2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

자기 루프 스위치를 가진 결함-허용 다단계 상호연결망

김금호*, 김형욱*, 남순현**, 윤성대*
*부경대학교 전자계산학과
**부경대학교 전산교육학과
e-mail:kim3513@dol.pknu.ac.kr

A Fault-Tolerant Multistage Interconnection Network with Self-Loop Switch

Gum-Ho Kim*, Hyung-Wook Kim*, Soon-Hyun Nam**, Sung-Dae Youn*
*Dept. of Computer Science, Pukyung National University
**Dept. of Computer Science Education, Pukyung National University

요 약

본 논문은 다단계 상호연결망에서 높은 처리율과 결함 허용을 위해 중복 경로를 제공하는 E-Cube-network 구조를 제안한다. Cube network의 확장된 형태인 E(Extended)-Cube network 구조를 통해 패킷 전송중 충돌이 발생한 경우 스위치 자신을 순환하고, 재차 충돌이 발생하면 멀티플렉서를 통해 또 다른 새로운 경로로 패킷을 전송하는 새로운 알고리즘을 제시한다. 그리고 모의실험을 통해 기존의 Cube network 구조보다 높은 종단간 처리율을 보인다.

1.서론

하드웨어 발달과 더불어 컴퓨터 네트워크와 디지털 데이터통신은 놀라운 속도로 발전해 왔다. 네트워크를 구성하는 요소 중 교환망은 다양한 유형이 있지만 그 중 가장 보편적으로 사용되는 다단계 상호연결망(MIN:Multistage Interconnection Network)은 끊임없이 연구되어지고 있다[1]. 1980년대 초반부터 관심을 가져온 다단계 상호연결망은 다중 병렬처리 시스템에서 다수의 프로세서와 다수의 프로세서 또는 메모리 모듈간의 통신할 수 있는 상호연결망으로 사용되어지고 있다. 그 결과 다중 병렬처리 시스템의 성능은 상호연결망의 설계에 의해 많이 좌우되며, 넓은 대역폭과 높은 속도를 요구하는 광대역 종합정보통신망(B-ISDN)의 가장 핵심 기술인 ATM 교환망의 기본 구조로 대두되고 있다[2]. 다중 병렬처리 시스템은 수백 혹은 수천 개의 프로세서를 포함하게 되므로 프로세서 수가 증가함에 따라 결함이 발생할 확률도 증가하게 된다. 그러므로 보다 높은 신뢰성을 유지하기 위해서는 결함이 발생했을 경우에도 동작하는 결함-허용 라우팅 알고리즘 및 하드웨어 체계가 필요하다. 단일 경로 제공시 스위치 결함이나 블록 상태를 경험하면 패킷이 폐기되므로 다중 경로를 두어 결함이 발생하더라도 이를 허용함으로써 다단계 상호연결망의 처리율을 증가시킨다[3]. 기존 연구로는 추가적인 stages를 첨가함으로 결함을 허용하는 ESC[4]가 제안되어졌고, Omega network에 멀티플렉서와 디멀티플렉서를 추가한 MD-Omega network[5,6]이 제안되어 있다.

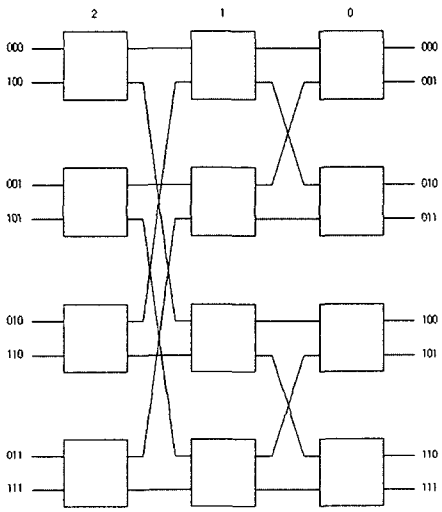
본 논문은 스위치에 보조적인 링크를 추가하여 블록시 결함을 허용함과 동시에, 멀티플렉서와 디멀티플

렉서를 추가하여 스위치의 결함에도 결함을 허용하는 다단계 상호연결망을 제안하고자 한다. 본 논문의 구성은 다음과 같다. 2장에서는 Cube network의 구조와 제안된 E-Cube network의 구조 및 라우팅 태그에 대해 알아보고, 3장에서는 제안된 네트워크 구조의 라우팅 알고리즘에 대해 설명한다. 4장에서는 모의실험에 대한 결과를 보여주고, 마지막 5장에서 결론 및 향후 과제를 제시한다.

