분산 공유메모리 환경의 다단계 버스망에서 트래픽에 적응하는 동적 라우팅 알고리즘

홍 강 운[†] · 전 창 호[††]

요 약

본 논문은 분산 공유메모리 환경의 다단계 버스망을 위한 동적 라우팅 방법을 제안한다. 제안된 라우팅 방법의 특징은 다단계 버스망이 제공하는 잉여경로를 활용하고 스위치 트래픽에 따라 적응적으로 경로를 결정하여 스위치의 트래픽을 분산시키는 것이다. 구체적으로는 잉여경로 상의 다음 단계 스위치의 트래픽 정도가 높고 낮음을 판단하여 트래픽 정도가 낮은 스위치로 패킷을 전달한다. 그 결과 평균 응답시간과 스위치 상의 평균 대기패킷수를 줄이는 효과를 얻는다. 프로세서수와 스위치 크기를 변화시키면서 시뮬레이션을 하여 제안된 알고리즘이 잉여경로를 고려하지 않는 기존의 알고리즘에 비하여 평균 응답시간은 약 9%, 스위치 상의 평균 대기패킷수는 21.6% 정도 향상시킨다는 것을 보여준다.

A Dynamic Routing Algorithm Adaptive to Traffic for Multistage Bus Networks in Distributed Shared Memory Environment

Kang Woon Hong[†] · Chang Ho Jeon[††]

ABSTRACT

This paper proposes an efficient dynamic routing algorithm for Multistage Bus Networks(MBN's) in distributed shared memory environment. Our algorithm utilizes extra paths available on MBN and determines routing paths adaptively according to switch traffic in order to distribute traffic among switches. Precisely, a packet is transmitted to the next switch on an extra path having a lighter traffic. As a consequence the proposed algorithm reduces the mean response time and the average number of waiting tasks. The results of simulations, carried out with varying numbers of processors and varying switch sizes, show that the proposed algorithm improves the mean response time by 9% and the average number of waiting tasks by 21.6%, compared to the existing routing algorithms which do not consider extra paths on MBN.

키워드 : 다단계 버스망(Multistage Bus Network), 라우팅 방법(Routing Algorithm), 버디관계(Buddy Property), 잉여경로(Extra Path), 시뮬레이션(Simulation)

1. 서 론

병렬 시스템은 다수의 프로세서, 메모리 모듈, 그리고 이들을 연결해주는 연결망으로 구성된다. 병렬 시스템의 성능은 여러가지 요인으로부터 영향을 받지만, 특히 프로세서들 사이에 빈번한 데이터 통신이 요구되는 작업을 수행할 경우 연결망을 통한 데이터 전송의 효율성에 의해 크게 좌우된다[1]. 효율적인 데이터 전송을 하려면 프로세서와 메모리 모듈 사이에 가능한 한 빠른 시간 내에 원하는 통신을 할 수 있도록 경로가 제공되어야 한다. 이런 이유로 여러가지의 연결망 구성과 라우팅 알고리즘에 대한 연구가 이루어져 왔다[2-6].

다단계 버스망[7,8]은 다단계 연결망의 한가지 형태로서 단계 사이의 연결 스위치로 크로스바 대신 버스를 사용하는 망을 말한다. 다단계 버스망은 기본적으로 다단계 연결망의 특징을 갖는 동시에 계층적 버스 구조를 띄고 있다. 이것을 데이터 전송 기능면에서 보면 모든 단계에서 동일한 대역폭을 가진다는 다단계 연결망의 장점과 단계별로 프로세서들 간에 그룹화된 지역성을 가진다는 계층적 버스 구조의 장점을 함께 가졌다고 할 수 있다. 따라서, 다단계 연결망에 사용할 수 있는 간단한 라우팅 알고리즘을 활용하면서 지역성에 따라 경로상의 스위치 수를 줄일 수도 있다. 본 논문의 관심대상인 다단계 버스망에서의 스위치는 버스를 의미하고 논문에서 스위치와 버스는 같은 의미로 혼용된다.

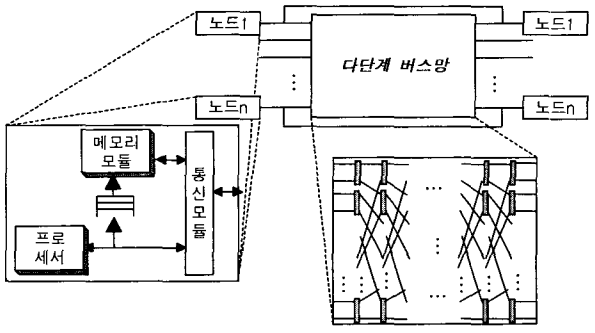
연결망이나 적용되는 알고리즘의 성능은 연결망에 적용되는 메모리 환경에 의해 영향을 받으며 그 종류는 시스템 메모리의 분산 여부에 따라 공유 메모리 환경과 분산 메모리 환경으로 나뉜다. 분산 메모리 환경과 공유 메모리 환경 각각의 장점을 이용하는 분산 공유메모리 모델이 제안되었다.

† 정 회 원 : 한국전자통신연구원 네트워크연구소 연구원
†† 종신회원 : 한양대학교 전자컴퓨터공학부 교수
논문접수 : 2002년 6월 20일, 심사완료 : 2002년 11월 5일

분산 공유메모리란 물리적으로는 독립된 메모리 공간으로 분산되어 있지만 논리적으로는 하나의 공유 메모리로 구현되는 메모리 모델이다. 이 메모리 모델에서 메모리 모듈은 프로세서의 지역 메모리로 사용되는 동시에 전체 공유 메모리의 부분으로도 사용된다. 프로그래밍 환경의 편리성과 통신 기법의 단순성을 제공하는 공유 메모리 모델의 특성과 시스템 확장성을 제공하는 분산 메모리 모델의 특성을 동시에 이용하고자 하는 것이 분산 공유메모리이다[9, 10].

