고속의 채널 기반 네트웍에서 효율적인 패킷 스케쥴링 기법

김영환, 고재진, 전기만
전자부품연구원 유비쿼터스 컴퓨팅 센터
{yhkim93, jaejini, kmjeon}@keti.re.kr

A Efficient Strategy to Schedule Packets in Channel based Network

Young Hwan Kim, Jae Jin Go, Jeon Ki Man
Ubiquitous Computing Research Center, Korea Electronics Technology Institute

요약

네트워크 사용자의 급속한 증가로 네트워크 내의 부하를 감당하기에는 많은 어려움을 가져왔다. 이와 같은 이유로 기존의 TCP/IP 에서 세션을 통하여 노드들 간의 통신을 연결하는 방식에서 현재는 하나의 채널을 통해 고속의 I/O 가 가능하도록 하는 기술의 많이 연구되고 있다. 그 대표적인 것으로 인피니밴드가 있다. 인피니밴드는 프로세싱 노드와 입출력 장치 사이의 통신, 프로세스간 통신에 대한 산업 표준이 되고 있고 프로세싱 노드와 입출력 장치를 연결하기 위해 스위치 기반의 상호 연결은 전통적인 버스 입출력을 대체하는 새로운 입출력 방식을 사용한다. 또한 인피니밴드는 서버 시스템에서 요구하는 여러 특징 중에서 어플리케이션에게 QoS (Quality of Service)을 보장할 수 있는 메커니즘을 포함하고 있다. 본 논문에서는 어플리케이션에서 요구하는 각 트레픽의 요구 대역폭 크기에 따라 구분하여 효율적인 중재 테이블의 구성을 통해 데이터 전송에 있어 QoS 에 합당하게 출력 포트를 통해 나갈 수 있도록 중재 테이블을 관리하는 기법을 제안하고 이를 NS-2 (Network Simulator)을 이용하여 성능 평가를 했다. 특히 기존 인피니밴드에서의 중재 테이블을 이용한 패킷 처리량과 본 논문에서 제안한 중재 테이블 구성 방식을 이용한 패킷 처리량을 비교했으며 성능 평가 결과는 기존의 인피니밴드에서의 패킷 처리량에 비해 약 19% 향상된 결과를 나타냈다.

키워드: 채널, 인피니밴드, 중재 테이블, QoS

1. 소개

네트워크 기술의 급속한 발달과 정보의 효율적 공유에 대한 요구의 증가로 거의 모든 컴퓨팅 장비들이 네트워크에 연결되어 있다. 따라서 복잡하게 구성되어 있는 전체 네트워크의 상태 및 장비들을 효율적으로 파악하고, 관리하기 위한 필요성이 대두되고 있다. 특히 대용량 스토리지와 서버사이 입출력 분야에서는 한 개의 프로세서와 여러 개의 입출력 장치를 가진 소규모의 서버에서 수백개의 프로세서와 수천개의 입출력 장치를 가진 대규모의 슈퍼 컴퓨터까지 사용 가능한 인피니밴드는 더욱 더 중요성이 대두 되고 있다.

현재 컴퓨터 시스템에서 사용되고 있는 입출력 버스 방식은 디스크 접근, 특히 고 성능의 서버에 있어서 병목 현상의 주요원인으로 나타나고 있다. 이러한 버스 방식은 구조가 단순하다는 큰 이점을 가지고 있어 지금까지 산업 전반적으로 사용되어 왔지만 버스 기반의 입출력 시스템은 현재의 디바이스 장치들이 요구하는 데이터 전송 대역폭을 처리할 수 있을 만큼의 시스템 입출력 성능을 가지고 있지 않다. 또한 현재 대부분의 네트워크 제품들은 최고의 패킷 처리량과 최소의 전송 지연, 그리고 전송 대역폭에 대한 보장을 요구해왔다. 이와 같은 이유로 해서 인피니밴드 표준화 단체 (IBTA)는 프로세싱 노드와 입출력 장치간 통신과 프로세스간 통신을 위한 새로운 표준을 정하기 위해 인피니밴드 기술을 개발하게 되었다.

