고속 라우터에 대한 고찰(Ⅱ) - STC104의 망 구성에 따른 성능분석

이 효 종†

요 약

본 논문에서는 '고속 라우터에 대한 고찰(I)' 논문에서 언급한 고속 라우터 스위치의 하드웨어 구조를 사건 중심의 시뮬레이터를 이용하여 망을 구성하고, 메시지 생성 율을 변화시키면서 각 망의 throughput, latency 및 패킷 정체시간을 측정하여 다양한 환경에서의 망의 성능을 분석하였다. 또한 점대점 통신만을 할 수 있도록 제작된 STC104를 이용하여 멀티캐스트를 지원할 수 있도록 변환된 U-mesh와 U-torus의 알고리즘의 제안에 따라 그물구조와 원환체 구조의 STC104망에 적용하여 실행하고 성능을 분석하였다. 그물구조, 원환체, 그리고 초입방체의 순서로 throughput은 향상되었으며 높은 메시지가 생성될 때에 성능의 차이가 두드러졌다. 망의 latency와 정체시간은 반대의 순서로 증가하며, 역시 높은 메시지가 생성될 때에 높은 latency를 보여주었다. 소프트웨어 멀티캐스트는 U-mesh와 U-torus가 비슷한 throughput을 보여주었으나, U-mesh가 약간 우수하였다. 90개의 목적지 노드에 대하여 개별 전송 방법으로 멀티캐스트를 구현한 경우보다 U-mesh와 U-torus는 각각 8~10배의 throughput을 보여주었다. 다중링크 환경은 유휴 링크를 활용하기 때문에 단일링크 환경보다 우수한 결과를 나타냈다.

Study on High Speed Routers(II) - Performance Analysis on Various Network Topology of STC104

Hyo Jong Lee†

ABSTRACT

A simulation package has been developed as an event-driven system that can handle the hardware configuration of STC104 and algorithm proposed in the sister paper of 'Study on High Speed Routers(II).' After various STC104 topology of meshes, torus, and hypercubes are constructed using up to 512 switches, the performances of each topology has been analyzed under different message generation rate in terms of throughputs, latency, and packet blocking time. Modified multicast algorithms for STC104 have also been proposed for STC104 after U-mesh and U-torus in order to overcome the multicasting difficulty because of the point-to-point communication method found in STC104. The performance of the multicast algorithms have been analyzed over meshes and torus configuration. Throughput gets higher in the order of mesh, torus, and hypercube. Throughput difference among topology were distinctive in the zone of high message generation rate. Latency and blocking time increased in the order of hypercube, torus, and mesh. U-mesh and U-torus of software multicast showed similar throughput, however, U-mesh performed slightly better result. These algorithms showed eight to ten times better results compared to individual message pass for 90 destination nodes. Multi-link environment also showed better performance than single-link environment because multi-link network used the extra links for communication.

키워드 : STC104, 멀티캐스트(multicast), throughput, 네트워크 토폴로지(network topology)

1. 서 론

정보화 사회에서 급속하게 증가하는 문자, 음성, 그래픽 및 동화상 형태의 대용량 정보를 고속으로 처리할 수 있는 기술의 요구가 늘어나고 있음을 본 논문의 자매 논문인 '고속 라우터에 대한 고찰(I)(이하 논문 I로 칭함)'에서 기술하였다. 언급한 논문에서는 고속 라우팅 칩의 하나인 STC104의 구조를 분석하고 고속 라우터 스위치가 지니고 있는 웜홀 라우팅, 구간 레이블링 및 적응 라우팅 방법에 알맞은 패킷 전송 알고리즘을 제안하였다.

본 논문에서는 STC104로 구성된 그물, 원환체 및 초입방체 망의 구조를 네트워크 시뮬레이터로 구현하여 성능을 분석하고 그 결과를 기술하였다. 한 개의 발원지에서 다수의 목적지로 데이터를 전송하는 멀티캐스트의 통신 방법은

† 종신회원 : 전북대학교 교수 · 공업기술연구소
논문접수 : 2000년 7월 10일, 심사완료 : 2001년 5월 29일

영상의 전달이나 병렬 프로세서 망 등에서 절실히 필요하다. 따라서, 논문 I에서 제안된 멀티캐스트의 알고리즘을 네트워크 시뮬레이터로 구현하고 그 성능도 평가하였다.

본 논문의 구성으로서 제2장에서는 이론적 설명에 기반을 두고 STC104를 컴퓨터 시뮬레이션을 통해 구현하고 임의의 특성을 가진 통신 환경을 만들기 위한 터미널 프로세서의 구현 방법에 대해 기술하였다. 제3장에서는 한개의 링크를 사용하여 망을 구성한 경우에서의 각 구조별 특성을 알아보고 복수 링크를 사용하여 망을 구성한 경우의 각 구조별 특성을 고찰한 후 성능을 분석하였다. 제4장에서는 범용망에서 중요한 요소인 멀티캐스트 구현을 위하여 앞에서 소개한 멀티캐스트 알고리즘을 소프트웨어적으로 구현하고 이를 STC104로 구성된 망에서 실험하여 그 성능을 분석하였다. 마지막으로 제5장에서 총괄적인 결론을 기술하였다.

2. 시뮬레이션 모델링

시뮬레이션 모델링은 일반적으로 컴퓨터에 의한 구조적 모델을 세워 성능을 측정하는 방법과 queueing 모델에 근거하는 확률론적 방법[13, 14]으로 나뉜다. STC104의 구조가 queueing 모델로 표현하기에는 다소 복잡하며 메시지의 성능을 position 분포의 형태보다도 인위적으로 최대 through put을 알기 위해 제어할 필요가 있어서 본 논문에서는 컴퓨터 시뮬레이션에 의한 접근 방법을 취하였다.

2.1 시뮬레이션 라이브러리

라우터의 특성 및 기능을 시뮬레이션하기 위해 RICE University의 NETSIM 라이브러리[15, 16]를 이용하여 시뮬레이션 소자를 구성하였다. NETSIM은 C 프로그래밍언어를 이용하는 이산 사건 시뮬레이션 라이브러리 YACSIM [16]에 그 기반을 둔 범용 상호 연결망의 시뮬레이터로 직접 네트워크와 간접 네트워크를 모두 시뮬레이션 하는데 이용되어질 수 있다. NETSIM의 두드러진 특징으로는 사용자가 라이브러리에서 제공하는 기본적인 네트워크 소자를 연결시켜 가상으로 임의의 네트워크를 구성할 수 있도록 모듈화 되어 있다는 점이며, 웜홀 라우팅과 virtual-cut-through 라우팅과 같은 다양한 라우팅 기술을 사용하는 네트워크를 시뮬레이션 할 수 있는 기능을 지니고 있다. NETSIM은 또한 시뮬레이션 프로그램이 실행되는 완전한 병렬 시스템의 시뮬레이션을 위한 시뮬레이션 시험대의 구성 요소로 사용되어질 수 있으며 또한 패킷을 임의로 생성시켜 네트워크로 보내는 네트워크 전용의 시뮬레이터로서도 사용될 수 있다. NETSIM에서는 많은 스위치 소자 형태를 지원하기보다는 다양한 형태의 스위치를 구성할 수 있도록 스위치보다 하위에 있는 multiplexer, demultiplexer, buffer, 또는 network port와 같은 기본 네트워크 소자를 제공한다. 사용자들은 이들을 호출하여 다양한 형태의 망을 구성할 수 있다.