본 논문에서는 분산 공유메모리 환경에서의 다단계 버스망에 효율적인 동적 라우팅 알고리즘을 제안한다. 기존의 분산 공유메모리 환경의 다중 프로세서 시스템을 위한 라우팅 알고리즘은 라우팅 경로 상의 스위치의 수가 최소가 되는 라우팅 방법을 선택하고 있다[7, 8]. 또한, 같은 프로세서가 같은 메모리 모듈을 참조하는 경우 항상 동일한 경로를 통과하게 되어있다. 본 논문에서 제안하는 라우팅 알고리즘은 다단계 버스망의 특성을 고려하여 스위치의 트래픽 정도에 따라 데이터 전송경로를 다르게 설정한다. 그리고, 프로세서 수와 스위치 크기를 변화시키는 시뮬레이션을 수행하여 평균 응답시간과 스위치와 메모리 모듈의 큐에 기다리는 평균 대기패킷수의 관점에서 제안된 알고리즘과 기존 라우팅 방법의 성능을 비교 분석한다.

본 논문의 2장에서는 다단계 버스망의 특성, 기존의 라우팅 알고리즘 그리고 최적 경로결정 알고리즘을 설명하고 이들 알고리즘의 문제점을 제시한다. 3장에서는 동적 라우팅의 개념을 설명하고 트래픽을 고려하는 동적 라우팅 알고리즘을 제안한다. 4장에서는 동적 라우팅 알고리즘의 성능을 평가하기 위한 시뮬레이션 모델과 시뮬레이션 결과를 제시한다. 마지막으로 5장에서 결론을 맺는다.



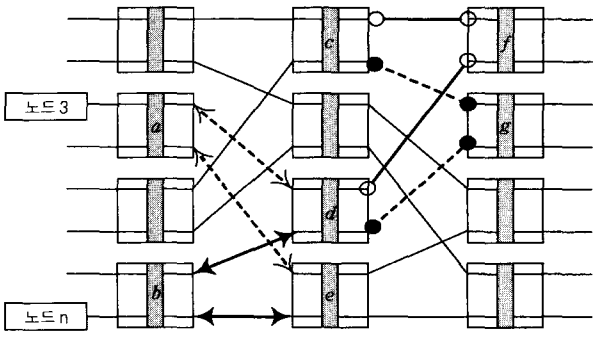
(그림 1) 다단계 버스망 기반의 분산 공유메모리

2. 다단계 버스망과 기존 라우팅 방법

2.1 다단계 버스망 구조

일반적으로 다단계 연결망은 프로세서와 메모리 모듈을 연결하는 형태로 구성된다. 하지만, 본 논문에서 다루는 분산 공유메모리 환경의 다단계 버스망은 (그림 1)과 같이 프로세서와 메모리 모듈이 하나의 쌍을 이루고 있는 노드들을 연결하게 된다. 다단계 버스망의 내부 스위치는 버스이고 다단계 버스망의 양 끝단에 연결된 같은 번호의 노드는 같은 노드를 나타낸다.

다단계 버스망에서는 연속된 두 단계중 어느 한 단계에 위치한 두 개의 버스들 사이에 버디관계가 성립한다. 버디관계란 어느 단계에 위치한 두개의 버스가 다음 단계에 위치한 두개의 버스에 공통으로 연결되는 관계를 말한다[6]. (그림 2)는 버디관계를 그림으로 표현하고 있다. (그림 2)에서 버스 a와 버스 b는 전방향으로 다음 단계에 위치한 버스 d, e와 공통으로 연결되어 버디관계가 성립하고, 마찬가지로 버스 f와 버스 g는 후방향으로 다음 단계에 위치한 버스 c, d와 공통으로 연결되어 버디관계가 성립한다. 버디관계는 잉여경로를 형성하는 조건이 되므로 3장에서 설명할 동적 라우팅에 중요하게 활용된다.



(그림 2) 버디관계

2.2 기존의 라우팅 알고리즘

2.2.1 기본 라우팅 방법

일반 다단계 연결망에서의 라우팅은 요청하는 프로세서에 할당된 이진수 표현 레이블과 대상 메모리 모듈의 이진수 표현 레이블을 이용한다. 데이터 교환은 스위치 내에서는 Exchange 연산을 수행하고 연속된 단계들 사이에서는 Shuffle/ReverseShuffle 연산을 수행함으로써 이루어진다. 이들 연산은 프로세서나 대상 메모리 모듈의 이진수 표현 레이블에 대응되는 스위치 포트의 이진수 표현 레이블을 이용한다. 기본 라우팅 방법으로는 전방 라우팅과 후방 라우팅 방법이 있다. 전방 라우팅 방법은 일련의 Exchange 연산과 Shuffle 연산으로 이루어지고 후방 라우팅 방법은 일련의 Exchange 연산과 ReverseShuffle 연산으로 이루어진다[11].

그러나, 다단계 버스망에서는 전방/후방 라우팅 방법 외에도 U자형 선회 라우팅 방법이 가능하다. U자형 선회 라우팅 방법이란 망의 중간 단계에서 경로 방향을 역으로 바꾸는 라우팅 방법을 말한다. 이 방법은 요청하는 프로세서로부터 대상 메모리 모듈로의 경로와 대상 메모리 모듈로부터 요청한 프로세서로의 경로가 임의의 중간 단계에서 동일한 스위치를 경유하는 특성을 이용한다. U자형 선회 라우팅 방법에는 전방 U자형 선회 라우팅과 후방 U자형

선회 라우팅, 두 가지 종류가 있다. 예를 들어, 전방 U자형 선회 라우팅 방법인 경우 단계 t에서 선회를 한다고 가정하면 단계 0부터 단계 t까지는 전방 라우팅을 하고 단계 t에서 선회한 후 다시 단계 0까지는 후방 라우팅을 한다[7].

U자형 선회 라우팅 방법을 사용함으로써 전방/후방 라우팅 방법을 사용했을 때보다 경로상의 스위치 수를 줄일 수 있다. 또한, U자형 선회 라우팅 방법에서 버디 특성을 이용하면 참조 요청을 하는 프로세서와 대상 메모리 모듈의 쌍이 같고 동일한 라우팅 방법이 선택되었다해도 데이터를 다른 경로로 라우팅 시킬 수 있어 스위치의 트래픽을 분산시킬 수 있다. 이러한 사실들은 다단계 버스망이 공유매체인 버스를 스위치로 사용한다는 제약을 고려할 때, 메모리 참조에 필요한 평균 응답시간을 줄이는데 중요한 개선사항이 된다.

