음 그래프는 비 균일 트래픽 분포 하에서 평균 버퍼 크기를 보여주고 있다. 여기서는 웜홀라우팅에 비해 버퍼를 많이 사용하는 VCT 라우팅과 제안된 HWCR 기법의 버퍼 사용량 비교에 초점을 맞춘다. 비균형 트래픽 하에서 제안된 HWCR 기법이 VCT 보다 훨씬 적은 양의 버퍼를 필요로 함을 알 수 있다. 부하(ρ =0.4)일 경우 VCT의 버퍼 크기(2.2~6.9)는 HWCR 기법의 버퍼 크기(1.1~2.2)의 약 2~3배가 요구된다. 그리고 적은 입력 부하(ρ =0.1) 하에서는 설계된 버퍼 양 보다 적은 버퍼 양만 실제 사용됨을 알 수 있다. 이 결과로 보아 각 SE의 버퍼에 저장된 패킷이 평균적으로 적음을 알 수 있다. HWCR 라우팅은 계층적 버퍼 할당 기법을 사용하고 있으며, VCT의 경우는 일정한 크기의 버퍼가 모든 스위칭 소자에 존재한다. 이 분석에서는 라우팅 시에 사용된 버퍼 량을 비교하기 위해 VCT 라우팅 기법과 비교한다.

a. SSSD 트래픽 패턴

제안된 HWCR 기법이 균일 분포에서의 결과와 같이 비균일 트래픽 하에서도 VCT보다 훨씬 적은 양의 버퍼를 필요로 함을 <그림 23>의 그래프에서 보여 준다. 적은 부하시(ρ =0.2) VCT (1.7)는 HWCR (0.6) 보다 약 두배의 버퍼를 필요로 하지만, ρ =0.4 일 경우 VCT(4.3) 는 HWCR(1.8) 보다 약 3배의 버퍼가 요구된다. 이 결과로 부터 제안된 HWCR은 일반적인 비균일 분포에서도 VCT 보다 적은 하드웨어를 사용함을 알 수 있다.

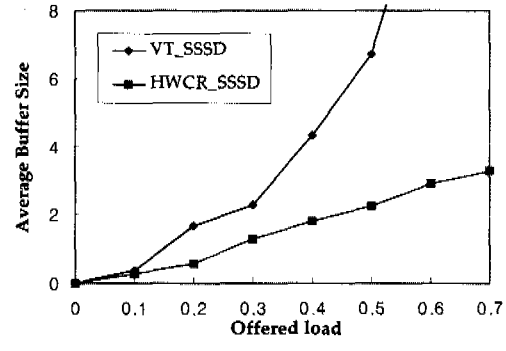

그림 23. SSSD 트래픽 하에서의 평균 버퍼 크기

b. Hot-Spot 트래픽 패턴

<그림 24>는 hot-spot 트래픽 패턴 하에서 사용된 평균 버퍼 량을 보여주고 있다. 이 그래프에서는 보여지는 것처럼 평균 버퍼 사용량은 SSSD 트래픽 패턴의 경우에서보다는 증가하는 것을 볼 수 있다. hot-spot 트래픽은 분포상 비 대칭성에 기인하기 때문이다. 여기서 부하가 0.2 일 경우 HWCR(0.9)과 VCT(1.8)는 약 2배의 버퍼 사용량이 차이가 나고, 0.4 일 경우 HWCR(2.2)과 VCT(6.9)는 약 3배의 차이가 난다. 그러나 HWCR의 경우는 VCT의 경우에서 보다 필요로 하는 버퍼량이 부하에 대하여 그렇게 급속하게 증가하지 않는 결과를 보여주는데, 이러한 결과는 HWCR에서는 계층적 버퍼할당 기법

의 적용으로 더 우수한 성능결과를 보여주는 것으로 분석 할 수 있다.

