N-차원 메쉬 네트워크에서의 부분적 적응성을 이용한 Deadlock-Free 결함포용 라우팅 기법

문 대 근† · 김 학 배††

요 약

열악한 환경에서 동작되는 멀티컴퓨터는 요소결함(component faults)이 존재하는 상황에서도 정상적 동작을 보장할 수 있도록 설계되어야 한다. 이를 위한 하나의 방법으로 결함포용 라우팅(fault-tolerant routing) 기법이 고려될 수 있다. 본 논문에서는 n-차원 메쉬 네트워크를 기본 토폴로지로 선택하여 이러한 네트워크의 임의의 장소에서 링크결함이 발생했을 경우에도 메시지들을 목적지로 전달시킬 수 있는 결함포용 라우팅 알고리즘을 제안한다. 제안된 결함포용 라우팅 알고리즘은 기본적으로 WH(WormHole) 라우팅 방식을 채택하며, deadlock-free를 실현하기 위하여 한 개의 물리적 채널을 공유하는 복수 개의 가상채널들(virtual channels)을 사용한다. 결론적으로 컴퓨터 시뮬레이션을 통해 제안된 알고리즘이 널리 알려진 X-Y 라우팅 알고리즘보다 향상된 성능을 갖는다는 사실을 입증한다.

A Deadlock-Free Fault-Tolerant Routing Method Using Partial-Adaptiveness in a N-Dimensional Meshed Network

Dae-Keun Moon† · Hag-Bae Kim††

ABSTRACT

The multicomputers operated in harsh environments should be designed to guarantee normal operations in the presence of the component faults. One solution for this is a fault-tolerant routing. In the paper, we consider n-dimensional meshed network for the basic topology and propose a simple fault-tolerant routing algorithm that can transfer messages to their destination as desired in the presence of some component faults. The built algorithm basically adopts a WormHole(WH) routing method and uses the virtual channels sharing a physical channel for deadlock-freedom. Consequently, we show that the suggested algorithm has a higher performance than the X-Y routing algorithm through simulation results.

1. 서 론

근래 널리 사용되고 있는 병렬구조의 분산메모리 멀티컴퓨터는 프로세서, 지역메모리, 라우터 등을 요소로 갖는 많은 노드들로 구성되어 있다. 각 노드의 컴퓨터들은 프로그램을 수행하는 동시에 네트워크내에 있는 노드와 노드사이에 데이터를 교환한다. 따라서 네트워크의 전체적인 성능(performance)은 노드를 이루는 컴퓨터의 성능과 네트워크사이의 연결수행능력으로 결정되고, 이것은 주로 네트워크 토폴로지와 라우팅 전략에 의해서 영향을 받는다. 따라서 라우팅 전략은 네트워크의 성능을 좌우하는 가장 중요한 요소중의 하

※ 본 논문은 한국과학재단의 1998~2000년도 핵심전문연구로서 연구지원되었음.
† 준 회 원 : 연세대학교 대학원 기계전자공학부
†† 정 회 원 : 연세대학교 기계전자공학부 전기전공 교수
논문접수 : 1998년 7월 2일, 심사완료 : 1999년 1월 22일

나이다. 특히 네트워크상에 요소결함(component faults)이 발생할 경우에 안정된 라우팅에 대한 중요성은 더욱 커진다.

WH(WormHole) 라우팅[3,5]은 메시지를 패킷으로, 패킷을 다시 플릿(flit : flow control digit)이나 워드로 나누고 이 플릿들을 네트워크상에서 파이프라인의 형태로 이동시키는 방식이다. 그러므로 WH는 각 라우터의 버퍼크기를 줄일 수 있고, 중간노드에서의 작은 지연시간 때문에 시간지연이 출발지와 목적지 사이의 거리에 별로 영향을 받지 않는다. 그러나, WH는 네트워크상에 정보량이 증가함에 따라 많은 채널들이 메시지에 의해서 점유되는 결점 때문에 deadlock에 민감하다. Deadlock은 한마디로 메시지가 네트워크상에서 더 이상 진행해야할 곳을 찾을 수 없는 경우로 정의되며, WH 방식에서 자주 나타난다. 이러한 deadlock을 없애기 위한 방법으로는 가상채널(virtual channels)[2]이 주로 사용된다. 가상채널은 물리적 채널을 여러 개의 가상채널로 나누는 방식으로 각각의 가상채널은 자신의 큐(queue)를 가지고 있다. 그러므로, 버퍼들 사이의 메시지 전달을 제한함으로써 deadlock을 피할 수 있다. 한편 가상채널은 라우터의 overhead가 매우 큰 방식이므로 필요한 가상채널의 수를 최소한으로 줄이는 것이 효율적이다.

