공유 버스를 사용한 멀티캐스트 Cut-through 스위치의 설계

(Design of Multicast Cut-through Switch using Shared Bus)

백 정 민 † 김 성 천 ††
(Jungmin Baek) (Sungchun Kim)

요 약 스위치 형식의 네트워크이 많은 주목을 받고 있다. 그것은 높은 네트워크 성능을 요구하는 환경에 매우 적합하기 때문이다. 일반적인 공유매체 지역 네트워크는 만족할 만한 처리율과 지연시간을 제공하지 못한다. 특히 멀티미디어 어플리케이션이 증가하면서 통신 성능이 보다 중요시 되고 있다. 이러한 환경에서 스위치 형식의 네트워크는 우수한 성능을 보인다.

스위치 형식의 네트워크는 높은 대역폭과 낮은 처리 시간을 얻을 수 있다. 따라서 스위치 형식의 지역 네트워크를 구성할 때 고속(high-speed)의 스위치가 중요하다. 효율적인 스위치 디자인이 스위치 형식의 네트워크 성능을 향상시키는 중요한 요소인 것이다. 또한 멀티캐스트 메시지 처리의 중요성이 높아지면서, 효과적인 멀티캐스트를 지원하는 스위치의 설계가 필요하다. 기존의 컷-스루(cut-through) 스위칭 기술(switching technique)에서는 스위치 원소(switch element)의 구조를 변경시켜 데드락을 피하면서 멀티캐스팅이 가능하게 하였다. 그러나 처리율의 저하와 스위치 크기의 증가의 문제를 안고 있다.

따라서 하드웨어적으로 유니캐스트와 멀티캐스트를 분리함으로써 효율적인 멀티캐스팅을 가능하게 한다. 본 논문에서는 이러한 구조를 통해 멀티캐스팅에 있어서 성능 향상을 보이는 스위치 구조를 제안한다.

Abstract Switch-based network is suitable for the environment of demanding high performance network. Traditional shared-medium Local Area Networks(LANs) do not provide sufficient throughput and latency. Specially, communication performance is more important with multimedia application. In these environments, switch-based network results in high performance.

A kind of switch-based network provides higher bandwidth and low latency. Thus high-speed switch is essential to build switch-based LANs. An effective switch design is the most important factor of the switch-based network performance, and is required for the multicast message processing. In the previous cut-through switching technique, switch element reconfiguration has the capability of multicasting and deadlock-free. However, it has problems of low throughput as well as large scale of switch.

Therfore, effective multicating can be implemented by using divided hardware unicast and multicast. The objective of this thesis is to suggest switch configuration with these features.

1. 서 론

네트워크 환경이 점점 고속화 대용량화하는 방향으로 변하고 있다. 그것은 사용자들이 좀 더 빠르게 대규모의 데이타를 전송하고 싶어하기 때문이다. 멀티미디어 환경에 필요로 하는 네트워크 성능을 사용자들이 요구하고 있는 것이다. 스위치 기반 네트워크는 높은 네트워크 성능을 요구하는 이러한 상황에서 널리 적용되고 있다.[1]

공유 매체 네트워크에서 존재하는 많은 문제들이 역시 스위치에서도 발생한다. 따라서, 좋은 스위치 디자인

† 비 회 원 : 서강대학교 컴퓨터학과
friends@arqlab1.sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
ksc@arqlab1.sogang.ac.kr
논문접수 : 1999년 2월 24일
심사완료 : 2000년 1월 21일

과 스위치의 상호연결은 LAN의 성능을 증가시키는데 매우 중요한 부분을 담당한다. 그것은 분산 멀티미디어 어플리케이션에 대한 효과적인 지원과 워크스테이션의 네트워크(Network Of Workstation : NOW)에 기반을 둔 높은 성능의 컴퓨팅에 있어서도 중요한 역할을 하는 것이다.

LAN의 스위치를 구성할 때 사용되는 두 가지 흔한 스위칭 기술이 있다. 즉, 저장하고 보내기(store-and-forward)와 컷-스루(cut-through)이다. 저장하고 보내기는 데이타를 전송하기 전에 전체 데이타 프레임이 버퍼에 저장되어야만 한다. 이 방법의 단점은 처리 시간의 증가와 많은 버퍼 공간을 필요로 한다는 것이다. 특히 프레임-스위칭 네트워크에서 그러하다. 컷-스루 스위칭은 패킷의 헤더를 받고 디코딩되었으면 부분적으로 받아진 데이타를 전송할 수 있다. 따라서 낮은 지연 시간을 보장하고, 버퍼 공간에 대한 부담을 덜 수 있다.[2]

멀티캐스트 통신은 비디오 회의, 분산 가상 현실등의 네트워크 어플리케이션에 있어서 기본적인 서비스가 되어 가고 있다. 따라서 효과적인 멀티캐스트를 지원하는 스위치 디자인이 또하나의 중요 요소이다. 더구나, 컷-스루 스위칭 기술은 하드웨어 멀티캐스트를 좀 더 복잡하게 한다. 그것은 유니캐스트 통신에 비해서 데드락(deadlock)을 피하기가 훨씬 어렵기 때문이다.[3]

기존의 멀티캐스트를 지원하는 컷-스루 스위치는 이러한 데드락을 피하려는 노력을 계속해왔다. 스위치를 구성하는 기본 원소를 양방향전송에서 동쪽-북쪽 라우팅으로 개선함으로써 데드락을 피하면서 멀티캐스트를 가능하게 하였다.[1] 그러나 스위치의 크기가 두 배 이상 커지는 단점을 가지고 있다. 이것은 스위치내의 전력소모를 크게한다. 또한 유니캐스트 메시지와 멀티캐스트 메시지를 같은 구조내에서 해결하려고 함으로써 메시지간의 충돌로 인한 블록킹의 빈도가 높아지게 된다.[4]

본 논문에서는 컷-스루 스위치의 멀티캐스트 시에 발생하는 이러한 문제를 해결한다. 즉, 유니캐스트 메시지와 멀티캐스트 메시지를 하드웨어적으로 분리해서 처리함으로써 메시지간의 충돌을 억제하고 또한 공유 버스를 사용함으로써 좀 더 빠른 전송을 가능하게 한다.