2.2 STC104 네트워크 구성 모델

2.2.1 STC104 시뮬레이션 소자의 구현

STC104에 대한 시뮬레이션 소자는 논문 I의 제2절에서 언급했던 링크 부와 버퍼, 크로스바 스위치와 같은 STC104의 기능 구조와 특징을 기반으로 하여 구현하였다. 시뮬레이션 소자 개발시 각 링크와 요소에서 일어나는 오류에 대한 처리 및 전체 소자의 제어 기능을 가지는 STC104의 제어부는 소프트웨어 시뮬레이션 환경에서 제어 기능을 필요로 하는 에러, 즉 패러티 에러, 패킷크기 에러, 잘못된 주소를 갖는 패킷으로 인한 에러는 일어나지 않음을 가정하였다.

네트워크를 통과하는 패킷에 대한 소자의 영향은 패킷의 전달시 지연시간인 latency로 표현될 수 있다. STC104에서는 두 개의 서로다른 클럭에 의해 latency가 결정되며 코어는 보통 25, 30, 40, 50MHz중에서 선택되어 고정된 값으로 사용된다. 시뮬레이션 소자에서 시스템 사이클은 칩의 코어에 적용되는 클록 사이클을 의미하며 링크 사이클(link cycle)은 DS-Link의 동작 사이클을 의미한다. 본 연구에서는 시스템 사이클과 링크 사이클로 각각 25MHz와 100Hz의 클록을 설정하였다.

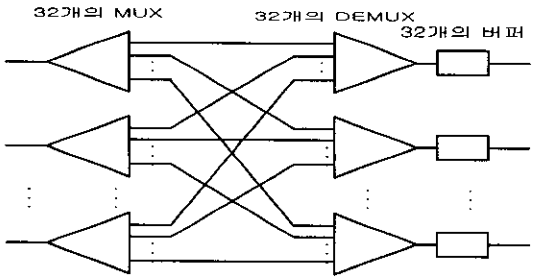
시스템의 latency를 계산하기 위한 전체 구성 요소별로 소요되는 클럭 사이클은 <표 1>과 같다. 따라서, STC104를 통한 패킷이 통과할 때의 전체 latency는 14시스템 사이클 + 39 링크 사이클이 된다.

〈표 1〉 구성 요소별 소요 클럭 수

DS-Link입력부	4 system + 14 link cycles
링크부의 입력측 FIFO	1 system cycles
토큰 류	4 system cycles
크로스바 스위치	3 system cycles
링크부의 출력측 FIFO	1 system cycles
DS-Link 출력부	4 system + 22 link cycles

2.2.2 크로스바 스위치의 구현

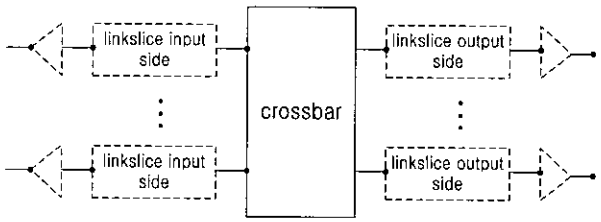
크로스바 스위치는 32개의 입력부에서 들어오는 패킷의 헤더로부터 출력할 링크를 결정하고 이에 따라 각 패킷을 해당 링크로 보내주는 기능을 갖는다. STC104 내부에서 크로스바 스위치는 동시에 32개의 경로를 설정할 수 있는 능력을 지니며 경로 결정에 걸리는 시간은 1 시스템 사이클 정도이다. 크로스바 스위치의 구현은 입력이 32개의 출력을 가질 수 있다는 점에 착안하여 하나의 입력으로부터 정보를 분배하는 역할을 하는 demultiplexer로 각각의 입력 단을 구성하였다. 또한, 크로스바 내의 버퍼를 구현하기 위하여 32개의 버퍼를 각각 multiplexer에 연결하였다. (그림 1)은 NETSIM의 기본 소자로 구성된 크로스바를 보여주고 있다.



(그림 1) 크로스바 스위치의 구성

2.2.3 STC104의 구현

STC104의 구현은 앞서 설명한 크로스바, 링크부, 입력포트 및 출력포트로 하며 각 링크에 대한 구간 라우팅을 위한 구간값을 저장하는 구간표로 구현되는데 (그림 2)와 (그림 3)에 그 구성과 소자의 정의를 나타내었다. (그림 3)에서 레이블링은 STC104의 링크당 하나씩 대응되는 구간 표를 정의한 것이며 mode는 만일 STC104의 레이블링이 그룹적응 라우팅이 적용된 경우 1의 값을 갖는다. inuseport와 outuseport는 입·출력포트와 링크부의 동적 할당을 위한 변수로 각각 1에서 32까지의 입·출력포트에 대응하여 해당 포트의 사용 여부를 나타낸다. STC104와 STC104를 연결시키는 경우 NETSIM의 소자 특성 정의에 따를 때 출력포트와 입력포트의 연결은 허용하지 않는다. 그래서 여기에서는 이의 구현을 위해 STC104의 출력단과 STC104의 입력단을 연결하는 DS-Link를 모의하는 일종의 링크 동작 모의 프로세서를 (그림 4)와 같이 정의하였다.



(그림 2) STC104의 구현

```
Module STC104
  link       : 32 pointer ARRAY of Linkslice
  inport     : 32 pointer ARRAY of input port
  outport    : 32 pointer ARRAY of output port
  csb        : crossbar
  mode       : grouped adaptive mode flag
  inuseport  : input port use flag
  outuseport : output port use flag
  label      : interval labeling table
```

(그림 3) STC104 소자의 정의

```
Process DS_LinkProcess
  VAR :
    opt : Output port of STC104₁
    ipt : Input port of STC104₂
    pkt : received packet
  DEFINE :
    DS_OUT_LATENCY : DS-Link output latency
    DS_IN_LATENCY : DS-Link input latency
    FUNCTION IPortSpace(VAR input port)
        return free buffer space of input port
    FUNCTION OPortSpace(VAR output port)
        return the number of
                waiting packet at output port
    PROCEDURE SetDelay(VAR delay time)
        set the processing delay
    FUNCTION PacketReceive(VAR output port)
        return packet from output port
    PROCEDURE PacketSend(VAR input port, packet)
        send the packet to input port
    PROCEDURE SemaphoreWait(VAR semaphore)
        decrement the semaphore's value and the process
        continue executing until the value is zero
    BEGIN
        WHILE (1)
          if (IPortSpace(ipt) > 0) then
            if (OPortPackets(opt) > 0) then
              SetDelay(DS_OUT_LATENCY)
              pkt = PacketReceive(opt)
              SetDelay(DS_IN_LATENCY)
              PacketSend(ipt, pkt)
            else SemaphoreWait(semaphore of opt)
          else SemaphoreWait(semaphore of ipt)
        END
    END
```