링크(또는 노드)에 요소결함이 발생하면 메시지는 그 링크(또는 노드)를 통하여 전달될 수 없고 이러한 문제를 해결하기 위한 하나의 방법으로 결함포용 라우팅(fault-tolerant routing) 기법이 고려될 수 있다. 결함포용 라우팅이란 메시지들이 요소결함에 때문에 그들의 목적지에 도달할 수 없을 경우에 요소결함들을 피해서 이들을 전달시키는 라우팅기법으로 정의된다. 최근까지 진행된 결함포용 라우팅에 대한 연구에 대해 살펴보면 다음과 같다. Linder와 Harden은 [3]에서 가상채널의 개념을 가상 네트워크의 개념으로 확장시키면서 결함포용 라우팅을 위해서 각 차원당 가상채널의 수를 두배로 증가시켰다. 따라서 이 방법은 너무 많은 가상채널을 필요로 한다는 단점을 지닌다. [4]에서 Glass와 Ni는 n-차원 메쉬구조에서의 결함포용 라우팅 알고리즘을 제시하였다. 하지만 이 알고리즘은 2D 메쉬의 경우에 한 개의 결함만을 포용할 수 있다. [5]에서 Boppana와 Chalasani는 메쉬 네트워크에서 블럭화된 요소결함만을 포용할 수 있는 결함포용 라우팅을 제안하였다. 한편, Boura와 Das가 [6]에서 제시한 방법은 Boppana의 방법과 유사하지만 블럭형태가 아닌 결함도 포용할 수 있다. 그러나 정상적인 노드가 제시된 규칙에 의해서 비활성화될 수도 있다.

본 논문에서는 링크결함만을 고려하며, 각 노드는 자신과 연결된 링크의 요소결함정보만을 인식하는 지역메모리를 가진다고 가정한다. 또한, deadlock-free를 위하여 한 개의 물리적 채널을 여러 개의 가상채널로 나누고 이를 기반으로 링크결함이 임의로 배열된 n-차원 메쉬에서의 결함포용 라우팅 알고리즘을 제안한다. 제안된 알고리즘은 단순한 X-Y 라우팅 알고리즘보다 링크결함이 존재하는 네트워크를 운영하는데 있어서 보다 효율적으로 사용될 수 있으며, 이러한 사실은 본 논문에서 실행된 시뮬레이션을 통하여 증명될 것이다.

2. Preliminary

향후 사용되는 기본적인 가정들은 적응적 라우팅을 허용하는 것을 제외하고는 Dally와 Seitz[1]에 의해서 제안된 가정들을 기초로 한다. 또한 본 논문에서는 네트워크, 라우팅함수, 배열 등을 표현하기 위해서 다음과 같은 간단한 정의들을 사용한다.

- **정의 1** : 실제 네트워크는 $Net = G(N, C)$으로 표현된다. 여기에서 N은 노드들의 집합을, C는 채널들의 집합을 나타낸다.
- **정의 2** : 각 채널 c는 vc_{ijk}로 표현되는 여러 개의 가상채널로 나누어진다. 여기에서 i는 채널의 차원, j는 채널의 방향, k는 가상채널의 번호를 나타낸다.
- **정의 3** : 적응적 라우팅함수 R은 다음과 같이 표현된다.

$$R : vc_{ijk} \times S_{FREE-vc} \rightarrow vc_{lmn}$$

여기에서 vc_{ijk}는 현재 사용되고 있는 가상채널이고, vc_{lmn}은 다음에 사용될 가상채널이다. 그리고, $S_{FREE-vc}$는 이용가능한 가상채널들의 상태를 나타낸다.

- **정의 4** : 주어진 네트워크와 라우팅함수 R에 대한 채널의존그래프는 $D = G(C, E)$으로 표현된다. 여기에서 C는 네트워크상의 채널들을, E는 라우팅함수 R에 의해서 연결된 채널의 쌍들을 나타낸다.

$E=$ {(c_i, c_j) | c_i : 현재 사용중인 채널,
c_j : R에 의해 결정된 다음에 사용될 채널}

n-차원 메쉬에 대해서는 다음과 같은 표현들을 사용한다. n-차원 메쉬는 차원당 k_i, $i=0,1,\ldots,n-1$ 개의 노드를 가진다. 여기에서 $k_i>2$이다. 이 네트워크상의 노드 X는 다음과 같이 표현된다.

$$X=n_{x_0,x_1,\cdots,x_{n-1}}, \quad 0\le x_i \le k_i-1, \quad 0\le i \le n-1$$

만약 노드 Y가 j차원에서만 $y_j=x_j\pm1$이고 나머지 차원에서는 $x_i=y_i$, $0\le i\le n-1$라면, 이 노드는 X의 이웃에 위치한 노드이다. 그러므로, 각 노드는 메쉬상의 위치에 따라서 n에서 $2n$개의 이웃노드를 가진다. 한편, 출발노드 S와 목적노드 D, 현재의 노드 C는 다음과 같이 표현된다.

$$S=n_{s_0,s_1,\cdots,s_{n-1}}$$
$$D=n_{d_0,d_1,\cdots,d_{n-1}}$$
$$C=n_{c_0,c_1,\cdots,c_{n-1}}$$

이제 메시지들을 그들의 목적지로 보내기 위한 정보를 가진 라우팅표 RT를 정의한다.

$$2\ RT=[r_0,r_1,\cdots,r_{n-1}]$$
$$=[c_0-d_0,c_1-d_1,\cdots,c_{n-1}-d_{n-1}]$$

여기에서 현재 노드가 출발노드라면, $c_i=s_i$, $0\le i\le n-1$이다. 한편, 메시지들은 라우팅표의 모든 원소들이 0이 될 때 목적지에 도착하기 때문에 라우팅표의 원소들이 0으로 수렴하는 방향으로 라우팅 되어야 한다.