하드웨어적으로 분리하는 메시지 처리방법은 처리율의 향상을 얻을 수 있다. 그것은 스위치의 크기가 비교적 작고, 메시지의 크기가 작을 때 기존의 방법에 비해서 월등한 처리율의 향상을 보인다. 따라서 멀티캐스트가 중요시 되는 통신 환경에 적합한 방법이라 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 컷-스루 스위치의 디자인에 있어서 중요한 점을 살펴보고, 멀티캐스트 통신에 대해서 알아본다. 3장에서는 일반적인 컷-스루 스위칭에서의 멀티캐스트를 개선하는 멀티캐스트 컷-스루 스위치의 디자인을 제안한다. 4장에서는 기존의 컷-스루 스위치와 멀티캐스트 컷-스루 스위치와의 성능 비교를 통해 향상된 점을 알아 본다. 마지막으로 5장에서 결론을 내린다.

2. 컷-스루 스위치

2.1 컷-스루 스위치

컷-스루 스위치는 멀티컴퓨터 시스템에서 중요한 스위치 기술이 되었다. 일반적인 컷-스루 스위치의 디자인 모듈은 <그림 1>와 같다.

<그림 1>은 8개의 입력 포트와 8개의 출력 포트로 구성되어진 스위치이다. 사용되지 않는 채널은 스위치간의 확장에 사용된다.

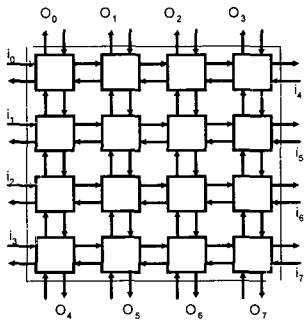

그림 1 모듈로 구성된 2D 메쉬형태의 스위치

데드락 방지를 위해서 일반적으로 XY라우팅이 사용된다. XY라우팅에서 메시지는 먼저 X 축방향으로 진행하고, 그 다음 Y축으로 방향을 전환한다. ATOMIC[5] 프로젝트는 이 방법을 기본 라우팅으로 사용한다. 그러나 이러한 디자인은 멀티캐스트를 지원하려고 할 때 데드락에 빠지는 단점을 가지고 있다. <그림 2>가 예이다. 예를 들어 입력 i_0에서 출력 o_5와 o_6으로 멀티캐스트를 한다. 그리고 동시에 입력 i_4에서 o_5와 o_6으로 멀티캐스트를 한다고 가정한다. 이 때 스위치내에 다른 트래픽은 존재하지 않는다고 가정할 때, 상호 배제 상황(mutual blocking situation)이 발생한다. 또한 스위치가 완전 적응(fully adaptive) 라우팅을 사용한다고 할

때 데드락이 발생된다.

따라서 멀티캐스트를 지원하고 완전 적응 라우팅을 지원하기 위해서 다음과 같은 <그림 3>의 모듈을 Ni가 제안하였다. <그림 3>은 두 개의 입력 포트와 두 개의 출력 포트를 가지고 있고 다른 모듈과의 연결을 위해서 4개의 추가적인 채널을 가지고 있다.

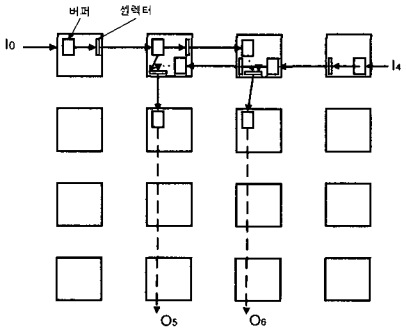

그림 2 멀티캐스팅시에 데드락의 발생

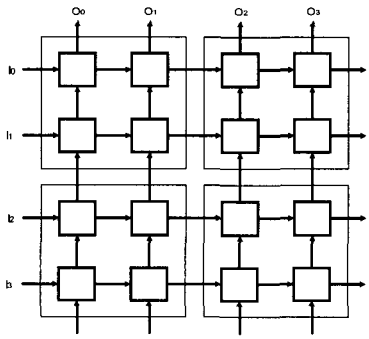

그림 3 Ni의 스위치 모듈로 구성한 4x4 스위치

이것은 크로스바와 같은 모양을 하고 있다. 그러나 크로스바와 근본적으로 다른 점은 입력과 출력사이에서 연결을 설정하는 것이 아니라 , 각각의 연결점인 스위치 원소에서 컷-스루 스위치을 사용하는 것이다. 따라서 입력과 출력 사이에서 연결 정보(connection information)를 유지하고 있을 필요가 없고 모든 채널은 매순간 필요할 때마다 즉시 할당되게 된다. <그림 4>는 2x2 모듈을 사용하여서 4x4의 스위치를 구성한 그림이다.

이런 새로운 크로스바 비슷한 스위치는 스위치 로직을 단순화 시킨다. 각각의 스위치 원소는 기존의 것이 4개의 입력과 4개의 출력을 가진 것에 비해, 2개의 입력과 2개의 출력만을 가진다.