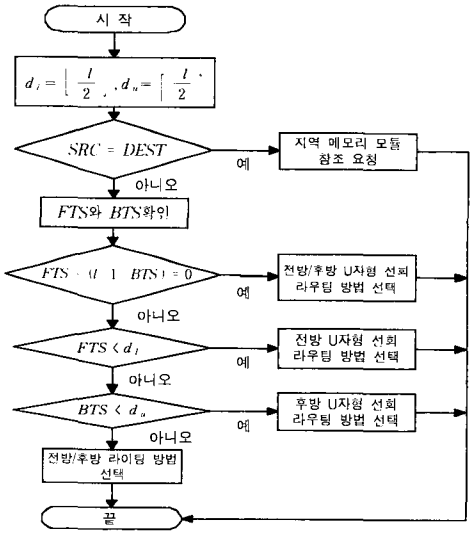
2.2.2 최적 경로 결정

최적 경로 결정이란 바로 앞 절에서 설명한 네 가지 라우팅 방법중 경로상의 스위치수가 가장 적은 라우팅 방법을 선택하는 과정을 말한다. (그림 3)은 Bhuyan[7]이 제안한 방법을 나타내고 있다. 이 그림에서 사용된 기호는 <표 1>과 같다.

〈표 1〉 Bhuyan의 최적 경로 결정 방법에서 사용된 기호

l : 다단계 버스망의 단계수 SRC : 요청한 프로세서 $DEST$: 대상 메모리 모듈을 가진 프로세서 FTS : 전방 U자형 선회 라우팅시 선회가 가능한 단계 ($0 \le FTS \le l-1$) BTS : 후방 U자형 선회 라우팅시 선회가 가능한 단계 ($0 \le BTS \le l-1$)

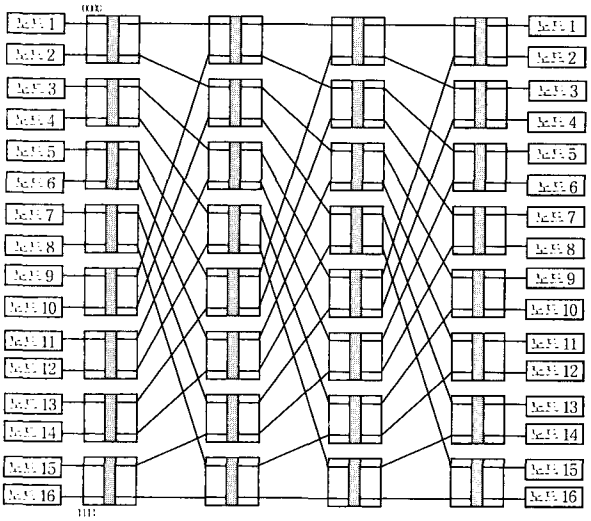
이 최적 경로 결정 방식에서는 (그림 3)의 각 조건문에서 알 수 있듯이 요청하는 프로세서와 대상 메모리 모듈을 가진 프로세서가 같은 경우 스위치를 경유하지 않고 프로세서의 지역 메모리 모듈을 참조한다. U자형 선회가 첫 번째 단계



(그림 3) 최적 경로 결정 방법

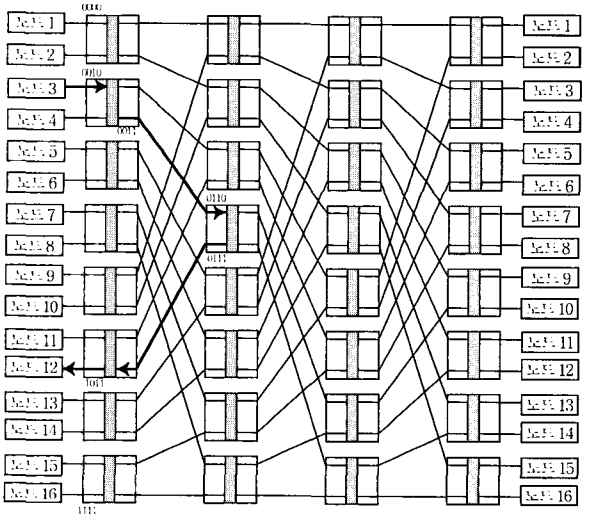
또는 마지막 단계에서 가능한 경우 전방/후방 U자형 선회 라우팅 방법을 선택한다. 정중앙 단계보다 이전 단계에서 U자형 선회가 가능한 경우 전방 U자형 선회 라우팅 방법을 선택하고, 정중앙 단계보다 이후 단계에서 U자형 선회가 가능한 경우 후방 U자형 선회 라우팅 방법을 선택한다. 전방/후방 라우팅 방법을 선택했을 때와 경로상의 스위치수가 같거나 더 많은 경우에는 전방/후방 라우팅을 선택한다. 여기에는 다음절에서 지적할 몇 가지 중요한 문제점이 있다.

2.2.3 적용예



(그림 4) 간단한 다단계 연결망 구성예

다음은 (그림 4)의 다단계 연결망 구성예를 사용하여 기존의 라우팅 알고리즘과 최적 경로 결정 방법을 적용한 예를 나타낸다. 노드 3에서 노드 12로 메모리 액세스 요청을 한다고 가정한다. 노드1은 요청을 송신하기 전에 최적 경로 결정 방법을 통해 사용할 라우팅 방법을 결정한다. FTS값은 1이고 d_u값은 2이므로, $FTS < d_u$ 조건을 만족하여 전방 U



(그림 5) 노드 3에서 노드 12로의 전방 U자형 선회 라우팅 방법

자형 선회 라우팅 방법을 선택하게 된다. (그림 5)는 노드 3에서 노드 12로 전방 U자형 선회 라우팅 방법을 사용했을 때의 경로를 나타낸다.

2.2.4 문제점

2.1절에서 말한 버디관계를 가지는 다단계 버스망은 U자형 선회 라우팅시 연속된 단계들 사이에 항상 두 개 이상의 경로를 갖게 된다. 하지만, 기존의 라우팅 방법은 동적으로 경로를 변경할 수 없다. 그래서, 스위치의 트래픽을 분산시킬 수 없을 뿐만 아니라 유휴 상태에 있거나 사용빈도가 낮은 스위치를 활용하지도 못한다.