3. 결함포용 라우팅 알고리즘

결함포용 라우팅의 목적은 주어진 네트워크에 결함이 존재하는 상황에서도 노드들 사이에 메시지를 전달하기 위해서 요소결함이 없는 노드들의 능력을 최대화하는 것이다. 그러므로, 각 노드는 자신의 상태를 알아야 하고 메시지들을 다른 요소결함이 없는 노드로 전달할 수 있어야 한다. 이 절에서는 X-Y 라우팅 알고리즘[1]을 기초로 한 적응적 라우팅을 사용하여 결함포용 라우팅을 구현한다. X-Y 라우팅 알고리즘은 낮은 차원의 라우팅을 먼저 실행하고 점차 차원을 높여가면서 메시지들을 그들의 목적지로 전달하는 라우팅방식이다. 즉, 메시지들은 먼저 그들의 라우팅표 중에 r_0가 0이 되도록 라우팅되고, r_0가 0이 되면 이번에는 r_1이 0이 되도록 라우팅된다. 이러한 방식으로 $r_2,\cdots,r_{n-1}$들이 모두 0이 되면 메시지들은 그들의 목적지에 도달하게 되는 것이다. X-Y 라우팅 알고리즘은 구조가 간단하고 쉽게 deadlock-free를 구현할 수 있다는 장점이 있지만, 결정적 라우팅이기 때문에 네트워크상에 요소결함이 존재하면 라우팅에 방해를 받을 수 있다는 단점도 가지고 있다. 이를 극복하기 위해서 X-Y 라우팅 알고리즘에 적응적 라우팅을 첨가한다. 즉, n-차원 메쉬의 어떤 차원에서 라우팅이 진행되고 있던 메시지가 요소결함에 의해서 방해를 받으면 이를 피하기 위한 방법을 제안한다. 이를 위해서 우선 2D 메쉬 네트워크에서 deadlock을 방지를 위한 가상채널을 사용한 결함포용 라우팅 알고리즘을 구성한 후에 이를 n-차원 메쉬로 확장시킬 것이다.

3.1. 2D 메쉬

본 논문에서는 2D 메쉬에서 3개의 가상채널을 사용하는데, 이 가상채널들은 다음과 같이 정의된다.

- vc_0 : 이 가상채널은 기본적으로 0차원에서 사용되지만 요소결함 때문에 1차원에서 사용되기도 한다.
- vc_1 : 이 가상채널은 1차원에서만 사용된다.
- vc_2 : 이 가상채널은 라우팅표 [r_0, r_1]의 r_1이 0인 메시지가 0차원에서 라우팅되다가 요소결함을 만났을 때나 1차원에서 라우팅되던 메시지가 요소결함에 의해서 0차원으로 우회(detour)해야할 때 사용된다.

메쉬구조는 규칙적인 구조가 아니기 때문에 한 노드에 연결된 물리적인 채널의 수가 노드의 위치에 따라 다르다. 즉, 각 노드의 메시지를 처리할 수 있는 능력이 그 노드와 연결된 채널수에 따라 다르다. 이러한 경우들을 한번에 설명하기 위해서 테두리노드의 링크 연결이 안된 부분을 영구적 결함으로 생각한다.

이제 이러한 가상채널들을 제안될 결함포용 라우팅 알고리즘에 적용시켜 보자. 네트워크상에 요소결함이 없으면 결함포용 라우팅 알고리즘은 X-Y 라우팅 알고리즘을 수행한다. 즉, 메시지들은 그들의 라우팅표 r_0가 0이 될 때까지 vc_0를 통해 전달된 후에 r_1가 0이 될