<그림 5>는 스위치 원소의 그림이다. 두 개의 작은 버퍼를 가지고 있다. 만약 버퍼가 모두 차면 백 프레저가 발생된다. 두 개의 채널 셀렉터(selectors)는 버퍼에 있는 데이타가 어느 출력 채널을 사용할 지 결정하게 된다.

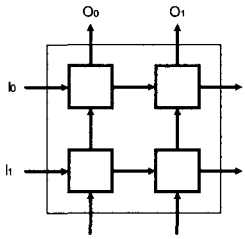

그림 4 Ni의 스위치 모듈

컷-스루 라우팅은 네트워크에 메시지를 작은 단위로 나눠서 스위치되게 한다. 헤더 유닛은 패킷의 길과 채널 예약등에 필요한 라우팅 태그를 가지고 있다. 헤더 유닛이 특정한 길을 따라서 진행하면 , 나머지 유닛들은 파이프라인 형식으로 따라서 진행하게 한다. 트레일러(trailer) 시그널 또는 특별한 캐릭터가 할당된 채널들을 해제시킨다.

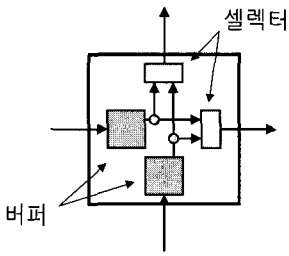

그림 5 Ni의 스위치 원소

2.2 멀티캐스트의 지원

컷-스루 스위치 기술을 사용하는 네트워크에서 멀티캐스팅은 쉽지 않은 일이다. 데드락에 빠지지 않는다는 보장을 하는 것이 어렵기 때문이다.[6]

기존의 멀티캐스트를 지원하는 스위치는 다음과 같은 방법으로 진행된다. 데이타의 프레임은 스위치 원소에서 복사가 일어나서, 두 개의 출력 채널을 통해서 진행한다. 예를 들어 <그림 2>에서는 하나의 메시지 헤더가

두 개로 복사가 되어서 두 개의 출력 채널을 통해서 진행하는 것을 볼 수 있다.

멀티캐스트를 위한 라우팅은 XY의 라우팅의 변형인 relaxed XY 라우팅을 하게 된다. 즉, 먼저 헤더 유닛이 X축방향으로 진행한다. 멀티캐스팅을 위한 목적지의 Y축에 도달할 때까지 X축 방향으로만 진행하게 된다. 목적지의 Y축에 도달하게 되면, 현재 위치의 스위치 원소에서 복사가 일어나서 하나는 Y축방향으로 하나는 X축방향으로 진행하게 된다. 이와 같이 라우팅을 함으로써 데드락을 피하면서 효과적인 멀티캐스팅을 할 수 있다.

3. 멀티캐스트 컷-스루 스위치

3.1 공유 매체를 적용한 스위치의 구현

3.1.1 제안 사항

스위치 방식의 네트워크에서 멀티캐스트의 구현은 쉽지가 않다.[7] 그것은 데드락에 대한 위험이 크고, 처리 시간의 단축을 보장받을 수 없기 때문이다.

그러나 공유 매체를 사용하는 네트워크에서는 물리적인 특성에 의해서 모든 호스트들이 쉽게 메시지를 전달받을 수 있다. 즉, 멀티캐스팅에 아무런 문제가 없는 것이다.

이러한 특성을 스위치 내부에도 적용시켜본다. <그림 6>이 예이다. 즉, 유니캐스트와 멀티캐스트를 분리한 스위치를 디자인한다.

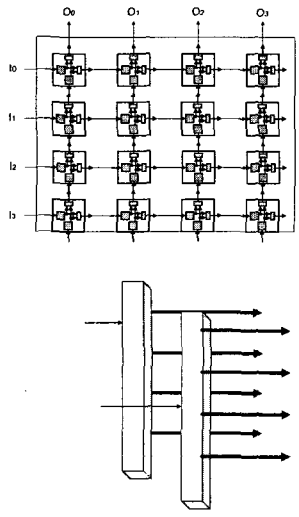

그림 6 공유 버스를 사용하는 스위치 구조

윗부분의 스위치 원소들로 메쉬와 같은 모양을 하고 있는 것이 기존의 컷-스루 스위치이다.

이 기존의 스위치 구조는 메시지들중 유니캐스트 메시지에 대해서만 라우팅을 담당하게 되는 것이다. 그리고 멀티캐스트 메시지가 들어오면 아랫부분의 공유 버스를 이용해서 메시지를 전송하게 된다. 이렇게 유니캐스트와 멀티캐스트 메시지에 대해서 각각 다른 스위치 구조를 사용하게 함으로써 동시에 처리할 수 있는 메시지의 양이 훨씬 증가될 수 있다. 또한 메시지 지연 시간에 있어서도 이득이 될 것이다.

3.1.2 기존의 스위치와의 라우팅 비교

기존의 컷-스루 스위치의 처리 시간은 멀티캐스트 메시지의 목적지중 가장 긴 경로의 목적지에 의해서 결정된다. 그것은 파이프라인 형식으로 전송을 하게 되기 때문이다. 이 때 공유 버스를 사용하는 스위치 구조에서는 하나의 패킷을 한 번에 처리할 수 있기 때문에 훨씬 효과적인 전송을 하는 것이다.

또한 유니캐스트 메시지들은 따로 기존의 스위치 구조를 통해서 전송될 수 있기 때문에, 유니캐스트의 경우에 있어서도 훨씬 효과적인 방법이 될 수 있다.