또한, 기존의 최적 경로 결정 방식에서는 전방/후방 라우팅시 경로상의 스위치 수와 전방/후방 U자형 선회 라우팅시 경로상의 스위치 수가 동일한 경우 전방/후방 라우팅 방법을 선택한다. 즉, 경로 상의 스위치 수 만을 따지고 스위치의 트래픽은 고려하지 못하고 있다.

3. 트래픽을 고려한 동적 라우팅 알고리즘

3.1 동적 라우팅 방법

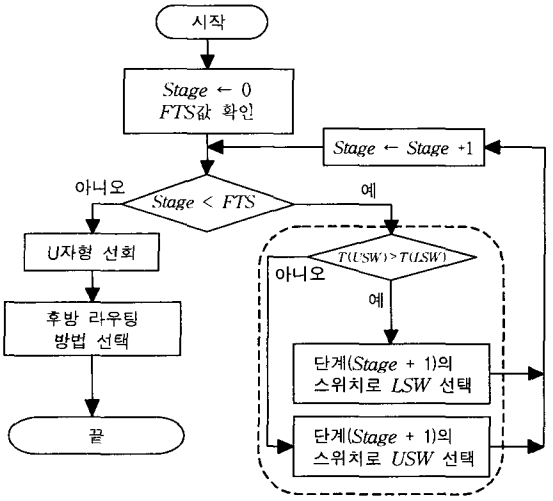
기존에 제안된 전방/후방 라우팅 방법이나 전방/후방 U자형 선회 라우팅 방법은 데이터 전송전에 경로가 결정되는 정적 라우팅이다. 동적 라우팅이란 경로 결정 시기면에서 정적 라우팅과 반대되는 개념이다. 즉, 조건에 따라 경로를 설정함으로써 경로상의 특정 스위치나 링크에 이상이 있는 경우 다른 경로를 선택한다. 이러한 동적 또는 적응적 라우팅 방법은 통신상의 충분한 대역폭 확보와 병목현상 회피를 위한 목적으로 연구가 이루어져 왔다[2].

본 논문에서 제안하는 동적 라우팅 방법은 기존의 전방/후방 U자형 선회 라우팅 방법이 스위치의 트래픽을 고려하도록 하고 있다. (그림 6)은 동적 전방 U자형 선회 라우팅 방법을 나타내고 있다. <표 1>의 기호 이외에 여기서 추가로 사용된 기호는 <표 2>에 보인 것과 같다.

〈표 2〉 동적 전방 U자형 선회 라우팅 방법에서 사용된 기호

$T(SW)$: 해당 스위치의 트래픽 정도를 나타내는 수치, sw 값은 USW 또는 LSW
S_i : i번째 비트, $(0 \le i \le l-1)$
$S_0S_1S_2 \wedge S_{l-1}$: 프로세서와 스위치 포트의 이진수 표현 레이블
CS : 경로상의 현재 단계, $(0 \le CS \le l-1)$
USW : 단계 i의 스위치에 연결된 단계 $(i+1)$의 두 스위치중 $S_i=0$인 경우에 연결된 스위치,
LSW : 단계 i의 스위치에 연결된 단계 $(i+1)$의 두 스위치중 $S_i=1$인 경우에 연결된 스위치

이 라우팅 방법은 임의의 단계에 위치한 스위치에 연결된 스위치들중 스위치의 트래픽 정도에 따라 경로를 선택하는 과정을 기본으로 수행한다. (그림 6)에서 경로 선택과정은 U자형 선회가 가능한 단계까지 전방 라우팅 방법에서와 같다. 하지만, 단계($CS+1$)에 위치한 스위치의 $T(SW)$ 값을 비교하여 스위치를 선택하는 과정이 스위치 내에서 수행되는 연산에 포함된다. 스위치 내에서 수행되는 연산은 (그림 6)에서 점선으로 표시하고 있다. 이 연산의 결과로서 단계 (CS)에서의 스위치 포트의 이진수 표현 레이블은 단계 ($CS+1$)에서 바뀌게 된다. 이 변환 과정은 다음과 같다. 단계($CS+1$)의 스위치로 LSW를 선택하는 경우, 단계 (CS)에서의 스위치 포트에 대한 S_{l-1}값이 0이면 레이블은 $S_0S_1 \wedge S_{l-2}S_{l-1}$에서 $S_1S_2 \wedge \overline{S_{l-1}}S_0$로 바뀌고 S_{l-1}값이 1이면 레이블은 $S_0S_1 \wedge S_{l-2}S_{l-1}$에서 $S_1S_2 \wedge S_{l-1}S_0$로 바뀐다. 같은 방식으로 단계($CS+1$)의 스위치로 USW를 선택하는 경우, 단계 (CS)에서의 스위치 포트에 대한 S_{l-1}값이 0이면 레이블은 $S_0S_1 \wedge S_{l-2}S_{l-1}$에서 $S_1S_2 \wedge S_{l-1}S_0$로 바뀌고 S_{l-1}값이 1이면 레이블은 $S_0S_1 \wedge S_{l-2}S_{l-1}$에서 $S_1S_2 \wedge \overline{S_{l-1}}S_0$로 바뀐다. 선회가 가능한 단계에서는 그 단계에서 U자형 선회를 한 후, 후방(전방) 라우팅 방법을 따른다.