(그림 4) 상호 연결을 위한 DS-Link 모의 소자의 정의

2.3 터미널 프로세서의 구현 한

실제 응용되는 네트워크에서는 라우터 스위치에 처리 능력을 가지는 프로세서나 다른 장치와의 연결 소자(interface module)를 부착하게 된다. 이러한 프로세서들은 사용자들의 요구에 따라 패킷을 네트워크로 보내거나 또는 네트워크로부터 패킷을 받아들이는데 이러한 활동은 전체 네트워크 측면에서 볼 때 특정 규칙이 없이 임의로 이루어지고 있다. 때문에 이를 통해서는 통계적인 성능의 측정이 어려우며 실제로 그러한 임의로 동작하는 프로세서들을 모의한다는 것 역시 어렵다. 따라서 본 논문에서는 성능 측정을 원하는 환경에 따라 일률적인 특성을 지니는 모의 프로세서를 설계하였다.

설계된 프로세서는 먼저 지정된 메시지 발생률을 가지고 네트워크상의 임의의 노드로 패킷을 보내는 기능을 가지는 프로세서와 연결된 네트워크로부터 패킷을 받아들이는 수신 프로세서 그리고 STC104 망에서의 멀티캐스트를 위한 알고리즘을 구현한 멀티캐스트 프로세서들이다.

2.3.1 메시지 발생 프로세서의 구현

메시지의 발생 방법은 지정된 비율로 메시지를 발생시키는 방법과 발생이 가능한 때면 언제나 발생시키는 연속적인 메시지의 발생 비율을 인자로 결정하였다. 발생 지연 시

간은 입력 포트의 차단(block)여부에 관계없이 메시지 발생 비율에 따라 결정된다. 입력 포트가 사용 가능할 때 지속적으로 메시지를 보내는 연속 발생 프로세서는 입력 포트 다음에서 패킷의 정체가 일어나더라도 관계없이 메시지를 보낸다. (그림 5)는 메시지 발생 프로세서를 나타낸 것이다.

```
Process RMGProcess        random message generating process
  VAR :
     desn          : integer that indicate destination node address
     pkt           : packet
     mpr           : time between passing two message
     no-of-message : the number of messages to pass
     nodesize      : the number of all nodes
     ipt           : input port to send the packet
  DEFINE :
     FUNCTION NewPacket
        return a new packet
     PROCEDURE PacketSend(VAR input port, packet)
        set the packet to input port
     PROCEDURE WaitProcess(VAR time)
        wait the process for time
  BEGIN
     FOR I = 1 TO no-of-message DO
          pkt = NewPacket
          desn = random() % nodesize
          PacketSend(ipt, pkt)
          WaitProcess(mpr)
     END
  END
```

(그림 5) 메시지 발생 프로세서의 정의

2.3.2 메시지 수신 프로세서의 구현

일반적으로 실제 네트워크에서는 메시지를 보내는 프로세서와 받는 프로세서가 동일한 프로세서에 의해 수행되며 대

```
Process GRProcess      general purpose message receiver
 VAR :
   pkt        : packet
   opt        : output port to receive the packet
DEFINE :
   FUNCTION OPortPackets(VAR output port)
     return the number of waiting packet at output port
   FUNCTION PacketReceive(VAR output port)
     return packet from output port
   PROCEDURE PacketFree(VAR pkt)
     delete packet
   PROCEDURE SemaphoreWait(VAR semaphore)
     decrement the semaphore's value and the process continue
     executing until the value is zero
BEGIN
    WHILE (1)
       if (OPortPackets(opt) >0) then
          pkt = PacketREceive(opt)
          PacketFree(pkt)
       else SemaphoreWait(semaphore of opt)
    END
END
```

(그림 6) 메시지 수신 프로세서의 정의

개의 통신에 관련된 종작이 수신된 메시지에 따라 그에 대한 응답으로 메시지를 보내거나 송신한 메시지의 응답으로서의 메시지를 수신한다. 그러나, 그러한 동작은 통신 응용에 관계되는 부분으로 여기에서는 그에대한 고려는 제외하고 메시지의 단순한 송신과 수신에만 초점을 두었으며 이러한 관점에서 메시지의 수신이 메시지의 송신 동작에 따른 변화 특성은 배제하였다.

메시지 수신 프로세서의 구현은 메시지 송신과는 독립적으로 구성되며 일종의 프로세서 소자로 구현하였다. 메시지 수신 프로세서의 동작은 프로세서가 연결된 STC014의 출력 포트를 검사하여 해당출력 포트에 들어오거나 대기중일 경우 바로 패킷을 받아들여 네트워크로부터 패킷을 제거하는 역할을 한다. (그림 6)은 수신 프로세서의 시뮬레이션 정의이다.

2.3.3 멀티캐스트 프로세서의 구현

멀티캐스트 프로세서는 논문 I의 3.4절에서 언급했던 구조별 멀티캐스트 알고리즘을 기반으로 구현하였다. 멀티캐스트 트리를 몇 개의 부분 집합으로 나누고 이를 데이터와 함께 메시지로 구성하여 목적 노드로 전송하는 프로세서를 형성하며, 목적 노드에서는 발원 노드로부터 전달된 멀티캐스트 메시지로부터 자신이 보낼 부분 집합을 추출하여 더 이상 보낼 노드가 남아있지 않을 때까지 그 과정을 되풀이 하도록 하였다. 이에 대한 구조별 알고리즘은 앞장에서 이미 설명한 것과 동일하며 궁극적으로 각각의 발원 노드에서 목적 노드로의 전송은 점대점 통신에서 구현된 프로세서들을 그대로 적용하였다.

3. 구조별 성능 실험 및 고찰

앞 절에서 설명했던 네트워크의 구성 방법과 시뮬레이션 소자 모형에 따라 개발된 시뮬레이션 소프트웨어를 이용하여 그물, 원환체, 및 초입방체 구조를 구성하고 각각에 대해 성능을 실험하고 그 결과를 고찰하였다.

네트워크의 성능 측정은 네트워크의 throughput, 메시지 전송 시간을 나타내는 latency, 그리고 망의 통신 패턴의 상태를 나타내는 패킷 정체(blocking)시간으로 나타낼 수 있다. 성능 측정에 관련된 변수들은 다음과 같이 정의한다.