3.2 스위칭방법

3.2.1 공유 버스를 사용하는 스위치에서의 라우팅 방법

공유 버스를 적용한 스위치의 전체적인 구조는 <그림 7>와 같다. 기존의 컷-스루 스위치의 구조에 공유 버스를 중앙에 추가하였다.

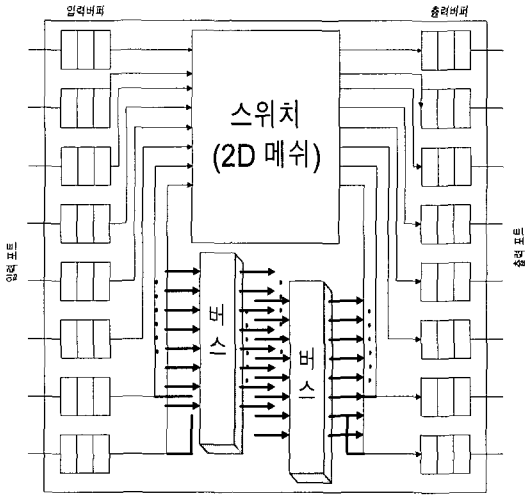

그림 7 공유버스를 사용하는 스위치의 전체적인 구조

이러한 공유 버스를 사용하는 스위치의 라우팅은 다음과 같다. 기존의 헤더에 있어서 다중목적지를 위한 비트가 있다. 즉, 이 비트값이 0이면 유니캐스트 메시지를 의미하고 0이 아니면 멀티캐스트 메시지가 된다.

유니캐스트 메시지는 위쪽의 컷-스루 스위치 원소를 통해 전송이 된다. 멀티캐스트 메시지의 경우는 아래의 공유 버스로 전송이 되게 된다.

공유 버스로 전송이 된 메시지는 모든 출력 포트의 버퍼로 전달이 된다. 이 버퍼에서는 콘트롤 로직에 의해서 자신의 포트에 해당하는 메시지이면 복사를 해서 전달을 하고, 그렇지 않으면 버리게 된다. <그림 8>은 메시지들이 작은 패킷들로 나뉜 후 각각의 패킷들의 내용을 나타낸다.

데이타	멀티캐스트 비트	XY 라우팅 헤더

그림 8 패킷의 내용

자신의 출력 포트와 일치됨을 찾는 방법은 다음과 같다. 즉, 메시지의 헤더에 각각의 출력 포트 수만큼의 비트를 갖고 있다. 각각의 비트의 자리는 해당하는 출력 포트의 위치를 나타낸다. 따라서 비트가 1인 곳이 바로 출력 포트를 요청한 곳이 된다. 예를 들어 8x8 스위치라면 8개의 비트를 가지게 된다. 8개의 비트가 0000 1110이라면 맨 오른쪽이 0번 출력 포트가 되고 맨 왼쪽이 7번 포트가 된다. 위의 예에서는 1번,2번,3번 출력 포트를 요청하고 있는 멀티캐스트 메시지가 되는 것이다. XY 라우팅 헤더의 값은 유니캐스트 메시지의 라우팅에 사용된다.

3.2.2 다중 멀티캐스트의 지원

다중 멀티캐스트의 지원을 위해서는 공유 버스가 다수개 존재하여야 한다. 즉, 동시에 여러개의 입력 포트에서 멀티캐스트를 요청할 경우에 대한 지원을 해야한다.

일반적인 컷-스루에서는 다중 멀티캐스트가 쉽게 지원된다. 그러나 스위치내에서의 패킷 충돌이 빈번하게 되고, 그 만큼 대기 시간이 증가하게 되어서 처리율은 좋지 못하게 된다.

공유 버스를 한 개를 사용하는 경우는 한 번에 오직 하나의 메시지에 대한 멀티캐스트만을 허용하게 되어서 다중 멀티캐스트를 지원하지 못한다. 따라서 n개의 입력과 출력을 가지는 스위치 구조에서 n개의 공유 버스를 갖는 다면 완벽한 다중 멀티캐스트를 지원할 수 있을 것이다. 그러나 이것은 많은 자원의 낭비를 초래한다. 또한 출력 포트에서 동시에 모든 메시지를 수용할 수 없고, 일정한 한계가 있게 마련이다. 따라서 적당한 개수의 공유 버스가 필요하게 된다. 이것은 실험을 통해 얻을 수 있었다. 실험 평가에서 알 수 있지만, 공유 버스를 통한 멀티캐스트는 일반 멀티캐스트에 비해서 월등한 성능 향상을 보이고 있다. 따라서 2개의 공유 버스만으로도 기존의 스위치에서보다 높은 성능을 얻을 수 있었다.

공유 버스의 수는 임의로 증가할 수 있다. 메시지의 처리율과 비교해서 보면 입력포트의 수를 n이라 하면 최대 n/2 개의 버스면 될 것으로 본다. 즉 2개의 입력 포트당 하나의 공유 버스를 사용하는 경우이다. 따라서 한번에 n/2개의 메시지들의 다중 멀티캐스트를 지원할 수 있는 것이다. 공유 버스를 사용하는 경우의 처리율과 처리 시간을 고려해 볼 때 충분한 성능 향상을 기대할 수 있다.

3.2.3 로드 밸런싱과 콘트롤 로직

기본적으로 공유 버스를 사용하는 스위치 구조에서 유니캐스트 메시지는 2D 메쉬 구조의 스위치 원소들을 통해 라우팅하고, 멀티캐스트 메시지에 대해서만 공유 버스를 통해 라우팅을 한다.