때까지 vc_1을 통해 전달되어 목적지에 도달한다. 그러나, 네트워크상에 요소결함이 생기면 라우팅을 하는데 있어서 다른 부가적인 방법들이 필요하다. 이러한 부가적인 방법들을 포함하고 있는 결함포용 라우팅 알고리즘은 (그림 1)의 pseudo-code로 정리된다. 이 알고리즘의 이해를 돕기 위하여 5×5 메쉬에서 발생할 수 있는 다음의 네가지 경우에 대한 메시지 라우팅을 제안된 결함포용 라우팅 알고리즘의 관점에서 설명한다.

```
update;

if ( r0 is 0 and r1 is 0 ) then goto C1;
if ( r0 is not 0 ) then goto C2;
if ( r0 is 0 and r1 is not 0 ) then goto C3;

C1 : /* when all members of the routing_tag are 0 */
     send to local process;
C2 : /* when the routing_tag for 0 dimension is not 0 */
     if ( link for 0 dimension is not FAULT )
       then send to 0 dimension through vc0;
       else if ( link for 0 dimension is FAULT
                 and routing_tag for 1 dimension is not 0
                 and link for 1 dimension is not FAULT )
              then send to 1 dimension through vc0;
       else if ( link for 0 dimension is FAULT
                 and routing_tag for 1 dimension is 0
                 and link for 1 dimension is not FAULT )
              then send to 1 dimension through vc2;
       else if ( link for 0 dimension is FAULT
                 and link for 1 dimension is FAULT )
              then send to previous node through vc2
                   (i.e., backtracking);
     endif
C3 : /* when the routing_tag for dimension 0 is 0 and the
        routing_tag for dimension 1 is not 0 */
     if ( link for 1 dimension is not FAULT )
       then send to 1 dimension through vc1;
       else if ( link for 1 dimension is FAULT
                 and link for 0 dimension is not FAULT )
              then send to 0 dimension through vc2;
       else if (link for 1 dimension is FAULT
                and link for 0 dimension is FAULT)
              then send to previous node through vc2
                   (i.e., backtracking);
     endif
```

(그림 1) 제안된 결함포용 라우팅 알고리즘
(Fig. 1) A proposed fault-tolerant routing algorithm

[예제 1] 먼저 라우팅표의 $r_0 \neq 0$, $r_1 \neq 0$인 메시지가 0차원의 링크결함을 만났을 경우를 생각해 보자. 이 경우를 설명하기 위해 두 노드 n_{31}과 n_{41}사이가 링크결함이고 n_{11}에서 n_{44}로 전달되는 메시지가 발생했다고 가정한다. 그러면 이 메시지는 다음의 경로를 통하여 목적노드로 전달된다.

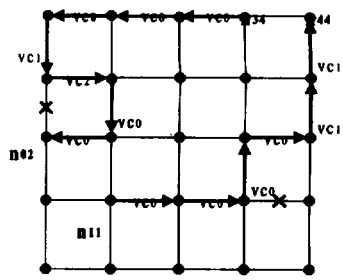
$$n_{11} \xrightarrow{[3,3],\,vc_0} n_{21} \xrightarrow{[2,3],\,vc_0} n_{31}$$
$$\xrightarrow{[1,3],\,vc_0} n_{32} \xrightarrow{[1,2],\,vc_0} n_{42}$$
$$\xrightarrow{[0,2],\,vc_1} n_{43} \xrightarrow{[0,1],\,vc_1} n_{44}$$

여기에서 화살표 위에 표시된 것은 라우팅표와 메시지가 사용하는 가상채널이다.

[예제 2] 이번에는 라우팅표의 $r_0 = 0$, $r_1 \neq 0$인 메시지가 1차원의 링크결함을 만났을 경우를 생각해 보자. 이에 대한 예로써 n_{02}과 n_{03} 사이의 링크가 요소결함이고, n_{34}에서 n_{02}로 전달되는 메시지가 발생한 경우를 고려한다. 이 메시지는 vc_2를 통하여 우회한 후에 목적노드에 도달되며 그 이동경로는 다음과 같다.

$$n_{34} \xrightarrow{[-3,-2],\,vc_0} n_{24} \xrightarrow{[-2,-2],\,vc_0} n_{14}$$
$$\xrightarrow{[-1,-2],\,vc_0} n_{04} \xrightarrow{[0,-2],\quad vc_1} n_{03}$$
$$\xrightarrow{[0,-1],\quad vc_2} n_{13} \xrightarrow{[-1,-1],\,vc_0} n_{12}$$
$$\xrightarrow{[-1,0],\quad vc_0} n_{02}$$

(그림 2)는 [예제 1]과 [예제 2]에 대한 메시지의 이동경로와 그 때의 사용채널을 보여준다.



(그림 2) 예제 1과 예제 2
(Fig. 2) Example 1 and 2

[예제 3] 다음으로 라우팅표의 $r_0 \neq 0$, $r_1 = 0$인 메시지가 0차원의 링크결함을 만났을 경우를 생각해 보자. 이 경우는 n_{23}과 n_{33} 사이의 링크가 요소결함일 때 n_{13}에서 발생하여 n_{43}으로 전달되는 메

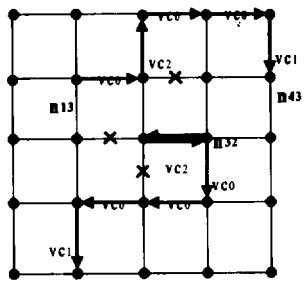
시지로 설명될 수 있다. 이 메시지는 요소결함을 만나는 노드 n_{23}에서 vc_2를 통하여 n_{24} 또는 n_{22}로 전달될 수 있으며, n_{24}를 거쳐서 전달될 경우에는 다음의 경로를 따른다.