① Throughput : 정해진 메시지 발생률로 메시지를 보낼 때 실제로 수신단에 받아들여지는 메시지 양

② Latency : 패킷의 헤더가 네트워크로 들어선 순간부터 패킷의 꼬리가 네트워크로부터 빠져 나올 때까지 소요되는 시간

③ 패킷 정체 시간 : 네트워크에서 패킷이 다른 버퍼나 다른 소자로 이동하려 할 때 해당 버퍼나 소자가 다른 패킷에 의해 사용되어 패킷이 전송되지 않고 현재의 장소에 머물러있는 시간들의 축적된 값

메시지와 패킷은 동일한 크기로 결정했으며 메시지의

발생률은 초당 바이트 단위의 데이터 발생률로 정하였다. 실험 결과에 사용되는 throughput, latency, 패킷 정체 시간은 발생된 모든 패킷에 대해 얻어지는 값의 평균을 이용한다. 통신 트래픽 패턴은 임의로 목적지를 결정하는 메시지 발생 프로세서를 사용하여 랜덤 트래픽 상황으로 설정하였다.

3.1 다중 링크 환경의 실험 및 고찰

STC104는 32개의 양방향 링크를 가지고 있기 때문에 상호 연결에 하나씩의 링크만을 사용하면 연결되지 않은 링크들은 통신에 사용되지 않기 때문에 최대 효과를 낼 수가 없다. 즉, 하나의 터미널 노드에서 다른 터미널 노드로 가기 위해 통과할 수 있는 링크의 수는 원환체의 경우 4개, 6차 초입방체의 경우 6개 등으로 서로 불평등한 조건이 형성되고 통신망간의 정확한 평가를 할 수 없다. 따라서, 각 망 구조의 노드를 구성하는 STC104 라우팅 스위치에 한 개의 메시지 발생 프로세서와 메시지 수신 프로세서를 연결하고 스위치와 스위치간의 상호 연결에는 프로세서의 연결에 사용되지 않은 모든 링크를 연결되는 각 스위치에 대해 공평하게 분배하여 할당하는 다중 링크 환경(MLE : multiple links environment)으로 설정하였다. 구조별로 연결되는 각 스위치에 분배되는 링크의 수는 스위치당 터미널 노드의 수를 a라 할 때 링의 경우에는 (32-a)/2, 그물 구조와 원환체는 (32-a)/4, 초입방체는 차원수 n에 따라 다르며 일반적으로 (32-a)/n으로 결정된다. 예를 들어 3-초입방체의 경우에는 연결되는 스위치 하나 당 10개의 다중 링크를 사용한다. 라우터와 스위치당 터미널 수와 연결 링크의 관계는 <표 2>와 같이 요약할 수 있다.

실험은 메시지의 발생률을 조절하면서 수행하였으며 실험을 통해 얻어낸 throughput과 latency, 패킷 정체 시간을 통해 각 망간의 성능을 비교하였다. 실험에 사용된 망은 2차원 그물, 2차원 일환체 및 초입방체 구조이며 스위치의 수는 64, 128, 256, 512개로 변환시키며 실험하였다.

〈표 2〉 망구조 구성 변수

구 조	표 현	의 미
그 물	(k, a, w)	k : k*k개의 STC104로 구성 a : 스위치당 터미널 수 w : 상호 연결 링크수 n : 초입방체의 차원(dimension)
원환체	(k, a, w)	
초입방체	(n, a, w)	

스위치당 터미널 수는 하나로 제한하고 메시지의 발생률은 1MByte/s에서부터 4MByte/s까지 변화시켰다. 서로 연결되는 스위치간의 다중링크의 수는 구조마다 최적의 조건을 가지도록 <표 3>와 같이 지정하였다.

〈표 3〉 실험 표본 집합

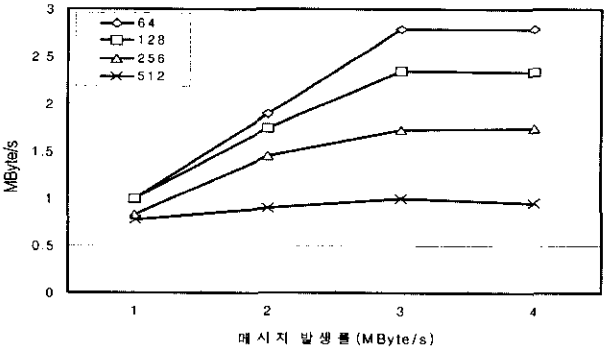
구 조	Network cost (STC104 수, link 수)	Network size (terminal 수)	구조구성 변 수
그 물	(64, 1632)	64	(8, 1, 7)
	(121, 3201)	121	(11, 1, 7)
	(144, 3840)	144	(12, 1, 7)
	(256, 6976)	256	(16, 1, 7)
	(484, 13420)	484	(22, 1, 7)
	(529, 15341)	529	(23, 1, 7)
원환체	(64, 1856)	64	(8, 1, 7)
	(121, 3509)	121	(11, 1, 7)
	(144, 4176)	144	(12, 1, 7)
	(256, 7424)	256	(16, 1, 7)
	(484, 14036)	484	(22, 1, 7)
	(529, 15341)	529	(23, 1, 7)
초입방체	(64, 1984)	64	(6, 1, 5)
	(128, 3712)	128	(7, 1, 4)
	(256, 6400)	256	(8, 1, 3)
	(512, 14336)	512	(9, 1, 3)

3.2 구조별 실험

각 구조별 실험은 throughput, latency, 패킷 정체시간의 세 가지 변수를 측정하며 실시하였다. 각 구조별로 64개, 128개, 256개, 512개의 터미널 노드를 연결한 경우로 가정하고, 메시지 발생률을 증가시키면서 이에 따른 결과를 고찰하였다.

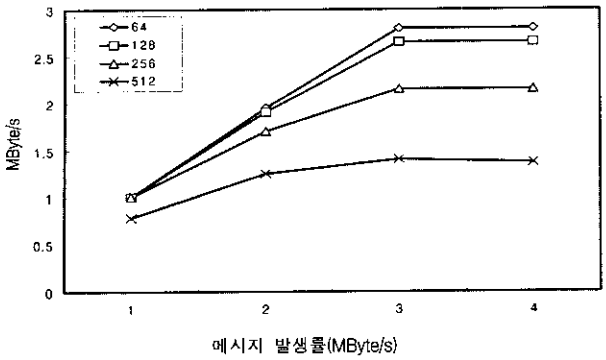
3.2.1 Throughput

다중 링크를 이용하는 그물구조의 throughput을 (그림 7)에 나타내었다. 그림에 나타나 있는 바와 같이 전체적으로 증가하는 특성을 보이고 있는며, 특히 메시지 발생률 3MByte/s까지 노드수가 적을수록 높은 비율로 throughput이 증가되고 있으며 그 이후부터는 그 증가율이 낮아 대략 메시지 발생률 3MByte/s인 때부터 네트워크가 포화상태에 도달하는 것으로 판단된다.