많은 어플리케이션이 현재의 네트워크에서 요구하는 대역폭에 만족하도록 기존의 인터넷 상에서 QoS 을 제공하도록 하기 위한 컴퓨터 시스템 구조가 개발되고 있는데, 대표적으로 Differentiated Service 가 있다. 이와 같이 어플리케이션에게 QoS 을 보장하기 위한 방법으로 인피니밴드에서도 사용되고 있는 기법이 있는데, 각 어플리케이션의 서비스 레벨에 따라 서브넷 매니져에 의해서 관리되는 중재 테이블을 이용해서 데이터 패킷을 목적지 노드까지 원할히 전송하는 방법이다. 본 논문에서 우선 QoS 을 보장하기 위해 각 트레픽에 대한 타입을 대역폭 별로 분류하고, 기존의 인피니밴드에서 하위 우선순위를 갖는 서비스 레벨에 대해서는 상위 우선순위에 의해 패킷 전송의 대

기 시간이 길어지는 단점을 보완하기 위해 새로운 중재 테이블을 생성하는 기법을 제안한다. 이 기법은 기존에 상위 우선순위의 대역폭에는 거의 영향을 끼치지 않고 하위 우선순위의 데이터 처리를 향상 시킴으로서 대기 시간을 줄이는 것이다 [1-3].

본 논문의 구조는 다음과 같다. 2 장에서는 인피니밴드에 대해 간략히 소개하고 3 장에서는 본 논문에서 제안하는 새로운 중재 테이블 생성 방법에 대해 설명하며 4 장에서는 NS 을 이용해서 제안한 기법을 시뮬레이션을 한다. 마지막으로 5 장에서는 요약과 함께 시뮬레이션을 통해 얻은 결과로 결론을 내린다.

2. 인피니밴드

2.1 시스템 구조

인피니밴드는 고속의 점대점 링크에 스위치를 기반으로 노드가 서로 연결된 망으로 설계되었다. 인피니밴드망은 하나 또는 그 이상의 스위치와 프로세싱 노드, 입출력 디바이스 장치로 구성된 서브넷이 라우터에 의해서 상호 연결되어 있다. 인피니밴드에서 라우팅은 각 스위치에 저장된 포워딩 테이블를 기반으로 하며, 유연성과 확장성을 제공하기 위하여, 사용자에 의해 정의된 어떠한 토폴로지도 지원한다.

인피니밴드 링크는 양방향 점대점 통신 채널로 되어 있고, 링크의 신호 발생율은 현재 2.5GHz 이며 물리적 링크 레벨에서는 보다 높은 대역폭을 얻기 위해 병렬적으로 사용되는데, 가장 낮은 대역폭은 1X 로서 2.5Gbps 이고 최대 12X 인 30Gbps 까지 지원한다.

인피니밴드 스위치는 서브넷 매니져에 의해 토폴로지의 초기화와 네트워크의 변화에 따른 수정에 의해 얻어진 포워딩 정보를 기반으로 만들어진 포워딩 테이블을 통해서 로컬에서 목적지로 메시지를 전송하게 된다. 그 메시지는 링크 상에서 스위치를 통해 전송 되도록 패킷의 형태로 세그먼트화 된다. 그 패킷의 크기는 헤더를 제외하고 256bytes, 1KB, 2KB, 4KB 까지 될 수 있다[8-10].

인피니밴드 전송 메커니즘은 종단노드 간에 여러가지 타입의 통신 서비스를 제공하며 이와 같은 타입은 연결 지향성과 데이터그램 서비스 그리고 신뢰성과 비신뢰성으로 분류 될 수 있다. 또한 대역폭 보장과 최대 지연 마감과 같은 QoS 요구 조건을 만족하도록 하기 위해 어플리케이션은 자원 할당을 할 수 있도록 신뢰성있는 연결을 사용해야 한다. 더 자세한 내용은 인피니밴드 명세서 1.0a 를 참조하기 바란다[6].