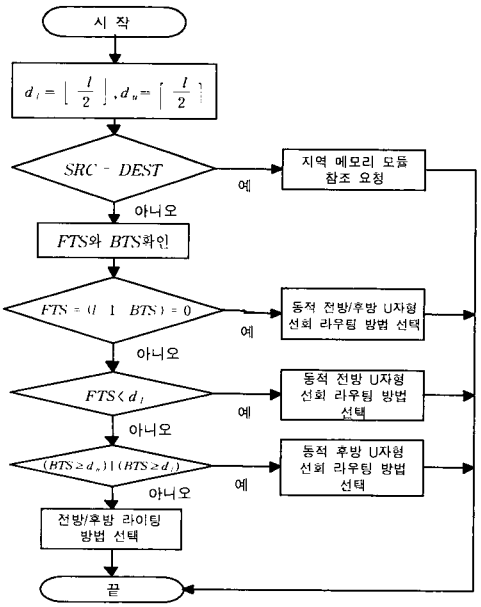


(그림 6) 동적 전방 U자형 선회 라우팅 방법

3.2 동적 라우팅을 위한 최적 경로 결정

(그림 7)은 개선된 최적 경로 결정 방법을 나타낸다. 이 방법은 기존의 최적 경로 결정 방법과 비교해 제안하는 동적 라우팅 방법을 선택할 확률이 높다라는 점에서 다르다. 즉, 라우팅시 경유하는 스위치 수를 최소로 하는 라우팅 방법을 선택하는 것을 목적으로 하는 최적 경로 결정 방법을 개선시켜 스위치를 최소로 경유하면서 동적 라우팅 방법의 활용도를 높였다. 특히, (그림 7)의 조건문 ($FTS \le d_i$)과 ($BTS = d_i$)는 경로상의 스위치 수 면에서 기존의 라우팅 방법과 동적 라우팅 방법이 동일한 경우에 동적 라우팅 방법의 활용도를 높이기 위해 사용된다. 이때, 동일한 조건 하에서 FTS/BTS 단계에 도달했을 때 U자형 선회를 하지 않는다면 그 단계부터 전방/후방 라우팅을 할 수도 있다. 이것은 FTS/BTS

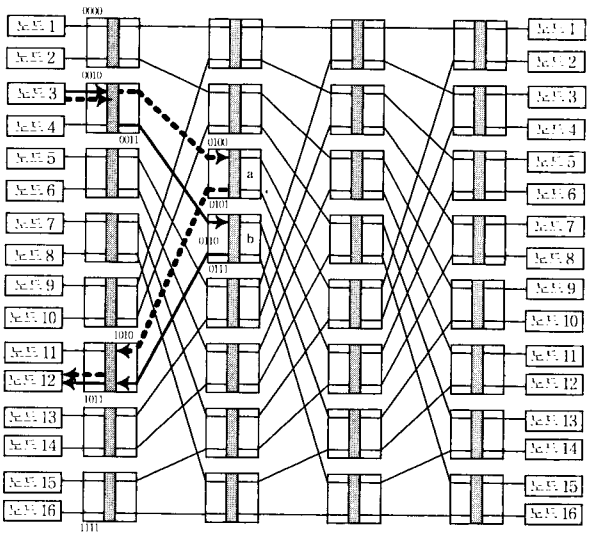
단계에 도달할 때까지 동적 라우팅시의 경로가 전방/후방 라우팅시의 경로와 일치한다는 조건을 만족해야 한다. 하지만, 이 조건에 대한 확률은 선회할 단계가 t인 경우, 1/2'가 된다. 즉, t가 증가함에 따라 확률은 지수분포로 줄어든다. 그러므로, 본 논문에서는 동적 라우팅 방법과 전방/후방 라우팅 방법이 동일한 스위치 수를 경유하는 경우, 항상 동적 라우팅 방법을 선택하도록 한다.



(그림 7) 개선된 최적 경로 결정 방법

3.3 적용 예

다음은 (그림 4)의 다단계 연결망 구성예를 사용하여 동적 라우팅 알고리즘과 동적 라우팅을 위한 최적 경로 결정 방법을 적용한 예를 나타낸다. 노드 3에서 노드 12로 메모리



(그림 8) 노드 3에서 노드 12로의 동적 전방 U자형 선회 라우팅 방법

액세스 요청을 한다고 가정한다. 노드 1은 요청을 송신하기 전에 최적 경로 결정 방법을 통해 사용할 라우팅 방법을 결정한다. FTS값은 1이고 d_u값은 2이므로, $FTS < d_u$ 조건을 만족하여 동적 전방 U자형 선회 라우팅 방법을 선택하게 된다. 이 방법에 의한 경로들은 기존 전방 U자형 선회 라우팅 경로뿐만 아니라 버디특성에 의한 잉여경로들도 포함한다. (그림 8)은 노드 3에서 노드 12로 동적 전방 U자형 선회 라우팅 방법을 사용했을 때의 경로들을 나타낸다. 이 그림에서 실선으로 연결된 경로는 기존 전방 U자형 선회 라우팅 경로를 나타내고 점선으로 연결된 경로는 버디특성에 의한 잉여경로를 나타낸다.

4. 시뮬레이션

본 논문에서 제안하는 동적 라우팅 방법의 성능을 분석하기 위해서 분산 공유메모리 환경의 다단계 버스망에 적용하여 프로세서의 수와 스위치의 크기를 변화시켜 가면서 시뮬레이션 하였다. 시뮬레이션은 주메모리가 32M인 266MHz 펜티엄II 컴퓨터에서 Simscript를 시뮬레이션 언어로 사용하여 실행하였다.

4.1 시뮬레이션 모델

시뮬레이션은 분산 공유메모리 환경의 다단계 버스망에서 서로 다른 라우팅 방법을 사용하는 시스템을 비교한다. 시스템은 사용하는 라우팅 방법에 따라 전방/후방 라우팅 방법을 사용하는 시스템(FB)[7], FB 시스템에 Bhuyan의 전방/후방 U자형 선회 라우팅 방법을 추가한 시스템(FBU)[5], 그리고 FBU에 본 논문에서 제안하는 동적 전방/후방 U자형 선회 라우팅 방법을 추가한 시스템(FBDU)으로 구분된다. 또한, 각 시스템은 평균 응답시간, 스위치의 평균 대기 패킷수, 그리고 메모리 모듈의 평균 대기패킷수를 비교척도로 비교된다.

동적 전방/후방 U자형 선회 라우팅 방법에서 사용되는 스위치의 트래픽 정보는 현재 스위치의 서비스를 받고 있는 작업을 제외한 스위치의 대기 큐에서 기다리는 작업수로 정의한다. 스위치는 주기적인 제어 패킷에 의해 링크로 연결된 스위치의 트래픽 정보를 정확히 알고 있다는 것을 가정한다.