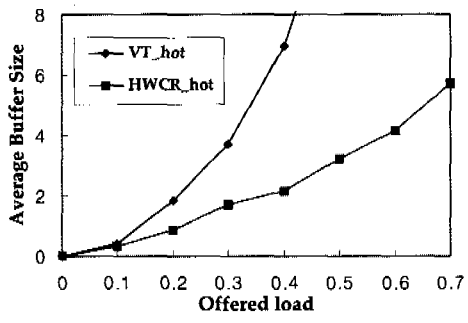

그림 24. hot-spot 트래픽하에서의 평균 버퍼 크기

c. 수평 및 대각선 트래픽 패턴

<그림 25>는 수평 트래픽 및 대각선 트래픽 패턴에 대하여 요구되는 버퍼 량을 보여주는 그래프이다. 이 경우에는 균일 트래픽에 가장 근사하는 결과가 얻어진다. 부하 0.4 에서 VCT의 수평(4.3)과 대각선 트래픽(4.1)은 HWCR의 수평트래픽(1.9)과 대각선 트래픽(1.3)의 약 3배의 버퍼가 요구된다. 그러므로 HWCR의 경우가 부하에 민감하지 않음을 알 수 있다. 본 연구에서 대각선 트래픽(DT) 형태가 수평 트래픽(HT)에서 보다 더 적은 버퍼 양을 필요로 하는 것을 볼 수 있는데, 이것은 상호 통신하려는 프로세서 모듈들이 서로 먼 거리에 있는 노드들끼리 연결되어 네트워크내의 부하가 분산되기 때문이다. 특히 HWCR의 경우는 HT와 DT 패턴간의 성능 차이가 매우 큼을 보여준다

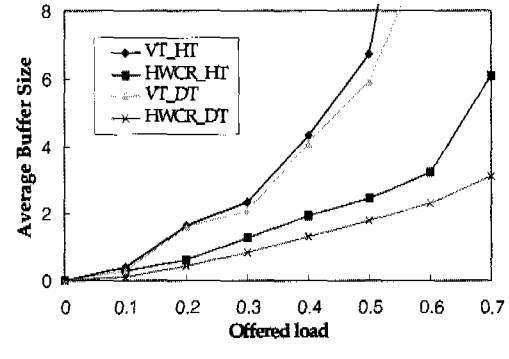

그림 25. 수평 및 대각선트래픽 하에서 평균 버퍼 크기

d. CR 트래픽 패턴

<그림 26>은 연속적인 요청에 의해 만들어지는 CR 트래픽 패턴에 관한 결과다. 이 경우는 각 라우팅 경로 상에 이웃하는 프로세서 모듈들이 연속적으로 통신하기 때문에 부하의 증가에 따라 여러 라우팅 패스간에 공통 링크를 사용하는 경우가 급속히 증가하게 된다. 부하가 0.1 이하일 경우는 이 두 라우팅은 거의 버퍼를 사용하지 않기 때문에 버퍼 크기가 필요로 하지 않는다. 부하가 0.4 일 경우 HWCR은 버퍼 크기는 약 2.1 이고 VT는 약 6.5 임으로 약 3배가 증가된다. 따라서 이 경우에 네트워크 성능은 매우 급격히 저하되므로 적당한 크기의 버퍼를 사용하여야 한다.

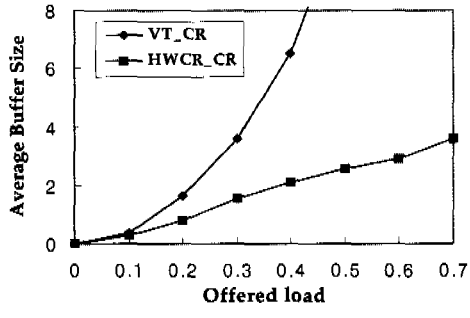

그림 26. CR트래픽 하 에서 평균 버퍼 크기

Ⅵ. 결 론

본 논문에서는 양방향 MIN 상에서 트래픽이 많은 웜홀 스위칭 하에서 링크의 충돌이 발생하였을 때, 성능이 급격히 저하되는 것을 방지하기 위한 새로운 라우팅(HWCR) 기법을 제안하였다. 그리고 제안한 라우팅 기법을 위해 적당한 하드웨어 자원(SE, Buffer)을 개발하여 사용하였고, 이 기법에 알맞는 알고리즘을 개발하여, 안정되고, 높은 수준의 성능을 얻을 수 있도록 하였다. 그리고 제안된 HWCR은 멀티캐스팅 하에서도 쉽게 적응할 수 있는 라우팅 알고리즘임을 보였다. 설계된 라우팅 기법의 성능을 분석하기 위해 여러 가지 트래픽 패턴 하에서 VCT와 웜홀과 비교하였다. 웜홀과 VCT 라우팅 기법의 장점을 혼합한 HWCR 기법은 링크 상에서 웜홀 기법으로 라우팅을 수행하다가 블럭킹이 발생할 경우, VCT 라우팅 기법으로 전환하여 링크의 사용을 원활하게 하고 다시 블럭된 노드가 해제되면 원래의 웜홀 라우팅 방법으로 되돌아가는 방법이다. 이러한 두 기법을 장점을 도입함으로써 링크 자원을 효율적으로 사용하게 되었고, 결국 전체 네트워크의 성능이 향상됨을 보였다. 또한, 실험 결과 버퍼크기가 네트워크의 첫 몇 개 스테이지부터