2.2 인피니밴드 QoS

인피니밴드는 QoS 을 제공하기 위하여 서비스 레벨 (Service Level), Virtual Lane (VL), VL 중재 테이블 (VL Arbitration Table)의 세가지 메커니즘을 제공한다. 인피니밴드는 최대 16 개의 서비스 레벨을 정의하고 있지만 각 레벨의 정확한 특성에 대해서는 정의하고 있지 않다. 그러므로 어떻게 각 서비스 레벨에 따라 차별적인 데이터 전송을 수행 할지는 망 관리자에 의해서 정의 될 수 있다. 우리는 본 논문에서 각 트레픽을 카테고리별로 분류 함으로서 서로 상이한 서비스 레벨로부터 패킷들을 구분 할 수 있도록 했다.

인피니밴드 포트는 VL 이라는 하나의 물리적인 링크내에 다중의 가상 링크를 생성하는 메커니즘을 지원한다. 하나의 VL 은 한 포트에 송수신을 위한 버퍼로 나타낼 수 있고, 그 포트는 최소 두개에서 최대 16 개의 VL (VL_0...VL_{15})로 구성되며 모든 포트는 반드시 서브넷 매니져에 의해서 사용될 VL_{15} 를 지원해야 한다. 이 VL_{15} 는 다른 VL 에서의 데이터 트레픽보다 우선순위가 높다. 또한 VL 을 통해 전송되는 각 패킷의 헤더에는 SL 에 대한 정보를 가지고 있고, SL 과 VL 사이의 관계는 *SLtoVLMappingTable* 에 의해서 정해진다. 그리고 각 VL 은 데이터 흐름 제어를 위한 목적으로 별도의 리소스를 가지고 있어야 한다[4, 6-7].

두개 이상의 VL 이 실행될 때 각 데이터 VL 의 우선 순위는 *VLArbitrationTable* 에 의해서 정해지며, 이 중재 테이블은 관리목적의 패킷만을 위한 VL_{15} 을 제외한 데이터 VL 에 대해서만 적용된다. *VLArbitrationTable* 에는 두개의 테이블을 가지고 있는데 하나는 상위 우선순위을 가진 패킷을 스케쥴링을 위한 64 개의 테이블 엔트리를 가진 테이블과 또 다른 하나는 하위 우선순위를 가진 패킷을 스케쥴링을 하기위한 64 개의 테이블 엔트리를 가진 테이블이다. 그러나 인피니밴드 명세서에는 어떤 것이 상위 우선순위를 갖는지 하위 우선순위를 갖는지는 정의하고 있지 않다. 그 중재 테이블은 각 엔트리에 정의된 우선순위 수준에 따라 가중치 라운드 로빈을 수행한다. 각 엔트리에는 해당 VL 을 통해서 전송될 64 바이트 단위의 0 에서 255 까지 가중치가 설정되어 있는데 데이터를 전송하기 위해 이 테이블을 참조하게 되면 각 엔트리에 설정된 가중치 만큼을 전송하고 다음 엔트리를 처리하는 것을 반복해서 수행한다[5].

중재 테이블에는 *VLHighLimit* 이라는 필드 값을 가지고 있는데 이는 낮은 우선순위의 패킷이 전송되기 전에 높은 우선순위를 갖는 패킷을 전송할 수 있는 최고의 패킷 수를 나타낸다. 즉, 상위 우선순위 테이블의 VL 은 하위 우선순위의 테이블의 VL 을 통해 데이터를 전송하기 전에 최대 *VLHighLimit* * 4096 바이트를 전송할 수 있다. 만약 주어진 시간 동안 전송될 상위 우선순위의 패킷이 없다면 하위 우선 순위의 패킷을 전송하게 되는데 인피니밴드에서는 하위 우선순위를 갖는 테이블을 참조할 때 엔트리 한 개만 참조하고 다시 상위 우선순위 테이블을 참조하도록 되어 있다.