시뮬레이션은 프로세서수(메모리 모듈의 수와 동일)와 스위치 크기를 파라메터로 사용한다. 시뮬레이션에서 사용한 프로세서수는 8, 16, 32, 그리고 64개이고 단일 버스인 스위치는 2×2, 4×4, 그리고 8×8의 스위치 크기를 갖는다.

시뮬레이션에서 스위치 서비스 시간 대 메모리 서비스 시간의 비율은 1 : 8로 하였고, 스위치와 메모리 모듈의 큐는 무한 큐라고 가정하였다. 또한 프로세서의 요청 발생 확률과 대상 메모리 모듈의 선택 확률은 지수 분포를 따른다고 가정하였다. 이러한 시뮬레이션 조건은 Bhuyan의 시뮬레이션[7], Chen의 시뮬레이션[12] 등에서 적용한 것과 같다. 그리고 서비스 시간은 스위치나 메모리 모듈에 요구가 입력된

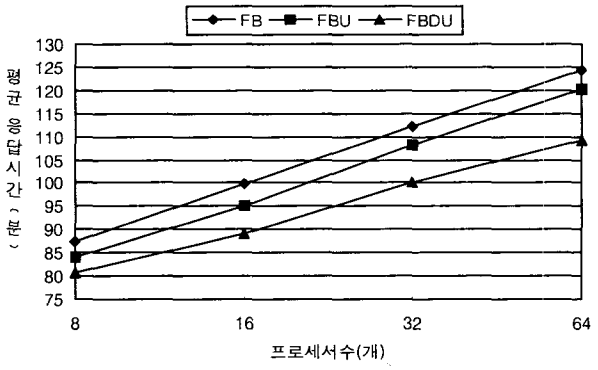
후 결과가 출력되기까지 걸리는 시간으로 정의한다.

프로세서는 1000번의 메모리 참조 요청을 발생시킨다. 이 시뮬레이션에서 얻어진 각 메모리 참조 요청에 대한 응답시간 값으로부터 평균 응답시간을 계산한다. 또한, 스위치의 평균 대기패킷수와 메모리 모듈의 평균 대기패킷수를 각각 스위치의 평균 대기패킷수와 메모리 모듈의 평균 대기패킷수로부터 계산한다. 이때, 평균값 계산시, 시뮬레이션의 해당 결과값중 최대값과 최소값을 제외한다.

4.2 시뮬레이션 결과 및 고찰

먼저 프로세서와 메모리 모듈의 수가 증가함에 따라 각 시스템이 나타내는 성능을 비교해 보았다. 각 시스템은 2×2 스위치를 사용한다.

(그림 9)는 각 시스템의 평균 응답시간을 비교한다. 각 시스템의 평균 응답시간을 비교함으로써 다음과 같은 사실들을 알 수 있다. 첫 번째로 FBDU 시스템의 평균 응답시간이 항상 가장 짧게 나타난다. 두 번째로 프로세서수가 증가함에 따라 FB 시스템과 FBU 시스템의 평균 응답시간 차는 일정하지만, FBDU 시스템과 다른 두 시스템의 평균 응답시간 차는 계속해서 증가함을 볼 수 있다. 세 번째로 FBU 시스템과 FBDU 시스템의 경우 프로세서 수에 의해 짝수개의 단계를 갖는 시스템에서 홀수개의 단계를 갖는 시스템으로 확장될 때, 평균 응답시간의 증가량이 크다는 것을 알 수 있다. 또한, 망의 단계수 증가에 따른 FBU 시스템과 FBDU 시스템의 평균 응답시간 증가량을 비교해 보면 FBDU 시스템에서의 증가량이 작음을 알 수 있다.

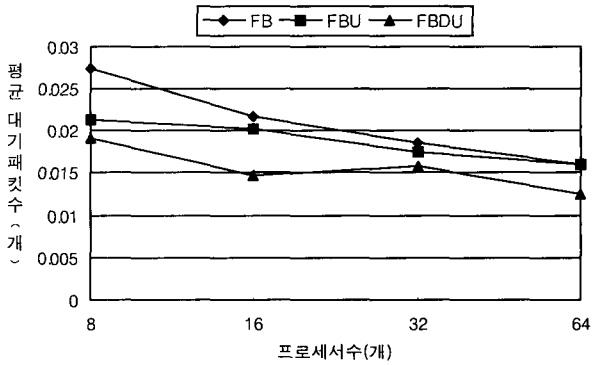


(그림 9) 프로세서수 대 평균 응답시간

이때, 홀수개의 단계을 갖는 시스템으로 확장될 때. 평균 응답시간의 증가량 면에서 차이가 생기는 현상은 망의 정중앙 단계에 위치한 스위치에 대한 서비스 요구 트래픽에 차이가 있기 때문이다. 즉, 짝수개의 단계를 갖는 시스템에서의 전방 U자형 선회 라우팅과 후방 U자형 선회 라우팅은 각각 망의 서로 다른 절반 영역에 위치한 스위치를 배타적으로 이용하여 서로 영향을 주지 않기 때문이다. 또한, 망의 단계수 증가에 따른 평균 응답시간 증가량 면에서, FBU 시스템에서의 증가량이 FBDU 시스템에서의 증가량보다 작은 것은 스위치의 트래픽을 고려함으로써 정중앙 단계에 위치한 스위치의 트래픽이 분산되기 때문이다. 스위치의 트래픽이 분산된다는 사실은 (그림 10)에서 보여주는 스위치의 평균 대기패킷수 비교에서 확인할 수 있다.

(그림 10)은 스위치의 평균 대기패킷수 면에서 각 시스템을 비교하고 있다.

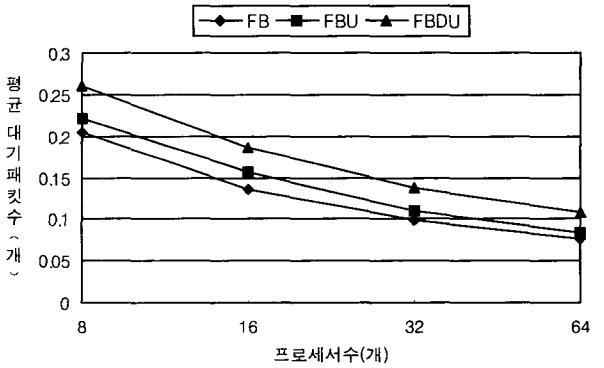
(그림 10)에서는 FBDU 시스템이 다른 두 시스템보다 스위치의 평균 대기패킷수가 작다는 것을 알 수 있다. 다시 말해서, 스위치의 트래픽이 분산됐음을 보여주는 결과다. 하지만, FBDU 시스템의 평균 대기패킷수는 프로세서수 증가에 따라 항상 감소하지는 않는 결과를 보여준다. 이것은 평균 응답시간에 대한 시뮬레이션 결과에서와 마찬가지로 망의 정중앙 단계에 위치한 스위치에 대한 서비스 요구 트래픽에 차이가 있기 때문이다.