그러나 멀티캐스트의 메시지가 많아지면 공유 버스의 로드가 많아지는 반면, 메쉬구조의 스위치 보드는 많이 쉬게 된다. 따라서 로드가 한 쪽에 과다하게 부과되는 단점을 보안하기 위해서 콘트롤 로직에 의해서 상호 보안적으로 동작하도록 한다. 즉, 공유 버스의 로드가 높을 경우에는 메쉬 구조의 스위치 보드를 사용하고, 또 반대로 유니캐스트 메시지의 로드가 높을 경우에는 유니캐스트 메시지일지라도 공유 버스를 통해서 신속한 라우팅이 가능하게 한다.

이를 위해서 기존의 콘트롤 로직과 함께 공유 버스를 담당하는 콘트롤 로직을 추가한다. 추가하는 콘트롤 로직은 멀티캐스트 메시지들을 공유 버스를 통해서 라우팅되게 한다. 또한 공유 버스의 로드가 높을 경우에는 메쉬 구조의 스위치 보드를 사용하게 한다.

기존의 컷-스루 스위치에 멀티캐스팅을 위해 공유 버스를 사용할 것을 제안하였다.제안한 방법의 성능 평가를 다음 장에서 한다.

4. 성능 평가

본 논문에서 제안하는 멀티캐스트 컷-스루 스위치와 기존의 컷-스루 스위치의 성능을 비교 평가하였다. 이

장에서는 실험 결과를 비교 분석한다.

본 논문에서는 Scientific and Engineering Software Inc. 사의 SES/Workbench를 사용하여 확률에 의한 이산 사건 모델링(discrete event modeling)으로 시뮬레이션을 수행하였다.[8]

제안하는 스위치의 구조에 대한 성능 평가 기준으로는 처리율과 지연 시간이 있겠다. 지연 시간은 하나의 메시지가 입력 포트에서 들어와서 출력 포트로 정상적으로 도달하는데 걸리는 평균 시간을 의미한다. 처리율은 입력 포트에 도달한 전체 메시지중 일정 시간 동안 출력 포트에 도착한 메시지의 비율을 나타낸다.

처리율(throughput) = 성공한 패킷 / 전체 패킷
평균 지연 시간(average latency) = 성공한 패킷들의 지연 시간의 평균.

4.1 시뮬레이션 수행을 위한 가정

시뮬레이션을 수행하는 스위치의 크기는 16x16을 가정한다. 이것은 앞서 가정한 인트라스위치 네트워크에 근거한다. 즉, 하나의 스위치만을 사용해서 지역네트워크을 구성한다는 가정이다.

각각의 스위치 원소에서의 버퍼의 크기는 2바이트이다. 메시지 패킷의 크기는 8바이트, 16바이트, 64바이트의 세가지 종류가 있다. 입력 포트에 도달하는 패킷의 도달 시간 간격은 지수 분포에 따른다. 스위치의 로드란 전체 스위치의 입력 포트들중 몇 개의 포트에서 메시지가 발생되었는지를 말한다. 즉, 로드가 50%라고 하면 전체 입력 포트중 반이 동시에 메시지를 발생하였다는 것이다.

4.2 시뮬레이션 결과 분석

스위치로 들어오는 메시지의 크기를 변화하면서 수행하였다. 결과는 처리율과 평균 지연 시간에 관한 비교를 하였다. 시간 단위는 us이다.

스위치에 들어오는 메시지들은 모두 멀티캐스트 메시지일 때와 멀티캐스트 메시지와 유니캐스 메시지가 비율이 50:50일때를 가정하였다. 멀티캐스트 메시지들은 적어도 2개의 목적지를 갖고 라우팅을 하게 된다. 목적지는 모두 다른 출력 포트를 요구한다.

메시지의 크기는 8바이트, 16바이트, 64바이트일 때를 가정하였다. 먼저 16x16 스위치에서 가장 작은 메시지 크기인 8바이트일 때 비교하였다. <그림 9>는 처리율이다.

공유 버스를 사용하는 멀티캐스트 컷-스루 스위치의 경우에 로드가 75%에 이르러서 처리율이 떨어지는 것을 볼 수 있다. 그러나 로드가 낮은 경우에는 여전히 100%에 가까운 높은 처리율을 보이고 있다. 기존 방법의 컷-스루 스위치 형식에서는 20%이하의 매우 낮은 처리율을 보였다. 평균적으로 제안하는 스위치 방법이 기존의 것보다 10배정도의 처리율의 향상을 얻을 수 있었다.

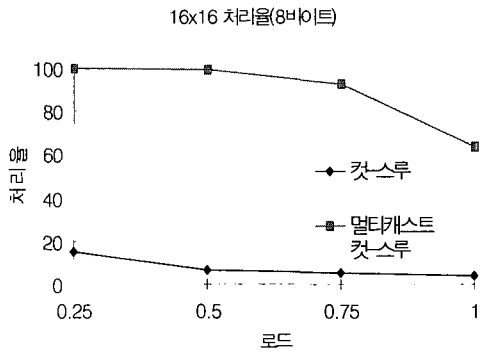

그림 9 처리율의 비교(16x16스위치 메시지크기=8바이트)

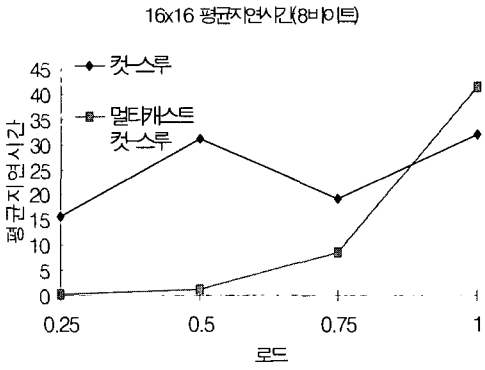

그림 10 평균 지연 시간의 비교(16x16스위치 메시지크기=8바이트)