2. 제안된 모델

2.1 Cube network

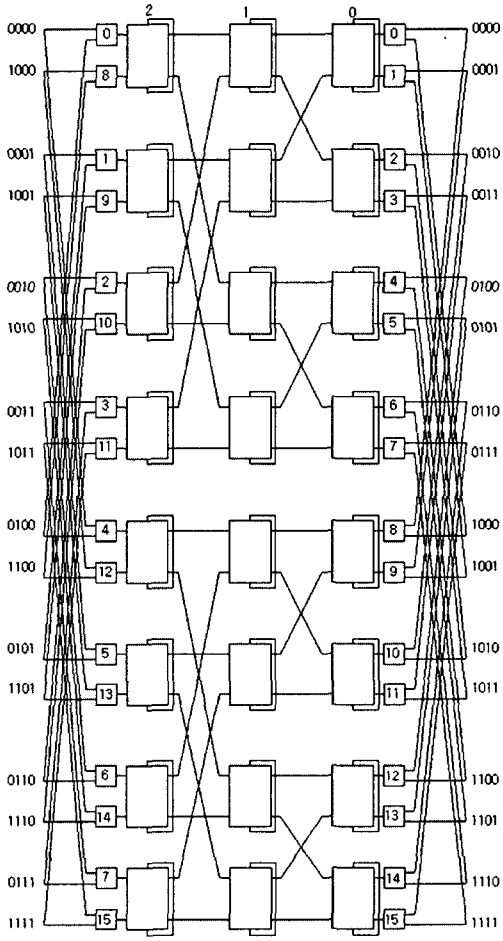
본 논문에서는 Cube network에 기반한 E(Extend)-Cube network을 제안한다. Cube network는 N개의 입력과 N개의 출력을 가지며, $n \times n$ 스위치를 가진 각 stage에 $\log_n N$개의 스위치로 구성되고, 단일 경로를 제공하는 네트워크 구조이다. 그림 1은 8×8 Cube network를 보여준다. Cube network의 라우팅 태그는 소스와 목적지 주소를 가지고 exclusive-OR연산을 통해 라우팅 태그를 계산한다. 만약 $(d_i \oplus s_i)$ 값이 0이면 직진접속이고 1이면 교차접속 라우팅 태그를 사용한다. 예를 들어 소스 0에서 6까지의 패킷 전송은 110의 라우팅 태그를 생성한다.



(그림 1) 8×8 Cube network

2.2 E-Cube network

E-Cube network는 Cube network의 확장으로 Cube network의 입력 stage에 2×1 멀티플렉서를, 마지막 stage에 1×2 디멀티플렉서를 추가한다. $N=2^n$포트를 가진 E-Cube network는 $n-1$개의 stages를 가진다. 기존의 다단계 상호연결망은 n개의 stages를 가지지만, E-Cube network의 구조는 하나의 stage는 멀티플렉서와 디멀티플렉서로서 대신한다. 블록이 발생시 ON 상태로 패킷이 다시 시도하게 도와주지만 충돌이 없을 때는 항상 OFF 상태를 유지하는 보조적인 링크를 추가한다. 그러므로 블록이 발생하면 한 클럭 사이클동안 보조적인 링크를 통해 스위치 자신을 돌아 다시 출력을 시도한다. 이때 두 가지 경로를 가지므로 두 가지 경로를 경험한 패킷인지 구별하기 위해 reroute tag를 추가하고, 스위치상의 블록을 경험한 패킷인지 구별하기 위해 loop tag를 패킷에 추가한다. 그림 2는 16×16으로 구성된 E-Cube network를 보여준다.



(그림 2) 16×16 E-Cube network

2.3 라우팅 태그

네트워크의 라우팅은 목적지까지 어떤 스위치들을 통해 갈 것인지를 나타내는데, 패킷의 헤더부분에 목적지까지 경로를 배정할 라우팅 태그를 포함하고 있다. 제안한 E-Cube network 역시 목적지 주소가 라우팅 태그인 self-routing 가능 network이다. 목적지 D의 $n-bit$ 표현은 $d_{n-1}d_{n-2}\cdots d_0$이고, 목적지 주소에서 MSB(Most Significant Bit) 비트를 제외한 나머지 비트가 라우팅 태그이다. 라우팅 태그의 $n-bit$ 표현은 $d_{n-2}d_{n-3}\cdots d_0$이다. 이때 stage의 숫자는 라우팅 태그의 아래첨자와 동일한 태그 비트에 의해 라우팅 된다. 만약 $d_i==0$이면 상위출력이고 $d_i==1$이면 하위출력이다. 예를 들어 프로세서 0에서 메모리 6으로 패킷을 전송하고자 할 때 라우팅 태그는 110이며 이때 stage 2는 1이고 stage 1은 1 stage 0은 0으로 가면 목적지까지 전달이 가능하다.