$$n_{13} \xrightarrow{[3,0],\ vc_0} n_{23} \xrightarrow{[2,0],\ vc_2} n_{24} \xrightarrow{[2,-1],\ vc_0} n_{34} \xrightarrow{[1,-1],\ vc_0} n_{44} \xrightarrow{[0,-1],\ vc_1} n_{43}$$

[예제 4] 마지막으로 라우팅표의 $r_0 \neq 0$, $r_1 \neq 0$인 메시지가 0차원과 1차원의 링크결함을 동시에 만났을 경우를 생각해 보자. 이를 위해 n_{12}와 n_{22}를 연결하는 링크, n_{21}와 n_{22}을 연결하는 링크가 요소결함이고 n_{32}에서 n_{10}으로 전달되는 메시지가 발생했다고 가정한다. 이 메시지는 채널 vc_2를 통하여 메시지를 이전의 노드로 다시 되돌리는 회귀(backtracking)를 수행한 후에 목적노드에 전달된다. 즉, 다음의 경로를 따른다.

$$n_{32} \xrightarrow{[-2,-2],\ vc_0} n_{22} \xrightarrow{[-1,-2],\ vc_2} n_{32} \xrightarrow{[-2,-2],\ vc_0} n_{31} \xrightarrow{[-2,-1],\ vc_0} n_{21} \xrightarrow{[-1,-1],\ vc_0} n_{11} \xrightarrow{[0,-1],\ vc_1} n_{10}$$

(그림 3)은 [예제 3]과 [예제 4]에 대한 메시지의 이동경로와 그 때의 사용채널을 보여준다.



(그림 3) 예제 3과 예제 4
(Fig. 3) Example 3 and 4

지금까지 서로 다른 라우팅표를 가진 메시지가 링크결함을 만났을 때에 대해 살펴보았다. 여기에서 특별히 네가지 경우에 대해서만 설명한 이유는 2D 메쉬에서 링크결함을 만난 메시지의 라우팅표는 위에서 설명한 네가지 경우밖에 존재할 수 없기 때문이다. 다시 말해 모든 메시지 라우팅들은 이 네가지 경우의 확장으로 표현될 수 있다.

● **정리 1** : 2D 메쉬 네트워크를 대상으로 제안된 결함 포용 라우팅 알고리즘은 deadlock-free이다.

증명) 제안된 알고리즘이 deadlock-free임을 증명하기 위해서 각 채널은 기본적으로 <표 1>과 같은 채널번호가 부여된다. 단, vc_0가 요소결함 때문에 1차원에서 사용되면 다음과 같은 채널번호를 가진다.

$$(v_{temp}-1)+1/2$$

〈표 1〉 노드 n_{x_0,x_1}에 연결된 채널들의 채널번호
〈Table 1〉 The channel number of the channels connected at node n_{x_0,x_1}

가상채널과 기본적으로 사용되는 차원	채널 사용 후 노드번호가 증가하는 경우	채널 사용 후 노드번호가 감소하는 경우
vc_0(0차원)	x_0+1	k_0-x_0
vc_1(1차원)	$k_0+(x_1+1)$	$k_0+(k_1-x_1)$

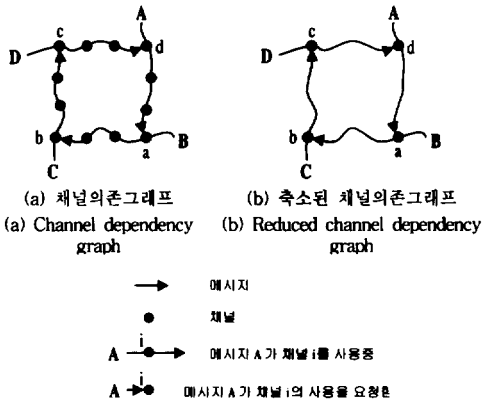
여기에서 v_{temp}는 링크결함이 없었으면 사용되었을 채널의 채널번호이다.

Deadlock은 채널의존그래프에서 사이클의 형태로 나타난다. 그러므로, deadlock-free를 이룬다는 것은 이 사이클을 없애는 것을 의미한다. (그림 4)는 deadlock이 발생한 채널의존그래프의 예이다. 여기에서 화살표는 메시지를, 까만점은 메시지가 사용중이거나 사용할 채널을 나타낸다. 먼저 채널 a, b, c, d중에 vc_2가 존재하지 않는 경우를 생각하자. 이 경우에 각 메시지가 사용할 채널의 채널번호는 반드시 이전에 사용한 채널의 채널번호보다 크거나 같아야 한다. 즉, 다음의 식들이 동시에 만족되어야 한다.