<그림 10>은 같은 환경에서의 평균 지연 시간의 비교이다. 공유 버스를 사용하는 멀티캐스트 컷-스루 스위치의 경우에는 로드가 작을 경우에는 큰 변화가 없었으나 로드가 높아짐에 따라 평균 지연 시간이 급격히 증가함을 볼 수 있다. 로드가 1일 때에는 기존의 것보다도 높게 나왔다. 그것은 평균 지연 시간은 라우팅이 성공하여서 출력 포트에 도달한 메시지들의 처리 시간을 비교하는 것이기 때문이다. 또한 공유 버스를 사용하는 경우

에 버스를 통한 메시지의 전송은 기존의 방법에 비해서 월등히 빠르나 두 개의 공유 버스를 사용하므로 상대적으로 라우팅 하지 못하고 기다리는 메시지가 많게 된다. 이러한 메시지들이 많고 또한 처리율이 높으므로 평균적으로 증가하는 것을 얻을 수 있었다. 또한 파이프라인 형식으로 전송하는 기존의 방식이 처리율이 높다면 처리 시간에서는 이득이 될 수 있음을 알 수 있다.

다음은 메시지의 크기를 16바이트로 증가시켰을 때 처리율과 평균 지연 시간의 비교이다.

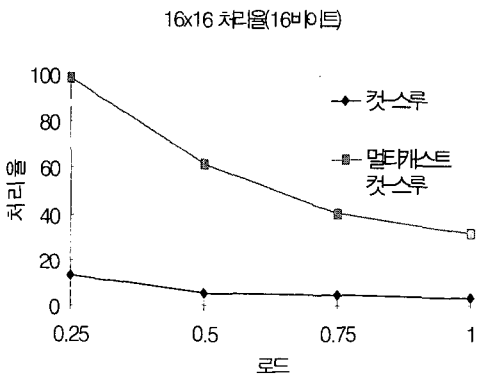

그림 11 처리율의 비교(16x16스위치 메시지크기=16바이트)

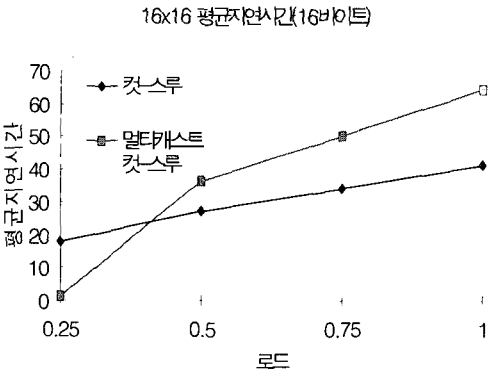

그림 12 평균 지연 시간의 비교(16x16스위치 메시지크기=16바이트)

<그림 11>은 처리율의 비교이다. 스위치의 로드가 증가함에 따라서 처리율의 감소를 얻을 수 있다. 로드가 50%를 넘으면서 공유 버스를 사용하는 멀티캐스트 컷-스루 스위치에서 처리율의 감소를 보였다. 기존의 방식은 로드가 높아짐에 따라서 큰 변화는 없이 매우 낮은 처리율을 보였다. 스위치의 크기가 증가하고 메시지의 크기 역시 증가함에 따라서, 기존의 것과 제안하는 것과의 차이가 가장 뚜렷하였다. 즉, 평균적으로 약 7배 정도의 처리율의 향상을 얻을 수 있었다.

<그림 12>는 평균 지연 시간의 비교이다. 처리율과 관련이 되어서 로드가 50%를 넘으면서 공유 버스를 사용하는 멀티캐스트 컷-스루 스위치 방식에서 평균 지연 시간이 다소 높음을 볼 수 있다. 기존의 것과의 비교에서도 약 26%정도 평균 지연 시간에서 약간 증가함을 볼 수 있었다. 역시 <그림 10>에서와 같은 이유이다.

즉, 처리율의 증가에 따라서 보다 많은 메시지들의 대기 시간이 계산되어서 평균 처리 시간이 기존의 것보다 높게 나왔다고 볼 수 있다.

다음은 메시지의 크기를 최대 크기인 64바이트로 가정하고 실험을 하였다. <그림 13>은 처리율이다. 처리율에 있어서 두 가지 방식 모두 급격한 감소를 보이고 있다. 공유 버스를 사용하는 멀티캐스트 컷-스루 스위치의 경우에는 로드가 적을 경우에는 높은 처리율을 보였다. 그러나 로드가 25%에 도달하면서 급격히 감소를 하여서 로드가 높아짐에 따라서 그 다음부터는 큰 변화는 없이 다소 낮은 처리율을 보이고 있다. 두 가지 경우의 비교에 있어서는 평균적으로 약 2배 정도의 처리율의 향상을 얻을 수 있었다.

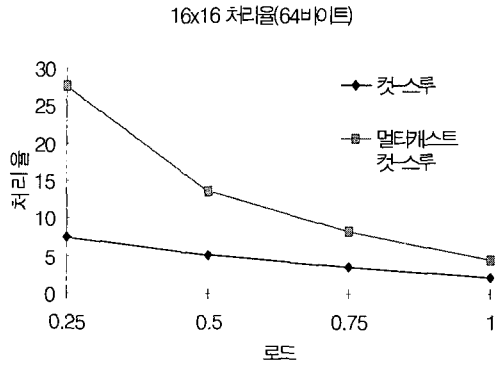

그림 13 처리율의 비교(16x16스위치 메시지크기=64바이트)

<그림 14>는 같은 환경에서의 평균 지연 시간의 비

교이다. 로드가 증가함에 따라서 공유 버스를 사용하는 멀티캐스트 컷-스루 스위치가 1.5배 정도 증가함을 볼 수 있다. 처리율의 감소와 함께 성공한 메시지들의 처리 시간 역시 매우 높음을 알 수 있었다.