3. 제안

이 장에서 우리는 인피니밴드에서 정의하는 각 서비스 레벨에 따라 트레픽을 분류하고 출력 포트의 *VLArbitrationTable* 을 제안하고자 하는 방식으로 재구성하며 관련된 필드 값을 새로이 구성하여 시뮬레이션을 할 것이다.

3.1 평균 요구 대역폭에 따른 트레픽 분류

다음의 [8]을 통해서 인피니밴드에서의 트레픽을 DBTS (Dedicated Bandwidth Time Sensitive), DB (Dedicated Bandwidth), BE (Best Effort), CH (Challenged)로 모두 4 개의 카테고리로 분류했다. 일부 첨가할 카테고리가 있지만 본 논문에서 시뮬레이션을 하는데는 4 개의 카테고리만으로 충분하다. 먼저 BE 와 CH 는 일반적인 이데넷망에서도 적용되는 것이므로 본 논문에서는 QoS 에 따라 평균 요구 대역폭으로 나누기 위해 DBTS 와 DB 만을 가지고 서비스 레벨을 분류할 것이다 (더 자세한 내용을 [8]을 참조하기 바란다). 또한 인피니밴드에서는 SL 을 최고 16 개로 구분하고 각 포트는 최고 16 개의 VL 을 가지고 있는데 본 논문의 시뮬레이션에서는 4 개의 SL 과 10 개의 VL 을 사용할 것이다.

3.2 패킷 스케쥴링

인피니밴드 명세서에 의하면 각 중재 테이블은 상위 우선 순위 64 개와 하위 우선 순위 64 개의 엔트리로 총 128 개의 엔트리로 구성되어 있고 각 엔트리는 최고 255 의 가중치를 가질 수 있으며 가중치당 64 바이트의 크기를 갖는다. 만약에 한 엔트리가 255 의 가중치를 갖는다면 그 엔트리로 전송이 가능한 데이터 크기는 255*64B=16KB 가 되고 상위 우선 순위의 64 개 모든 엔트리가 255 라면 전송 가능한 전체 데이터 크기는 64*16Kbytes=1MB 가 된다. 그리고 *VLHighLimit* 의 값을 두어 상위 우선 순위의 데이터 전송 크기에 제한을 두었다

인피니밴드에서는 패킷을 스케쥴링 하기위해 처음에는 상위 우선순위의 각 엔트리를 참조하다가 *VLHighLimit* 의 데이터 크기가 되면 하위 우선순위의 엔트리를 참조하게 된다. 그러나 하위 우선순위에서는 전송 가능한 제한된 데이터 크기가 없다. 단지 하위 우선순위에서 한 개의 엔트리 만을 참조하여 데이터를 전송한 후, 다시 상위 우선 순위의 데이터를 처리하면서 패킷 스케쥴링을 되풀이 한다. 여기서 문제가 되는 것은 패킷을 스케쥴링하기 위해 인피니밴드는 각 중재 테이블의 엔트리를 참조하여 해당하는 VL 에 쌓여 있는 데이터를 전송하게 되는데 하위 우선순위의 엔트리는 단지 한번 만을 참조하여 데이터를 처리함으로서 하위 우선순위의 VL 에 쌓여 있는 데이터는 대기 시간이 길어진다는 것이다. 여기서 우리는 별도의 하위 우선순위 엔트리에도 *VLLowLimit* 을 두어 전송 가능한 총 데이터 크기를 정의하는 것이다. 결과적으로 상위 우선 순위의 데이터 전송에는 영향을 거의 주지 않고 하위 우선순위 대해서는 사용하고자 하는 VL 에 대한 패킷의 전송 대기 시간을 줄이게 된다. 다음은 간단한 예를 통해 전체적은 동작을 알아본다.