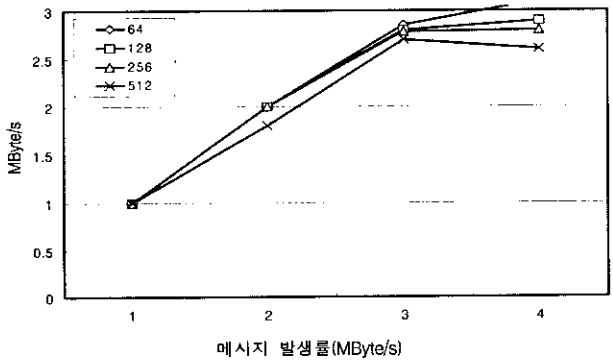
(그림 7) 그물구조의 throughput

.(그림 8)을 보면 원환체의 throughput도 메시지 발생률 3MByte/s까지는 비교적 높은 비율로 증가되고 있어 그물 구조에서의 throughput과 비슷한 특성을 보이는 것으로 나타났다. 노드의 수가 많은 경우에는 원환체의 throughput이 약간 우수함을 알 수 있다.



(그림 8) 원환체의 throughput

(그림 9)에 나타난 바와같이 초입방체의 throughput은 크기면에서 볼 때 그물구조와 원환체 구조보다 다소 높게 나타났으며 메시지 발생률 3MByte/s까지는 높은 비율로 throughput이 증가되고 그 이후부터는 낮은 비율로 증가되고 있는 것은 그물구조나 원환체가 보여준 throughput 특성과 비슷한 결과를 보이고 있다. 결과적으로 throughput면에서 볼 때 다중 링크는 노드수의 증가에 따라 그물구조, 원환체 및 초입방체에서 어느 정도 throughput의 향상에 도움을 주고 있음을 알 수 있다.

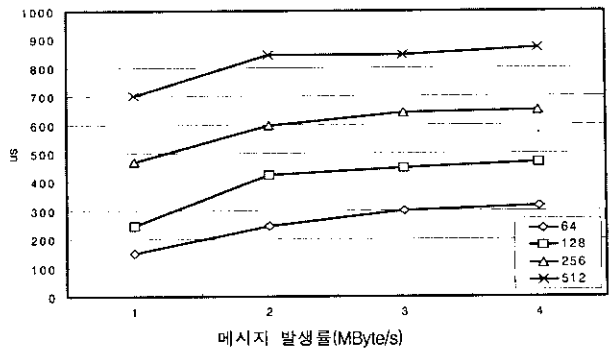


(그림 9) 초입방체의 throughput

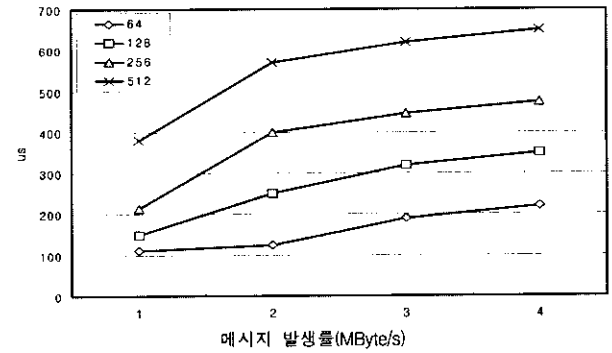
3.2.2 Latency

(그림 10)은 그물 구조의 latency를 보인 것으로 노드수가 많을수록 큰 latency를 나타내며 메시지 발생률이 증가됨에 따라 낮은 폭으로 latency가 증가되고 있다. 이 결과는 또한 단일 링크로만 연결한 망의 latency와 비교해 볼 때 다중 링크를 사용한 경우의 latency가 50%이하로 작게 나타나 다중 링크에 의해 블로킹 시간에서 큰 이득을 얻은 것으로 보인다.

(그림 11)은 원환체에서의 latency를 보였다. 그림에 나타나 있는 바와같이 원환체의 latency는 노드 수 256, 512개일 때 메시지 발생률 2MByte/s까지는 비교적 큰 비율로 증가되나 그 이후는 다소 낮은 비율로 증가되고 있다. 노드수 64, 128개 일 때의 latency는 메시지 발생률이 증가됨에 따라 낮은 비율로 점차 증가하고 있다.

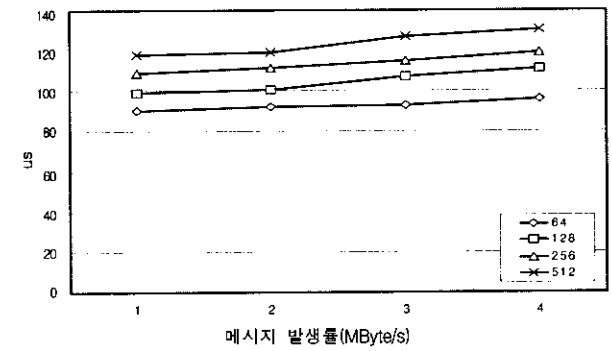


(그림 10) 그물구조의 latency



(그림 11) 원환체의 latency

(그림 12)는 다중 링크 환경에서 초입방체의 latency를 나타낸 것이다. 이 결과는 단일 링크 환경에서 초입방체의 latency와 비슷한 모양과 크기를 보이고 있다. 이것은 초입방체는 가정된 시험환경에서 다중 링크 환경이라 해도 실제 다중 링크를 사용한 만큼의 트래픽에 도달하지 않아 다중 링크에 의한 효과를 얻지 못했다는 것을 나타내며 트래픽을 분할하는 것이 다중 링크를 사용한 것과 같은 효과를 얻고 있다는 것을 보여주고 있다.

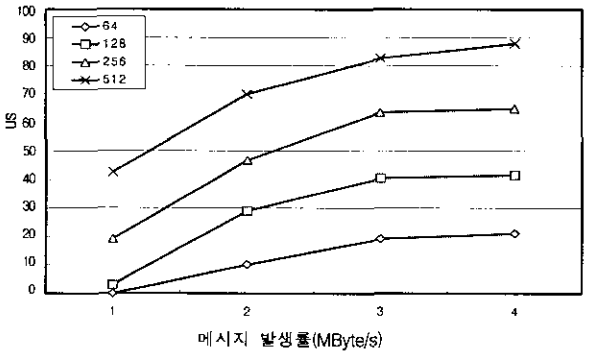


(그림 12) 초입방체의 latency

3.2.3 패킷 정체 시간

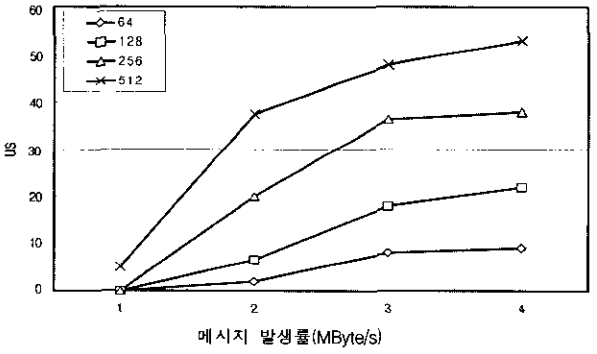
패킷 정체 시간은 망의 트래픽 효율을 알려주는 척도로 다중 링크 환경의 특성을 나타내기에 좋은 특성변수가 된다. (그림 13)은 그물구조의 패킷 정체시간을 나타낸 것으로 역

시 크기 면에서는 단일 링크 환경 때에 비해 크게 줄었으며 메시지 발생률이 증가함에 따라 패킷 정체 시간이 점차 증가하는 특성을 보이고 있다.