이상은 일정한 크기의 스위치에서 멀티캐스트 메시지의 크기를 변화하면서 실험 평가 한 것이다. 다음은 메시지의 종류가 오직 멀티캐스트일 때가 아닌 , 멀티캐스트 메시지와 유니캐스트 메시지가 혼합된 형태이다. 즉, 멀티캐스트 메시지와 유니캐스트 메시지의 비율을 50:50으로 하였다. 따라서 입력 포트에서 발생되는 메시지들의 반은 멀티캐스트 메시지를 갖고 나머지 반은 유니캐스트 메시지를 갖게 된다. 이 때 처리율과 평균 지연 시간의 비교를 하였다. 멀티캐스트 메시지의 라우팅 방법은 앞에서 제안한 방법과 동일하다. 유니캐스트의 경우에는 기존의 방식을 그대로 모두 사용하였다. 즉, 기존의 XY라우팅을 사용하여서 라우팅하는 방법을 두 가지 방식 모두에서 사용하였다.

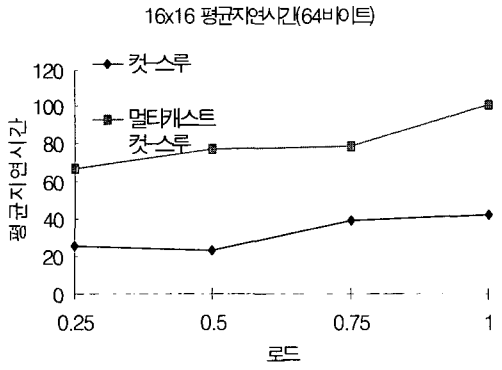

그림 14 평균 지연 시간의 비교(16x16스위치 메시지크기=64바이트)

이번에는 스위치의 크기에 따라서 구분하고 각각의 스위치에서 메시지 크기의 변화에 따른 결과를 한 번에 볼 수 있다.

다음은 먼저 8x8스위치에서 멀티캐스트 메시지와 유니캐스트 메시지의 비율이 50:50일 때 얻은 처리율과 평균 지연 시간이다.

먼저 메시지의 크기가 8바이트와 16바이트일 때 처리율이 <그림 15>와 같다.

공유 버스를 사용하는 멀티캐스트 컷-스루 스위치의 경우에는 메시지의 크기에 상관없이 100% 가까운 처리율을 얻을 수 있었다. 8바이트와 16바이트일 때 각각 1.4배와 3배의 처리율의 향상을 얻었다.

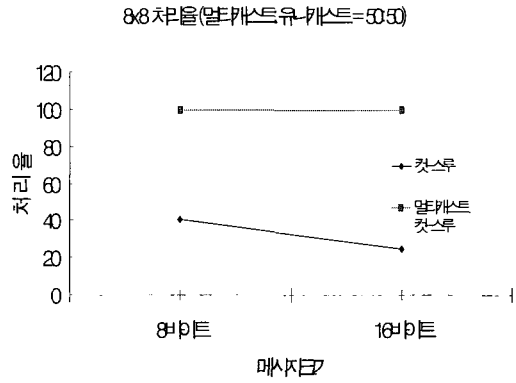

그림 15 처리율의 비교(멀티캐스트:유니캐스트=50:50)

<그림 16>은 같은 환경에서의 평균 지연 시간의 비교이다. 역시 2배 정도의 평균 지연 시간의 감소를 제안하는 방식에서 얻을 수 있었다.

다음은 스위치의 크기를 16x16으로 하고 8바이트, 16바이트 그리고 64바이트의 메시지에 대한 비교이다.

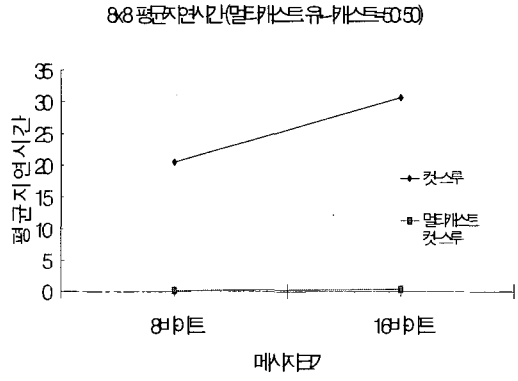

그림 16 평균 지연 시간의 비교(멀티캐스트:유니캐스트=50:50)

<그림 17>은 처리율의 비교이다. 8바이트, 16바이트 , 64바이트 일 때 각각 9배, 6배, 2배의 처리율의 향상을 얻을 수 있었다. 메시지의 크기가 커질 때 최소 2배의 처리율 향상을 얻을 수 있었다.

<그림 18>은 같은 환경에서의 평균 지연 시간의 비교이다. 8바이트일 때에서 76%의 감소를 보이나, 16바이트일 때에는 50%의 증가를 , 64바이트일 때에는 60% 정도 평균 지연 시간이 증가 하였다.