우선 우리는 다음과 같은 가정을 한다. 모든 데이터의 MTU (Maximum Transfer Unit)는 4KB 이고 *VLHighLimit* 는 4, 4*4KB=16KB 이다. 또한 VLLowLimit 은 1, 1*4KB=4KB 이다.

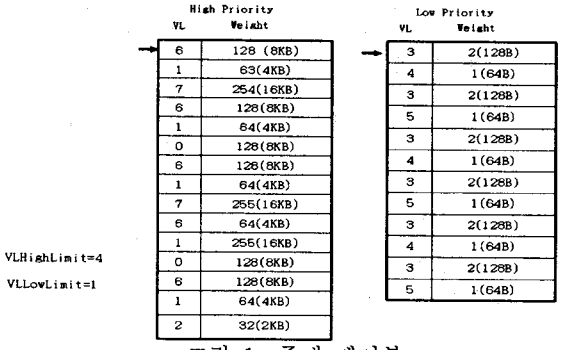

그림 1. 중재 테이블

그림 1 은 전체적인 패킷 스케쥴링에 대한 동작을 알아보기 위해 예를 든 중재 테이블이다. 우선 상위 우선순위의 첫 엔트리에 포인터가 있고 첫 엔트리의 전송 가능한 패킷의 크기는 128*64B=8KB 이다. MTU 가 4KB 이므로 두개의 패킷이 보내진다. 아직 전송된 패킷의 크기가 *VLHighLimit* 보다 작으므로 다음의 엔트리로 포인터가 이동한다. 두번째 엔트리는 전송할 패킷의 크기가 4KB 이므로 한 개의 패킷을 전송한다. 전송된 전체 패킷의 크기가 *VLHighLimit* 보다 아직 작다. 따라서 포인터는 다음 엔트리로 이동하고 이번에는 전송할 패킷 크기가 16KB 이지만 현재 앞에서 전송된 패킷이 총 12KB 이고 *VLHighLimit* 이 16KB 이므로 현재 한 개의 4KB 패킷만을 전송할 수 있다.

전송된 전체 패킷의 크기가 16KB 이므로 하위 우선순위 테이블로 포인터가 이동한다. 여기서 기존의 인피니밴드에서는 단지 첫 엔트리의 128B 만을 전송하고 다시 상위 우선순위로 포인터를 이동하지만 본 논문에서는 *VLLowLimit* 을 두어 하위 우선순위의 각 엔트리에서의 데이터 전송을 향상 시키도록 했다.

```
for(i=1;i<MAX_ENTRY;i++)
{
    currentVlno=VL[i][0];
    currentweight=VL[i][1];
    while(currentweight>0)
    {
        if((PQ[currentVlno].pkt!=NULL) && (PQ[currentVlno].pkt->size>0))
        {
            pktweight = (int)(PQ[currentVlno].pkt->size / WS);
            presize =   PQ[currentVlno].pkt->size;
            processsize = currentweight*WS;
            PQ[currentVlno].pkt->size -= (currentweight * WS);
            intVl = VL[i][1];
            VL[i][1] -= pktweight;
            currentweight -= pktweight;
            sendPacket[i]++;
        }
        else
        {
            PQ[currentVlno].pkt=PQ[currentVlno].pkt->next->next;
            processedEvent[currentVlno]++;
            PQ[currentVlno].queuesize--;
        }
    }
    if(VL[i][1]<=0)
        VL[i][1]=initVL[i][1];
    k=i+1;
    if(VL[k][1]==0)
        break;
}
```

위의 코드는 시뮬레이션에서 사용된 NS-2 소스코드이다. 각각의 상위 우선순위 가중치와 하위 우선순위 가중치는 해당하는 만큼의 패킷이 전송되면 현재의 가중치로 서로 전환되면서 상,하위 엔트리를

계속해서 참조하게된다. 그림 1 에서 하위 우선순위의 *VLLowLimit* 이 1KB 이므로 11 번째 엔트리까지 처리하고 상위 우선순위에서 하위 우선순위로 이동하기 전의 상태에 포인터가 이동한다. 이와 같은 패킷 스케쥴링 방식을 이용해서 우리는 다음 장에서 기존의 방식과 새로운 방식에 대해 시뮬레이션을 수행했다.