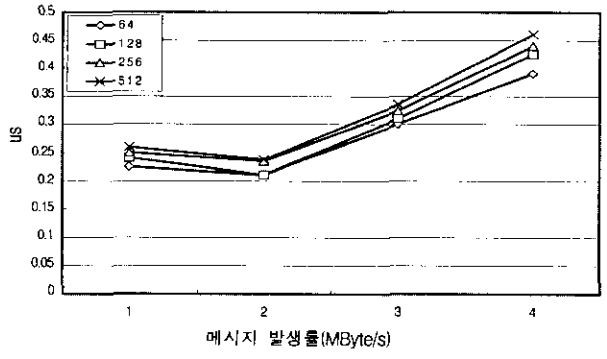
(그림 13) 그물구조의 패킷 정체 시간

(그림 14)는 다중 링크 환경에서 원환체의 패킷 정체 시간을 나타낸 것이다. 그물 구조에서와 같이 메시지 발생률이 증가함에 따라 패킷 정체 시간이 점차 증가하는 양상을 보이며 단일 링크 환경에서의 결과와 비교해 볼 때 크게 감소된 패킷 시간을 보이고 있다.



(그림 14) 원환체의 패킷 정체 시간

(그림 15)는 초입방체에서의 패킷 정체 시간을 보인 것이다. 그림에 나타나 있는 바와 같이 메시지 발생률 2MByte/s 이하에서 낮은 패킷 정체 시간을 보이며 2MByte/s 이상에서는 급격한 증가를 보이고 있는데 전체적인 특성은 단일 링크 환경의 패킷 정체 시간의 특성과 비슷한 형상을 보이고 있으며 크기면에서 큰 차이가 없는 것으로 나타났다. 이것은 초입방체에서 다중 링크에 의한 성능 향상은 미미하다는 것을 의미한다.



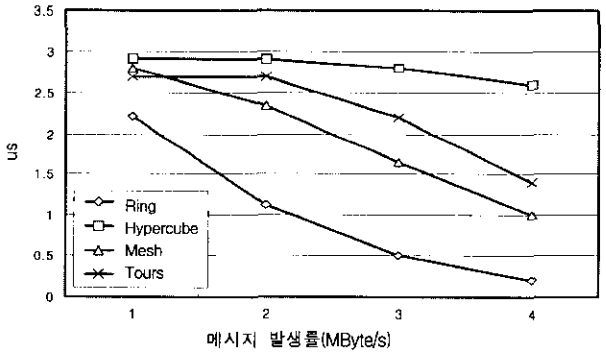
(그림 15) 초입방체의 패킷 정체 시간

3.3 구조별 성능 비교

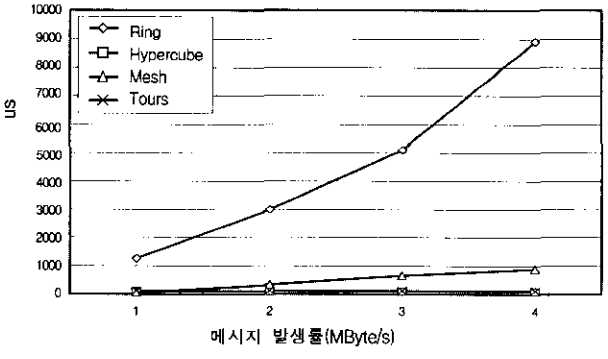
각 구조간의 성능 비교를 위하여 앞의 실험 결과중 메시지 발생률이 3MByte/s 때의 링, 그물, 원환체 및 초입방체의 throughput, latency와 패킷 정체 시간을 (그림 16-18)에 도식하였다.

(그림 16)을 보면 throughput은 초입방체가 가장 좋은 것으로 나타났고 다음으로 원환체, 그물구조의 순으로 나타났다. 다중 링크를 사용한 다른 구조들과 비교해 볼 때 노드수가 64, 128개일 때에는 그물구조의 throughput이 낮게 나타났으며, 노드 수가 128개 이상의 경우에도 일반적으로 원환체, 초입방체의 순서로 우수한 throughput을 보여주고 있다.

(그림 17)에서 latency는 그물구조, 원환체 그리고 초입방체의 순서로 증가하고 있음을 볼 수 있다. 이것은 단일 링크 환경에서의 결과와 유사하게 나타나는 것이다.



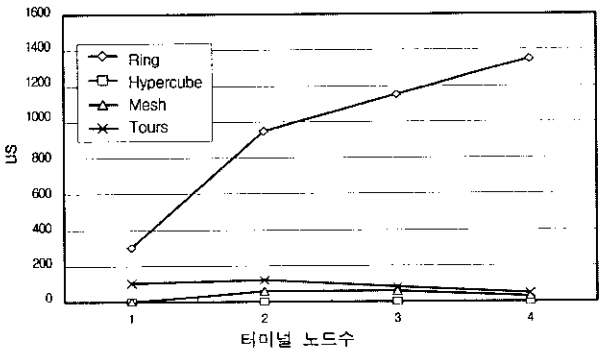
(그림 16) 다중 링크 환경에서 구조별 throughput



(그림 17) 다중 링크 환경에서 구조별 latency

패킷 정체 시간은 (그림 18)을 볼 때 그물구조, 원환체 및 초입방체의 순으로 나타났다. 링크구조의 경우 노드수가 많을수록 패킷 정체 시간이 커지고 있으며 그물구조, 원환

체, 초입방채등은 노드수에 관계없이 거의 일정한 패킷 정체 시간을 보이고 있다. 그물구조, 원환체, 초입방채 등은 다중 링크가 실험 환경에서 발생되는 트래픽을 충분히 수용할 수 있어 패킷 정체 시간이 다른 구조, 즉 링과 같은 구조와 비교해볼 때 월등히 작게 나타났다.



(그림 18) 다중 링크 환경에서 구조별 패킷 정체 시간

4. 멀티캐스트 실험 및 고찰

점대점 통신만을 지원하는 STC104로 구성된 네트워크에서 소프트웨어 멀티캐스트를 활용하는 것을 전제로 알고리즘을 구현하고 이를 구성된 망에 적용시켜 성능을 평가하였다. 멀티캐스트 알고리즘의 성능측정에 사용하는 특성변수는 다음과 같이 정의한다.