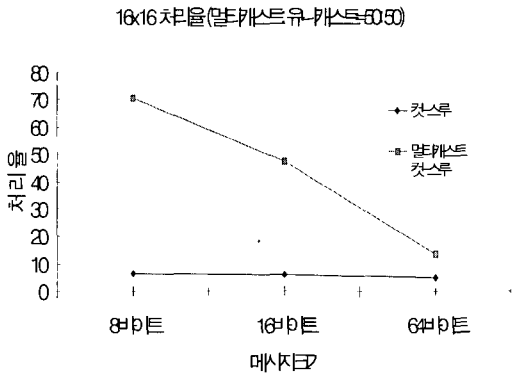

그림 17 처리율의 비교(멀티캐스트:유니캐스트=50:50)

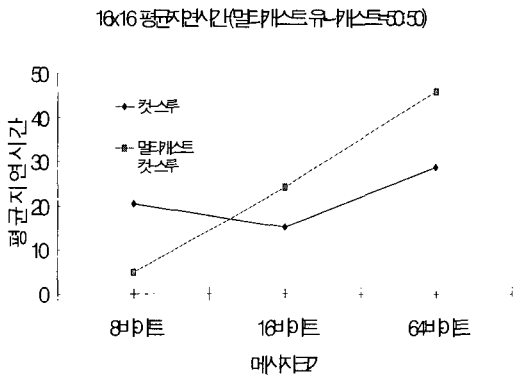

그림 18 평균 지연 시간의 비교(멀티캐스트:유니캐스트=50:50)

5. 결 론

멀티캐스트 통신에 대한 중요성이 높아지고 있다. 그것은 비디오 회의, 분산 가상 현실등의 네트워크 어플리케이션에 있어서 기본적인 서비스가 되어 가고 있는 것에 기인한다. 그에 따라 효과적인 멀티캐스트를 지원하는 스위치에 대한 관심이 증대되고 있다. 컷-스루 스위치 기술은 하드웨어 멀티캐스트를 좀 더 복잡하게 한다는 단점을 가지고 있다. 그것은 유니캐스트 통신에 비해서 데드락을 피하기가 훨씬 어렵기 때문이다.

본 논문에서는 기존의 컷-스루 스위치의 구조를 유지하면서 , 좀 더 효과적인 멀티캐스팅을 위해서 공유 버스를 사용하는 스위치를 제안하였다. 공유 버스를 사용하는 방법은 처리율에 있어서 월등한 성능 향상을 얻을 수가 있다. 그것은 공유 매체를 사용하는 방법이 물리적으로 멀티캐스팅을 쉽게 할 수 있다는 장점에 근거한다. 실험 결과는 그러한 성능 향상을 보여주고 있다. 스위치의 크기와 메시지의 크기의 변화에 따라서 멀티캐스트 컷-스루 스위치는 처리율에 있어서는 최소 2배에서 최대 10배정도의 향상을 얻을 수 있었다. 처리 시간에 있어서는 2배 정도의 감소에서 스위치의 크기와 메시지의 크기가 동시에 커지면 2배 정도 증가하는 결과를 보였다.

다중 멀티캐스트에 대한 고려를 본 논문에서는 하지 않았다. 그것은 처리 시간의 단축과 연관될 수 있다. 따라서 이 부분에 대한 연구가 더 진행되어야 할 것으로 본다.

참 고 문 헌

[1] M. Yang and L.M. Ni, "Design of scalable and multicast capable cut-through switches for high-speed lans," *Proceedings of the 1997 ICPP*, pp.324-332, Aug. 1997.

[2] N.J. Boden and et al., "Myrinet - a Gigabit-per-second local area network," *IEEE Micro*, vol.15, pp.29-36, Feb. 1995

[3] L.M. Ni ,"Should scalable parallel computers support efficient hardware multicast?," *Proc. of the 1995 ICPP workshop on challenges for Parallel Processing*, pp.2-7, Aug. 1995

[4] Myricom, "Myrinet link specfication," *http://www.myri.com/products/docu- mentation/link/.*

[5] D. Cohen and G. Finn, "ATOMIC: A low-cost, very-high-speed, local communication architecture," *Proceedings of the 1993 International Conference on Parallel Processing*, vol. 1, pp. 39 - 46, Aug. 1993

[6] L.L. Peterson and B.S. Davie, *Computer networks : a system approach*, Morgan Kaufmann 1996

[7] C. Partridge, *Gigabit Networking*, Addison-Wesley 1993.

[8] *SES/Workbench* Rel. 3.0, Scientific and Engineering Software, Inc., 1995

백 정 민

1997년 2월 서강대학교 전자계산학과 공학사. 1999년 2월 서강대학교 컴퓨터학과 공학석사. 1999년 3월 ~ 현재 삼성전자 정보통신총괄 네트웍 사업부 연구원.

김 성 천

1975년 서울대학교 공과대학 공업교육학(전기전공)학사. 1979년 Wayne State Univ. 컴퓨터공학 공학석사. 1982년 Wayne State Univ. 컴퓨터공학 공학박사. 1982년 ~ 1984년 캘리포니아주립대 조교수, 1984년 ~ 1985년 금성반도체(주) 책임연구원. 1986년 ~ 1989년 서강대학교 공과대학 전자계산소 부소장. 1989년 ~ 1991년 서강대학교 공과대학 전자계산학과 학과장. 1985년 ~ 현재 서강대학교 공과대학 전자계산학과 교수(1992.9 ~ 현재). 1989년 ~ 현재 한국정보과학회 병렬처리시스템 연구회 부위원장(1989~1993), 위원장(1994~1997). 대한전자공학회 및 한국통신학회 논문지 편집위원(1991~현재, 1993~현재). 관심분야는 병렬처리시스템(Parallel Computer Architecture, Interconnection Network)