(그림 10) 프로세서수 대 스위치의 평균 대기패킷수

(그림 11)은 메모리 모듈의 평균 대기패킷수 면에서 각 시스템을 비교하고 있다.

(그림 11)에서는 메모리 모듈의 평균 대기패킷수가 FBDU 시스템에서 항상 많다는 것을 알 수 있다. 이런 결과는 단축된 평균 응답시간이 메모리 모듈에 도달하는 시간의 단축으로부터 가능하다는 것으로 해석된다. 또한, 그 단축된 시간이 FBDU 시스템에서 가장 크다고 할 수 있다.

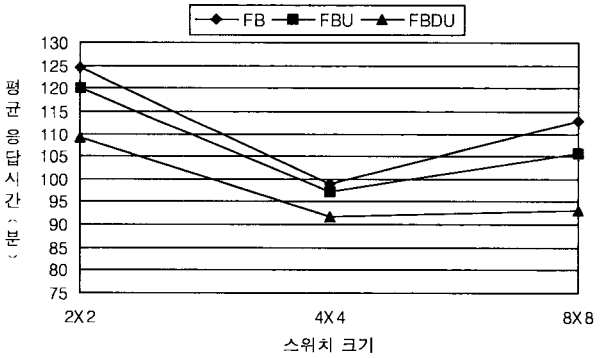


(그림 11) 프로세서수 대 메모리 모듈의 평균 대기패킷수

또한, 메모리 모듈의 평균 대기패킷수의 감소량 면에서 FBU 시스템과 FBDU 시스템을 비교해 보면, FBDU 시스템에서의 감소량이 더 크다는 것을 알 수 있다. 이것은 망의 크기가 커짐에 따라 메모리 모듈의 평균 대기패킷수가 단축된 평균 응답시간에 미치는 영향이 작아짐을 보여주는 것이다.

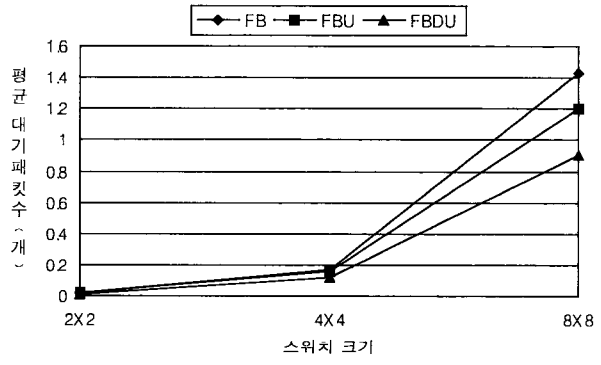
다음은 스위치의 크기가 증가함에 따라 각 시스템이 나타내는 성능을 비교한다. 각 시스템은 64개의 프로세서를 사용한다.

(그림 12)는 각 시스템의 평균 응답시간을 비교한다. 스위치의 크기를 증가시키는 경우, FBDU 시스템이 항상 짧은 평균 응답시간을 보이고 있다. 특히, 스위치 크기가 다른 각 FBDU 시스템중에서도 스위치의 크기가 4×4인 경우에 가장 짧은 평균 응답시간이 걸린다는 것을 알 수 있다. 또한, 모든 시스템에서 스위치의 크기가 4×4인 경우에 가장 짧은 평균 응답시간을 나타냈다가 다시 8×8의 스위치를 사용했을 때 다시 평균 응답시간이 길어진 것을 볼 수 있다. 이것은 4×4 스위치를 사용하는 경우 망이 세 개의 단계로 구성되어 정중앙 단계에서 U자형 선회 라우팅을 사용할 확률이 높기 때문이다. 하지만, 8×8 스위치를 사용하는 경우, 망은 두 개의 단계로 구성되어 U자형 선회 라우팅을 사용할 확률이 낮다.



(그림 12) 스위치 크기 대 평균 응답시간

(그림 13)은 스위치의 평균 대기패킷수 면에서 각 시스템을 비교하고 있다.

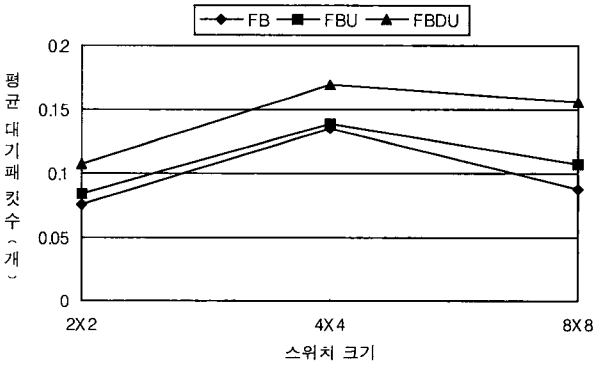


(그림 13) 스위치 크기 대 스위치의 평균 대기패킷수

(그림 13)에서는 FBDU 시스템이 다른 두 시스템보다 스위치의 평균 대기패킷수가 작다는 것을 알 수 있다. 다시 말해서, 스위치의 트래픽이 분산됐음을 보여주는 결과다. 하지만, 스위치의 크기가 증가함에 따라 평균 대기패킷수는 약 10배 가까이 증가한다. 이것은 스위치 크기의 증가에 따라 스위치의 서비스를 받기 위해 경쟁하는 참조 요청수가 증가하기 때문이다.

(그림 14)는 메모리 모듈의 평균 대기패킷수 면에서 각 시스템을 비교하고 있다.