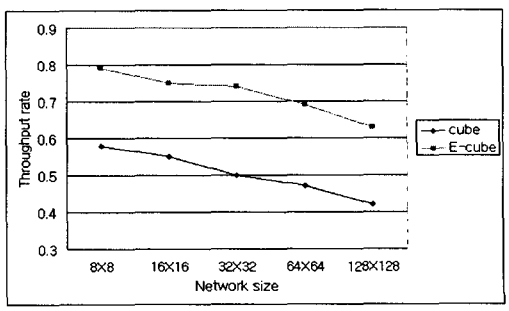
3. 라우팅 알고리즘

앞 절에서 언급한 라우팅 태그를 가진 결함-허용 라우팅 알고리즘을 소개한다. 프로세서에서 패킷을 전송할 때 먼저 프로세서 숫자와 같은 멀티플렉서로 접속하여 패킷을 전송한다. 스위치에서 충돌이 없으면 다음 stage의 스위치로 전송하고 만약 충돌이 발생하여 패킷이 다음 stage의 스위치로 전송이 불가능하면, 패킷은 보조적인 링크를 통해 다시 출력을 시도한 후, 성공하면 다음 stage로 전송되고, 실패하면 프로세서에서 다른 연결된 멀티플렉서로 다시 패킷을 전송한다. 그런 다음 위의 방법을 반복 수행한다. 이때 또 두 번 실패하면 그 패킷은 폐기한다.
이에 대한 알고리즘은 다음과 같다.

```
/* routing_algorithm 함수는 라우팅 태그에 따라 라우팅 하는데 사용하는 함수로 블록이 발생시 loop tag에 의해 다시 출력 시도한다. */
routing_algorithm() {
if (!conflict)                    // 블록 여부 체크
    reach destination;
else if( loop tag == 0 )    // 블록 경험 유·무 체크
        loop tag = 1;
        retry;
        if (conflict)
            loop tag = 0;
            call_reroute();
        else
            loop tag = 0;
            reach destination;
    else call_reroute();
}
/* call_reroute 함수는 이것이 첫 번째 경로인지 아닌지를 확인하여 두 번째 경로일 때 폐기하는 함수 */
call_reroute()
{
// 두 번째 경로이면서 두 번 실패이면 폐기
        if (reroute tag == 1)
            discard;
// 아니면 두 번째 멀티플렉서로 연결 설정
        else {
            reroute tag = 1;
            send to other multiplexer;
            call routing_algorithm();
            }
}
```

4. 모의 실험 및 결과

제안된 네트워크 구조의 종단간 신뢰도를 보이기 위해 네트워크 크기에 따라 모의실험을 수행한다. 모의실험에서 패킷 생성과 목적지 주소는 랜덤 함수로 발생시켜 얻고, 같은 네트워크 크기에서는 동일한 값을 가지고 Cube와 E-Cube network를 비교하였다. 그림 3은 Cube와 E-Cube network의 종단간 처리율을 나타내고 있다. 결과를 보면 8×8 네트워크 크기에서는 Cube는 0.58의 처리율, E-Cube는 0.79의 처리율을 가지고, 16×16 네트워크 크기에서는 Cube는 0.55 E-Cube는 0.75의 처리율을 보인다. 네트워크의 크기를 증가함에 따라 이와 유사하게 0.2 정도의 처리율 차를 보인다.



(그림3) Cube network와 E-Cube network의 종단간 처리율 비교

5. 결론

본 논문에서는 멀티플렉서와 디멀티플렉서를 첫 stage와 마지막 stage에 추가함으로써 스위치 결함에도 패킷 전송이 가능하도록 하였고, 패킷 전송 중 충돌이 발생한 경우 스위치 자신을 순환하고, 재차 충돌이 발생하면 멀티플렉서를 통해 또 다른 새로운 경로로 패킷 전송을 하도록 한 다중경로를 제공하는 다단계 상호연결망인 E-Cube network를 제안했다.

모의실험을 통해 네트워크의 종단간 처리율을 분석하여, 제안된 구조의 종단간 처리율 즉 소스에서 목적지까지 결함이 없는 경로를 발견하는 라우팅 알고리즘의 처리율이 네트워크의 크기가 증가함에 따라 거의 0.2 정도의 처리율 차를 보임으로써 일반적인 Cube network구조보다 더 효율적임을 알 수 있었다.

향후과제로는 본 논문에서 제시되지 않은 통신 지연시간과 비용을 고려한 성능 평가를 본 논문에서 제안한 모델을 적용하는 것이다.

참고문헌

[1] C. L. Wu, and T. Y. Feng, "On a class of multistage interconnection network", IEEE Trans. Computers, Vol. C-29, pp. 694-702, Aug. 1980

[2] F. A. Tobagi, "Fast packet switch architectures for broadband integrated service digital network", Proc. IEEE, Vol. 78, No. 1, pp. 133-167, Jan. 1990

[3] G. B. AdamsIII, D. P. Agrawal and H. J. Siegel, "A survey and comparison of fault-tolerant multistage interconnection networks", IEEE Computers, pp. 14-27, June 1987.

[4] G. B. AdamsIII, and H. J. Siegel, "The Extra Stage Cube : A Fault-Tolerant Interconnection Network for Supersystems", IEEE Trans. on Computers, Vol. C-31, No. 5, May 1982.

[5] S. J Wang, "Distributed routing in a fault-tolerant multistage interconnection network", Information Processing Letters 63, pp. 205-210, 1997

[6] S. J Wang, "Load-balancing in multistage interconnection networks under multi-pass routing", Parallel Distributed Comput. 36, pp. 189-194, 189-194