(a의 채널번호)≥(b의 채널번호)
(b의 채널번호)≥(c의 채널번호)
(c의 채널번호)≥(d의 채널번호)
(d의 채널번호)≥(a의 채널번호)

그러므로, 채널 a, b, c, d의 채널번호는 모두 같다. 그러나 이것을 만족하는 사이클은 존재할 수 없다. 따라서, (그림 4)와 같은 deadlock이 발생하려면 채널 a,

b, c, d 중에 적어도 하나는 vc_2이어야 한다. 한편, vc_2는 네트워크상에서 목적지에 접근하려는 메시지의 진행이 요소결함에 의해서 완전히 방해되었을 때만 사용되는 가상채널이다. 그러므로, vc_2가 포함된 deadlock이 발생했다는 것은 몇 개의 노드들이 고립되어 있음을 의미한다. 이러한 경우는 라우팅 자체가 무의미하다. 따라서, 결함포용 라우팅 알고리즘은 deadlock-free이다.



(a) 채널의존그래프 (b) 축소된 채널의존그래프
(a) Channel dependency graph (b) Reduced channel dependency graph

(그림 4) 4개의 메시지에 의한 deadlock 사이클
(Fig. 4) A deadlock cycle by four messages

3.2 N-차원 메쉬

이제 2D 메쉬에서 n-차원 메쉬로 토폴로지를 확장시켜 보자. n-차원 메쉬는 $n+1$개의 가상채널이 사용된다. 즉, 각 차원에 해당하는 가상채널 vc_i, $0 \le i \le n-1$과 우회를 위한 한 개의 가상채널 vc_n이 사용된다. 이러한 가상채널들을 사용하여 2D 메쉬에서 제안한 알고리즘을 확장시켜 보면 다음과 같이 정리된다.

각 메시지는 기본적으로 $r_0, r_1, \cdots, r_{n-1}$를 0으로 만드는 방향으로 라우팅된다. 그러나, 링크결함에 의해서 라우팅이 방해를 받았을 경우에 메시지는 현재 진행되고 있는 차원보다 한 단계 높은 차원의 링크를 통하여 전달된다. 예를 들어, i차원, $0 \le i < n-1$에서 r_i가 0이 되도록 라우팅되고 있던 메시지가 요소결함 때문에 더 이상 진행되지 못하면, 이 메시지는 r_{i+k}, $0 < k < n-i$가 0으로 수렴하는 방향으로 먼저 라우팅을 한 후에 다시 r_i를 0으로 만드는 방향으로 라우팅을 계속한다. 이 때 사용되는 가상채널은 vc_i이다. 그런데, i가 $n-1$이라면 이 차원보다 높은 차원이 없다. 이 경우에는 어쩔 수 없이 우회해야 한다. 따라서 본 논문에서 제안된 알고리즘은 특별한 가상채널 vc_n을 사용하여 0차원으로 우회하도록 한다. 또한, r_{i+k}, $0 < k < n-i$가 모두 0이어도 메시지들은 가상채널 vc_n을 통하여 우회한다. 한편, 우리의 알고리즘은 기본적으로 메시지들을 이전의 노드로 다시 되돌리는 회귀를 금지한다. 그러나 0이 아닌 r_i, $0 \le i < n$의 수가 2이상이고 메시지가 더 이상 진행될 수 없는 즉, 라우팅표를 0으로 수렴시킬 수 있는 링크들이 존재하지 않는 경우에 한해서 우리는 특별한 가상채널 vc_n을 통한 회귀를 허용한다.

- **정리 2** : n-차원 메쉬에서의 결함포용 라우팅 알고리즘도 deadlock-free이다.

  증명) 이것은 2D 메쉬에서 제안된 알고리즘이 deadlock-free임을 보인 정리 1의 증명과 같은 방법으로 증명될 수 있다. 이를 위해 정리 1과 유사한 방법으로 각 차원의 채널들에 대해서 기본적으로 <표 2>와 같은 채널번호를 부여한다. 단, vc_i, $i=0,1,\ldots,n-2$가 요소결함 때문에 j, $i<j \le n-1$차원에서 사용되는 경우에는 다음과 같은 채널번호를 가진다.

$$(v_{temp}-1)+j/n$$

〈표 2〉 노드 $n_{x_0,x_1,\cdots,x_{n-1}}$에 연결된 채널들의 채널번호
〈Table 2〉 The channel number of the channels connected at node $n_{x_0,x_1,\cdots,x_{n-1}}$

가상채널과 기본적으로 사용하는 차원	채널 사용 후 노드번호가 증가하는 경우	채널 사용 후 노드번호가 감소하는 경우
vc_0(0차원)	x_0+1	k_0-x_0
vc_1(1차원)	$k_0+(x_1+1)$	$k_0+(k_1-x_1)$
vc_2(2차원)	$k_0+k_1+(x_2+1)$	$k_0+k_1+(k_2-x_2)$
⋮	⋮	⋮
vc_{n-1} (n-1차원)	$k_0+k_1+\cdots+k_{n-2}+(x_{n-1}+1)$	$k_0+k_1+\cdots+k_{n-2}+(k_{n-1}-x_{n-1})$

여기에서 v_{temp}는 요소결함이 없었으면 이용되었을 채널의 채널번호이다.