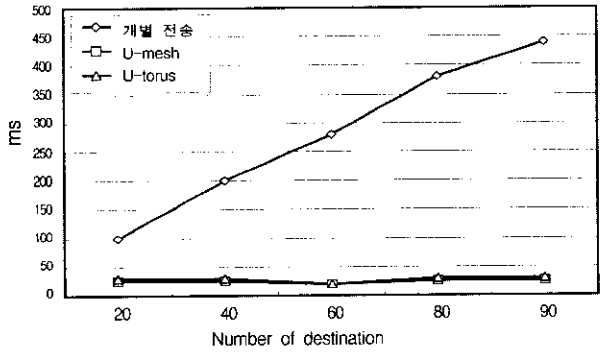
① 멀티캐스트 latency : 발원 노드가 메시지의 복사 본을 처음 보내기 시작할 때부터 마지막 목적 노드가 메시지를 받을 때까지의 시간

② 멀티캐스트 throughput : 단위 시간당 전송된 메시지의 양

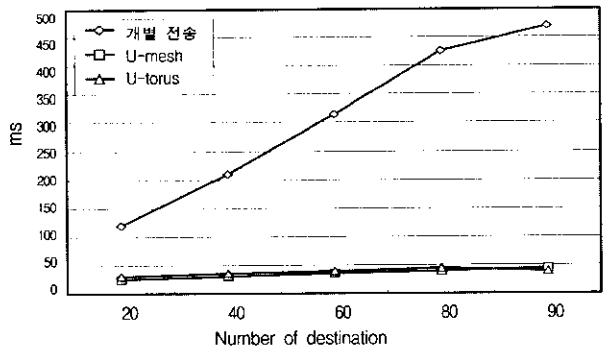
본 논문에서 구현된 알고리즘은 논문 I의 3.4절에서 소개한 U-mesh 알고리즘과 U-torus 알고리즘이다. 실험은 10 x 10 그물 구조와 10 x 10 원환체를 구성하고, 개별전송 방법과 U-mesh, U-torus 알고리즘의 성능을 비교 측정하였다. 구현한 멀티캐스트 프로세서를 연결하고 연결된 멀티캐스트 프로세서의 데이터에 대한 처리시간을 10μs로 가정하고 실험하였다. 성능 평가를 위하여 각 알고리즘 별로 목적 노드의 수를 증가시켜 가면서 n회에 걸쳐 지정한 발원 노드와 임으로 선택된 노드들에 대한 멀티캐스트실험을 실시하고 평균 멀티캐스트 latency와 평균 멀티캐스트 throughput을 구현하였다.

4.1 멀티캐스트 latency

(그림 19)와 (그림 20)은 그물구조와 원환체에서의 멀티캐스트 latency를 보인 것이다. 그물구조와 원환체 모두 U-mesh나 U-torus를 사용한 경우가 개별전송 방법을 사용한 경우보다 월등히 낮은 멀티캐스트 latency를 보이고 있다. 개별전송 방법의 경우 목적 노드수가 많아질수록 멀티캐스트 latency가 상대적으로 크게 증가되고 있는데 이것은 전송을 원하는 모든 노드에 대해 하나의 노드에서 차례로 전송이 되어야 하므로 멀티캐스트 latency가 목적 노드의 수에 크게 영향을 받는 것에 원인이 있다. 그러나, U-mesh나 U-torus와 같은 멀티캐스트 트리 알고리즘은 목적 노드의 수를 m이라 할 때 전송 단계의 수는 $\log_2(m)$으로 결정된다. 멀티캐스트 트리 알고리즘에서 목적 노드의 수가 20개라 할 경우 필요한 단계의 수는 4이며 80개의 목적 노드를 가지는 경우 6단계로 노드의 수가 60개임에도 불구하고 차이는 2단계에 불과하여 멀티캐스트 latency가 목적 노드의 수에 큰 영향을 받지 않음을 알 수 있다.



(그림 19) 그물구조에서의 멀티캐스트 latency



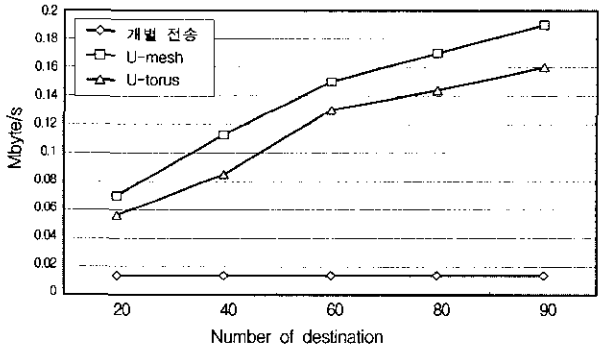
(그림 20) 원환체 구조에서의 멀티캐스트 latency

4.2 멀티캐스트 throughput

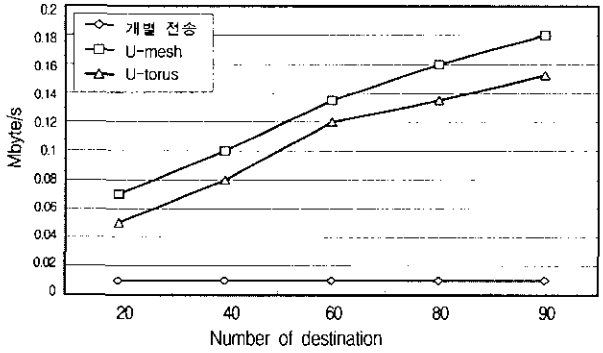
멀티캐스트 throughput은 멀티캐스트에서 단위 시간당 전달된 메시지 양을 나타내는 것으로 멀티캐스트 latency와 함께 알고리즘의 성능을 평가하는데 중요한 자료이다. 본논문에서는 네트워크의 성능과 함께 소프트웨어 멀티캐스트를 처리하는 멀티캐스트 프로세서의 성능을 멀티캐스트 throughput에 함께 반영된 것이다.

(그림 21)과 (그림 22)는 그물 구조와 원환체에서의 멀티캐스트 throughput을 나타낸 것이다. 그림에 나타난 바와 같이 그물구조와 원환체 모두에서 U-mesh가 우수한 멀티캐스트 throughput을 보여주고 있다. U-mesh나 U-torus모두 목적 노

드가 증가할수록 멀티캐스트 throughput이 증가하는 양상을 보이고 있다. 개별 전송의 멀티캐스트 throughput은 0.02 MByte/s이하로 극히 낮게 나타났으며 목적 노드의 수가 증가해도 멀티캐스트 throughput은 비교적 변화가 작게 나타났다.



(그림 21) 그물구조에서의 멀티캐스트 throughput



(그림 22) 원환체 구조에서의 멀티캐스트 throughput

5. 결 론

본 논문에서는 병렬처리 컴퓨터의 프로세서 망이나 일반 상용 네트워크 등에서 고속의 자료 전송을 처리해야 되는 경우를 위하여 STC104 상호 연결망의 전송 알고리즘과 그 성능에 관하여 연구하였다. STC104는 고속 라우팅 칩으로 웜홀 라우팅, 헤더 제거법, 구간 레이블링 기법 및 그룹 적응 라우팅의 기능을 내재하고 있어서 스위치의 구조적 특성과 기능적 특성을 시뮬레이션 하는 소프트웨어 패키지를 개발하였다. 개발된 소프트웨어는 event-driven network 시뮬레이션 라이브러리를 이용하여 STC104의 기능적 구조를 구현하였다.