증가하다 이 후 몇 스테이지에서 선회 지점까지 계속 커질 필요가 없다는 버퍼 할당 전략의 구현 가능성을 제시하였다. 이를 위해 HWCR, VCT 라우팅, 웜홀 라우팅을 균일 및 비균일 트래픽 패턴 하에서 시뮬레이션한 결과, 부하(ρ)가 0.1 이하 일 경우 거의 비슷한 버퍼 사용을 하지만, ρ =0.4 이상 일 경우, HWCR은 VCT 보다 약 2~3배 버퍼 크기가 절약되었다. 통신지연에 의한 처리율은 부하가 0.1 이하 일 경우 모든 트래픽 하에서 거의 비슷하였지만, 0.4 일 경우 HWCR이 VCT에는 거의 근접하였고, 웜홀 라우팅 보다 약 2.5~4배 우수함을 보였다.

차기 연구방향으로는 복잡한 트래픽 패턴에 하에서 하드웨어 비용과 성능 간에 심도 있는 분석이 요구되며, 여러 가지 트래픽 상에서 오류 허용 방안 및 데드락(deadlock) 문제의 연구가 더 필요하다.

참 고 문 헌

[1] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessing systems," in Proc. of the First ISCA, pp. 21-28, 1973.

[2] D. M. Das and J. R. Jump, "Analysis and simulation of buffered delta networks," IEEE Transaction on Computers, vol. C-30, pp. 273-282, Apr. 1981.

[3] C.Kruskal and M. Snir, "The Performance of multistage interconnection networks for multiprocessors," IEEE Transactions on Computers, vol. C-32, pp.1091-1098, Dec. 1983

[4] C.E. Leiserson, "Fat-Trees : Universal Networks for hardware-efficient supercomputing," IEEE Trans. on Computers, vol. C-34, pp. 892-901, Oct. 1985.

[5] G. Pfister and A. Norton, "Hot spot contention and combining in multistage interconnect networks," IEEE Transactions on Computers, vol. C-34, pp. 943-948, Oct. 1985.

[6] W.J.Dally, "Virtual channel flow control," in Proc. of the 17th International Symposium on Computer Architecture, pp. 60 -68, May. 1990.

[7] P.K. McKinly, H.Xu, A.H.Esfahanian, and L.M. Ni, "Unicast-based multicast communication in wormhole-routed direct networks," Proc. 1992 ICPP, vol-II, pp.10-19, Aug. 1992.

[8] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct network," IEEE Computer, vol. 26, pp. 62-76, Feb. 1993.

[9] Kai Hwang, "Advanced Computer Architecture: Parallelism, Scalability, Programmability" pp. 5-18, 1993

[10] J. Ding and L. N. Bhuyan, "Finite buffer analysis of multistage interconnection networks," IEEE Transactions on Computers, vol. 43, pp. 243-247, Feb. 1994.

[11] Hyunmin Park and Dharma P. Agrawal, "WICI:An Efficient Switching Scheme for Large Scalable Networks," IEEE 6th PDP, 1994.

[12] Message Passing Interface Forum, "MPI : A Message Passing Interface Standard," tech. rep. Univ. of Tennessee, Mar, 1994.

[13] L.M.Ni, Y. Gui, and S. Moore, "Performance evaluation of switch-based wormhole networks," Tech. Rep. MSU-CPS-ACS-96, Michigan State University, Department of Computer Science, July 1994

[14] Lionel M. Ni, Yadong Gui and Sherry Moore, "Performance Evaluation of Switch-Based Wormhole Networks," ICPP, Aug. 1995.

[15] ChangSoo Jang, SungChun Kim, "A Design and Analysis of A New Hybrid WC Routing Scheme in the Bidirectional Multistage Interconnection Network", To be appeared at the 1996 Fifteenth IASTED International Conference, Innsbruck, Austria, Feb. 19-22, 1996.

[16] ChangSoo Jang, SungChun Kim,"HWCR: A Cost-Effective Routing Scheme in Bidirectional Multistage Interconnection Network", The IASTED International Conference on MODELLING, SIMULATION AND OPTIMIZATION, pp. 226-242, Australia, May, 1996.

장 창 수(Chang Soo Jang) 정회원

1980년 2월 : 조선대학교 전자공학과(공학사)
1982년 8월 : 건국대학교 전자공학과(공학석사)
1997년 2월 : 서 강 대 학 교 컴퓨터공학과(공학박사)

1984년~현재 : 여수대학교 컴퓨터공학과 교수
<주관심 분야> 병렬처리 구조, 컴퓨터네트워크, DSP