그러면 정리 1의 증명과 같은 방식으로, n-차원 메쉬에서 제안된 결함포용 라우팅 알고리즘은 deadlock-free임이 증명된다.

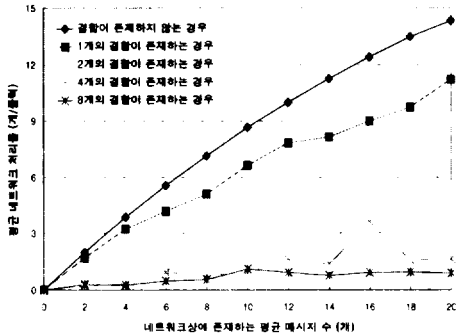
4. 시뮬레이션

제안된 알고리즘의 성능을 검증하기 위해 시뮬레이션은 먼저 간단한 2D 메쉬인 10×10 메쉬 네트워크에서 수행되며, 네트워크상의 각 노드는 프로세서, 라우터, 가상채널에 연결된 버퍼로 구성된다고 가정한다. 시뮬레이션은 다음과 같은 조건하에서 실행되었고, 본 시뮬레이션 결과는 n-차원 메쉬 등으로 확장된 토폴로지에서 시행될 때에도 유사한 결과가 예상된다.

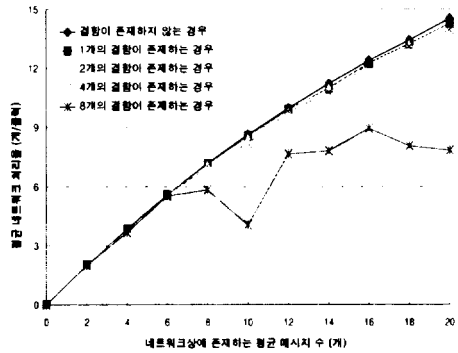
- 메시지의 길이는 평균이 10개의 플릿인 exponential distribution을 가진다.
- 각 메시지의 목적노드는 균일하게 분산된다.
- 네트워크상의 링크결함은 임의적인 장소에서 발생된다.
- 각 가상채널에 연결되어 있는 버퍼는 1개의 플릿만을 저장할 수 있다.
- A에서 B로 플릿이 전달될 때 걸리는 시간은 1 플릿사이클이다. 여기에서 A와 B는 이웃노드이다. 시뮬레이션상에서 이 플릿사이클은 네트워크클럭으로 사용된다.
- 모든 메시지들이 동일한 우선순위를 가진다.
- 네트워크상에 존재하는 평균 메시지 수와 링크결함의 수에 따른 각 시뮬레이션은 50,000클럭동안 실행된다. 그러면 strong law of large number에 의해 성능인자의 샘플 평균치는 ensemble average로 수렴된다.

라우팅 알고리즘의 성능은 평균 네트워크 처리율, 평균 메시지 도착률을 통하여 측정된다. 여기에서 네트워크 처리율은 매 클럭당 전달되는 평균 메시지 수이고 메시지 도착률은 발생된 메시지가 목적노드에 성공적으로 전달되는 비율을 말한다. 따라서 네트워크 처리율은 (메시지 수/클럭)으로, 메시지 도착률은 발생된 메시지와 도착한 메시지의 비로 표현된다.

(그림 5)와 (그림 6)은 각각 X-Y 라우팅 알고리즘을 사용했을 때와 결함포용기법인 적용된 알고리즘을 사용했을 때의 평균 네트워크 처리율을 보여준다. 두 그래프에서 확연하게 드러나듯이 요소결함이 증가함에 따라 X-Y 라우팅 알고리즘을 사용한 경우가 본 논문에서 제안된 알고리즘을 사용하는 경우보다 급격한 처리율 감소를 보인다는 것이 확인된다. 이러한 사실은 신뢰도가 중요시되는 응용에 사용되는 멀티컴퓨터에서 본 논문의 알고리즘을 사용하는 것이 X-Y 라우팅 알고리즘을 사용하는 것보다 네트워크를 효율적으로 운영할 수 있다는 것을 직접적으로 입증한다.