STC104 특성을 고려하여 그물, 환원체 및 초입방체의 망을 구성한 후, 개발된 시뮬레이션 패키지를 통해 그 성능을 비교 분석하였다. 실험은 다양한 크기를 가지는 각각의 구성망에 대해서 수행되었으며 일정한 메시지 비율로 각각의 송신 노드가 임의의 수신 노드로 메시지를 보내는 환경에서 실험하였다.

단일 링크 환경에서 링구조는 2개의 링크만이 외부로 연결되나 6-초입방체의 경우 6개의 링크가 외부로 연결되어 구조간의 불평등이 지적되어 스위치간의 상호 연결에 복수의 링크를 허용하는 다중 링크 환경에서의 실험을 수행하였다. 다중 링크가 적용된 상황에서 패킷 정체 시간이 단일 링크 환경에 비해 현저히 낮아져 이로 인한 throughput의 증가가 보였으나 노드수가 많은 경우에는 throughput의 감소가 발생되어 hop수의 증가에 의한 성능 감소 효과를 상쇄하기에는 역부족인 것으로 나타났다. 그물구조, 원환체의 경우 다중링크를 사용한 환경에서 단일링크 때와 다중링크 때의 성능에 큰 차이가 나타나지 않아 다중링크에 의한 효과가 보이지 않았다.

점대점 네트워크의 취약점을 보완하는 멀티캐스트를 STC104로 구성된 그물구조와 원환체 망에서 구현하고, 그 성능을 측정하였다. 성능의 측정은 개별전송 방법과 U-mesh 알고리즘 그리고 U-torus 알고리즘을 소프트웨어로 구현하고 실험하였다.

실험 결과 그물구조와 원환체 구조 모두 개별전송 방법의 경우 U-mesh, U-torus보다 크게 성능이 떨어졌으며 U-mesh 알고리즘이 가장 좋은 결과를 보여주었다. U-mesh와 U-torus 모두 목적 노드의 증가에 비해 멀티캐스트 latency는 큰 변화를 보이지 않았으나 개별전송 방법은 노드수가 증가할수록 멀티캐스트 latency가 크게 증가하였다. 이것은 개별전송의 경우 노드의 수가 곧 전송 단계의 수에 해당하지만 U-mesh, U-torus의 경우 $\log_2(m)$의 전송 단계 수를 가지는 것에 그 원인이 있다고 본다.

본 연구에서 수행한 결과는 STC104와 같은 고속 스위치를 이용하여 대규모의 통신망을 구성할 경우, trhoughput, 또는 latency 등을 예측하여 응용 시스템의 타당성을 점검할 수 있는 중요한 정보로 제공될 수 있다. 메시지의 생성율이나 노드의 개수가 각기 다른 망의 구성에서 미치는 특성을 분석하였기 때문에 최적 시스템의 변수들을 예측할 수 있도록 도와준다. 또한 하드웨어적으로 지원되지 않는 소프트웨어 멀티캐스트를 시뮬레이션한 결과는 하드웨어적인 멀티캐스트를 구현하는데 필요한 자료로 활용될 수 있다. 소프트웨어 멀티캐스트의 실험은 스위치 하나 당 터미널 노드 하나를 연결한 경우를 가정하였다. 만일 스위치 하나에 여러 개의 터미널 노드를 연결할 경우 하나의 스위치에 연결된 근거리 노드와 서로 다른 노드에 존재하는 원거리 노드 사이의 전달 latency의 차이가 멀티캐스트 노드의 순차성에 의존한 U-mesh, U-torus 알고리즘 모두에 영향을 미칠 것으로 예상되어 이에 대한 해결이 연구과제로 남아있다.

참 고 문 헌

[1] Simpson, M, Thompson, P. W., "The T9000 Communication Architecture," Networks, Routers & Transputers, Inmos Ltd. 1993.

[2] The STC104 Asynchronous Packet Switch, Preliminary Data Sheet, SGS Thompson Microelectronics, June 1994.

[3] Ni, L. M. and McKinley, P.K., "A Survey of Wormhole

Routing Techniques in Direct Networks," IEEE Computer, Vol.26, pp.62-76, February 1989.

[4] Van Leeuwen, J. and Tan, R. B., "Interval Routing," The Computer Journal, Vol.30, No.4, pp.298-307, 1987.

[5] Tsai, Yih-Jia and McKinley, Philip K., "An Extended Dominating Node Approach to Collective Communication in All-Port Wormhole-Routed 2D Meshes," Proc. 1994 Scalable High Performance Computing Conference, pp.199-206, May, 1994.

[6] Robinson, David E., McKinley, Philip K., and CHeng, Betty H. C., "Optimal Multicast Communication in Wormhole-Routed Torus Networks," Proc. 1994 International Conference on Parallel Processing, August, 1994.

[7] M. Simpson, P. W. Thompson, "DS-Links and C104 Routers," Networks, Routers, & Transputers INMOS Ltd., 1993.

[8] P. Kermani and L. Kleinrock, "Virtual cut-through : a new computer communication switching technique," Computer Networks, Vol.3, pp.267-286, 1979.

[9] Casner, "Frequently Asked Questions on the Multicast Backbone," ftp : //venera.ise.edu/mbone/faq.txt, May 6, 1995.

[10] Petri Koskelainen, "Report on Streamworks (tm)," http : //www.cs.tut.fi/~petkos/streams.html, August 1995.

[11] Philip K. McKinley, Hong Xu, Abdol-Hossein, "Unicast-based Multicast Communication in Wormhole-routed Networks," IEEE Transactions on Parallel and Distributed Systems, Vol.5, No.12, pp.1252-1265, December 1994.

[12] Ametek Computer Research Division, Arcadia, California, Ametek System 14, Mars System Software User's Guide Version 1.0, 1987.

[13] Chaudrhary, M. L. and Templeton J. G. C., A First Course in Bulk Queues. John Wiley and Sons, New York, 1993.

[14] Krishnamoorthy, A. and Ushakumari, P. V., "A Queueing System with Single Arrival Bulk Service and Single Departure," Mathematical and Computing Modelling, pp. 99-108, No.31, 2000.

[15] J. Robert Jump, NETSIM Reference Manual Version 1.0, ECE Dept., Rice University, May 1993.

[16] J. Robert, YACSIM Reference Manual Version 2.1, ECE Dept., Rice University, March 1993.

이 효 종

e-mail : hlee@mail.chonbuk.ac.kr

1982년 전북대학교 지구과학과 졸업(이학사)

1985년 University of Utah 기상학과 졸업(이학석사)

1986년 University of Utah 컴퓨터과학과 졸업(공학사)

1988년 University of Utah 컴퓨터과학과 대학원 졸업(공학석사)

1991년 University of Utah 컴퓨터과학과 대학원 졸업(공학박사)

1986년~1989년 University of Utah 컴퓨터과학과 조교

1989년~1991년 University of Utah 선임프로그래머

1991년 University of Utah 대기연구소 연구원

1991년~현재 전북대학교 교수

관심분야 : 병렬영상처리, 인공지능, 병렬인식 알고리즘 개발 및 소프트웨어 공학