메모리 모듈의 평균 대기패킷수는 4×4 스위치를 사용했을 때 가장 많고, 8×8 스위치를 사용했을 때 다시 감소하게 된다. 이것은 (그림 12)와 (그림 13)의 결과에서도 알 수 있듯이 4×4 스위치를 사용하는 시스템에서 8×8 스위치를 사용하는 시스템으로의 확장시 평균 응답시간의 증가와 함께 스위치의 평균 대기패킷수가 큰 폭으로 증가하기 때문이다. 즉, 8×8 스위치의 평균 대기패킷수가 증가함에 따라 상대적으로 메모리 모듈의 평균 대기패킷수는 줄어들게 된다.



(그림 14) 스위치 크기 대 메모리 모듈의 평균 대기패킷수

스위치 크기를 변화시켜본 시뮬레이션에서는 다른 크기의 스위치들을 사용하는 시스템들에 비해 상대적으로 4×4 스위치를 사용한 시스템의 성능이 우수한 것으로 나타났다. 이것은 64개로 고정된 프로세서수를 사용함으로써 사용하는 스위치 크기에 따라 망의 단계수가 결정되기 때문이다. 즉, 4×4 스위치를 사용한 시스템은 2×2 스위치를 사용한 시스템에 비해 스위치의 평균 대기패킷수는 증가하지만, 홀수개의 단계로 구성되어 U자형 선회 라우팅 방법을 활용할 확률이 높다.

5. 결 론

본 논문에서는 분산 공유메모리 환경의 다단계 버스망에서 스위치의 트래픽을 고려하는 동적 라우팅 방법을 제안하였다.

패킷이 이동하는 동안에 경로를 선택함으로써 기존의 정적 라우팅 방법에서는 사용하지 못하는 잉여 경로를 활용

하고 있다. 최적 경로 결정을 함에 있어 최대한 잉여 경로를 활용할 수 있도록 최적경로 결정 방법을 개선하였다. 시뮬레이션 결과에서도 알 수 있듯이 스위치의 트래픽을 분산시킴으로써 평균 대기패킷수를 줄여 스위치를 경유하는데 소요되는 시간을 줄일 수 있다. 결과적으로 시스템 전체의 평균 응답시간을 줄일 수 있다.

따라서, 본 논문에서 제안한 동적 라우팅 방법을 사용할 경우, 병렬처리 시스템용 응용이 결과를 얻는데 걸리는 시간을 줄일 수 있고 하드웨어로 제작시 좀더 작은 유한큐를 내장하는 스위치를 사용할 수 있어 제작비용을 감소시킬 수 있으리라 기대된다.

참 고 문 헌

[1] T. Fen, "A Survey of Interconnection Networks," IEEE Computer, Dec., 1981.

[2] P. T. Gaughan and S. Yalamanchili, "Adaptive Routing Protocols for Hypercube Interconnection Networks," IEEE Computer, Vol.26, No.5, pp.12-23, May, 1993.

[3] S. M. Mahmud, "Performance Analysis Of Multilevel Bus Networks For Hierarchical Multiprocessors," IEEE Trans. on Computers, Vol.43, No.7, pp.789-805, Jul., 1994.

[4] L. K. John and Y. Liu, "Performance Model for a Prioritized Multiple-Bus Multiprocessor System," IEEE Trans. on Computers, Vol.45, No.5, pp.580-588, May, 1996.

[5] A. K. Nanda, "Design and Analysis of Cache Coherent Multistage Interconnection Networks," IEEE Trans. on Computers, Vol.42, No.4, pp.458-470, Apr., 1993.

[6] D. P. Agrawal, "Graph Theoretical Analysis and Design of Multistage Interconnection Networks," IEEE Trans. on Computers, Vol.C-32, No.7, pp.637-648, Jul., 1983.

[7] L. N. Bhuyan, R. R. Iyer, T. Askar, A. K. Nanda, and M. Kumar, "Performance of Multistage Bus Networks for a Distributed Shared Memory Multiprocessor," IEEE Trans. on Parallel and Distributed Systems, Vol.8, No.1, pp.82-95, Jan., 1997.

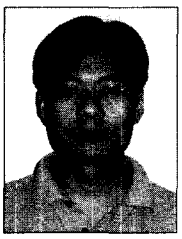
[8] L. N. Bhuyan, A. K. Nanda, and T. Askar, "Performance and Reliability of the Multistage Bus Networks," Proc. of the Int'l. Conf. on Parallel Processing, pp.26-33, Aug., 1994.

[9] J. Protic, M. Tomasevic, and V. Milutinovic, "Distributed Shared Memory : Concepts and Systems," IEEE Parallel & Distributed Technology, Vol.4, No.2, pp.63-79, Summer, 1996.

[10] M. stumm and S. Zhou, "Algorithms Implementing Distributed Shared Memory," IEEE Computer, Vol.23, No.5, pp. 54-64, May, 1990.

[11] G. S. Almasi and A. Gottlieb, "*Highly Parallel Computing*," 2nd Ed., Benjamin/Cummings Pub. Co., 1994.

[12] S. Chen, J. A. Stankovic, F. Kurose, and D. Towsley, "Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time System," Journal of Real-Time Systems, Vol.3, pp.307-336, Sept., 1991.



홍 강 운

e-mail : gwhong@etri.re.kr

1996년 한양대학교 전자계산학과 졸업 (학사)

1998년 한양대학교 전자계산학과(공학석사)

1998년~2000년 (주)콤텍시스템 기술연구소 주임연구원

2000년~현재 한국전자통신연구원 네트워크연구소 연구원

관심분야 : 병렬 처리, 고속 라우터, 메트로 이더넷, RPR 등

전 창 호

e-mail : chjeon@paral.hanyang.ac.kr

1977년 한양대학교 전자공학과 졸업(학사)

1982년 Cornell University 전기및컴퓨터공학과(공학석사)

1986년 Cornell University 전기및컴퓨터공학과(공학박사)

1977년~1979년 한국전자통신연구원 연구원

1986년~1989년 성균관대학교 전기공학과 조교수

1989년~현재 한양대학교 전자컴퓨터공학부 교수

1997년~1998년 한국정보과학회 학회지 편집위원장

관심분야 : 컴퓨터구조, 병렬처리, 성능분석 등