(그림 5) X-Y 라우팅 알고리즘의 평균 네트워크 처리율
(Fig. 5) The average network throughput of the X-Y routing algorithm



(그림 6) 제안된 라우팅 알고리즘의 평균 네트워크 처리율
(Fig. 6) The average network throughput of the proposed routing algorithm

<표 3>은 제안된 결함포용 라우팅 알고리즘이 사용되었을 때와 단순히 X-Y 라우팅 알고리즘만을 사용했을 때의 평균 메시지 도착률을 보여준다. 표에서 보듯이 X-Y 라우팅 알고리즘이 본 논문에서 제안된 알고리즘보다 링크결함 수의 증가에 따른 급격한 도착률의 감소를 보인다는 것이 확인된다. 이러한 사실은 노드간의 안정적인 메시지전달을 위해서 본 논문에서 제안된 알고리즘을 사용하는 것이 X-Y 라우팅 알고리즘을 사용하는 것보다 효과적이라는 의미로 해석되고, 따라서 열악한 환경으로 인한 네트워크 H/W 자원의 결함발생시 안정적인 메시지전달을 위해서는 본 논문에서 제안된 알고리즘이 필수적이라고 결론지을 수 있다.

〈표 3〉 평균 메시지 도착률
〈Table 3〉 The average arrival rate of messages

요소결함 갯수	X-Y 라우팅 알고리즘	제안된 알고리즘
0	1	1
1	0.996	1
2	0.956	1
4	0.892	0.999
8	0.811	0.996

5. 결 론

네트워크 요소는 언제나 요소결함이 발생할 수 있는 가능성을 지니고 있다. 중요한 것은 요소결함이 발생한 상황에서도 정상적으로 네트워크가 동작하느냐하는 것이다. 이를 위한 방법으로 본 논문에서는 결함포용 라우팅을 고려하였다. 본 논문에서 제안된 알고리즘은 기본적으로는 기존의 X-Y 라우팅 알고리즘을 사용하지만 결함포용을 위한 부분적인 적응성과 우회접근방식을 포함하고 있으며, 요소결함에 대한 지역메모리를 사용하므로 선처리과정(preprocessing)이 필요없고, 결정적 라우팅방식의 틀을 사용하기 때문에 상대적으로 구현하기가 쉽다는 특징을 지닌다. 또한 시뮬레이션의 결과로부터 열악한 환경에서 동작하는 네트워크에서 효과적인 메시지 전달을 위해서는 단순한 X-Y 라우팅 알고리즘보다 제안된 알고리즘을 사용하는 것이 고신뢰도를 가짐을 증명하였다.

본 논문에서는 결함포용 라우팅 알고리즘이 제안되었지만, 실제적으로는 결함포용 라우팅이 적용되었을 때와 적용되지 않았을 경우를 비교함으로써 결함포용 라우팅의 필요성을 부각시키는데 중점을 두었다. 따라서 앞으로는 보다 향상된 완전 적응적 결함포용 라우팅 알고리즘의 개발에 대한 연구가 필요하며, 우선순위에 기초한 라우팅방법이나 각 노드에서의 스케줄링, 대기방법 등도 연구되어야 할 부분들이다.

참 고 문 헌

[1] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers*, Vol.C-36, No.5, pp.547-553, May, 1987.

[2] W. J. Dally, "Virtual-Channel Flow Control," *IEEE Trans. on Parallel and Distributed Systems*, Vol.3, No.2, pp.194-205, March, 1992.

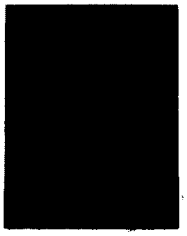
[3] D. H. Linder and J. C. Harden, "An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary *n*-cubes," *IEEE Trans. Computers*, Vol. 40, No.1, pp.2-12, January, 1991.

[4] C. J. Glass and L. M. Ni, "Fault-Tolerant Routing in Meshes," *FTCS-23*, 1993, pp.240-249

[5] R. V. Boppana and S. Chalasani, "Fault-Tolerant Wormhole Routing Algorithms for Mesh Networks," *IEEE Trans. Computers*, Vol.44, No.7, pp.848-864, July, 1995.

[6] Y. M. Boura and C. R. Das, "Fault-Tolerant Routing in Mesh Networks," *International Conference on Parallel Processing*, 1995.

[7] C. C. Su and K. G. Shin, "Adaptive Fault-Tolerant Deadlock-Free Routing in Meshes and Hypercubes," *IEEE Trans. Computers*, Vol.45, No.6, pp.666-683, June, 1996.



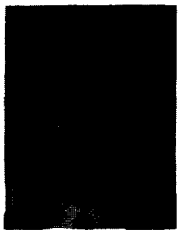
문 대 근

e-mail : dkmoon@kalman.yonsei.ac.kr

1997년 2월 연세대학교 전기공학과 졸업(학사)

1997년 9월~1999년 현재 연세대학교 대학원 기계전자공학부 전기전공 석사과정

관심분야 : Fault-tolerant 시스템, 실시간 제어시스템



김 학 배

e-mail : hbkim@bubble.yonsei.ac.kr

1988년 서울대학교 전자공학과 졸업(학사)

1988년~1994년 The Univ. of Michigan EECS 공학석사 및 박사

1994년~1996년 NASA Langley 연구센터 NRC 연구원

1996년~1998년 현재 연세대학교 기계전자공학부 전기전공 조교수

관심분야 : 실시간 시스템, Fault-tolerant computing, 자동화 및 제어 시스템