4. 성능 평가

성능 평가를 위해서 하나의 포트에 10 개의 VL 을 사용했고, 해당 VL 은 3 장에서 클래스에 따라 다음과 같은 카테고리로 분류되었다: SL0(8-64Kbps), SL1(64Kbps-1.55Mbps), SL2(64Kbps-64Mbps), SL3(64Mbps-300Mbps). 여기서 임의적으로 SL0 과 SL1 을 하위 우선순위 VL 로 두고 SL2 와 SL3 는 상위 우선순위 VL 로 사용하도록 했다. 또한 각 VL 은 자신의 버퍼를 가지고 있는데 버퍼크기는 패킷의 크기에 맞게 4096 바이트로 했다. 우리는 위와 같은 가정으로 NS-2 을 이용해 시뮬레이션을 했다.

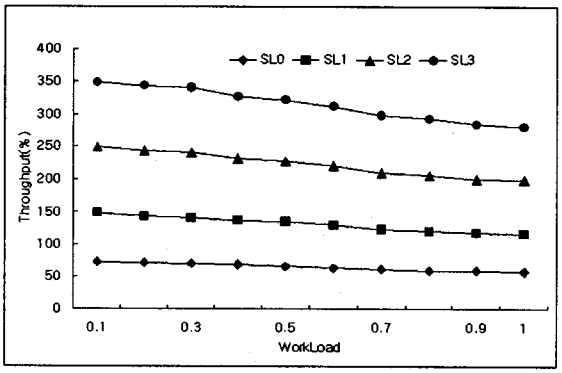

그림 2 인피니밴드에서 각 SL 에 따른 패킷 처리량

그림 2 는 기존의 인피니밴드에서 중재 테이블을 이용한 패킷 스케쥴링에서 패킷 처리량에 대한 결과인데, 그림에서 보듯이 하위 우선순위 테이블 엔트리를 이용하는 SL 에 대해서는 완전하게 데이터 패킷이 전송되지 않고, 있으며 실제 VL 에서 자신이 전송 될 차례를 기다리기 때문에 우선 순위가 낮은 SL 에 대해서는 전송 대기 시간이 길어진다.

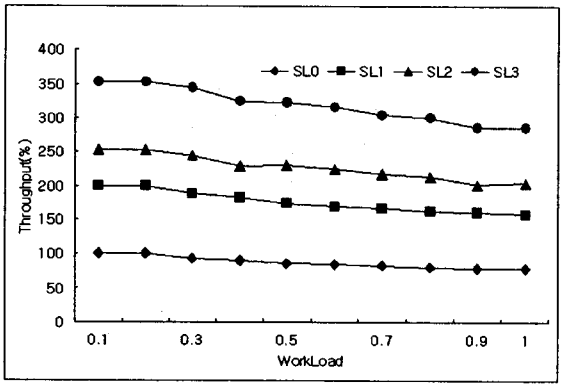

그림 3. 재구성을 통한 각 SL 의 패킷 처리량

그림 3 은 본 논문을 통해 제안한 방식으로 중재 테이블을 재구성하여 얻은 결과이다. 기존의 상위 우선순위의 데이터 처리량에는 거의 영향을 주지 않고 하위 우선순위의 데이터 처리량이 향상된 것을 알 수 있다.

5. 결론

우리는 본 논문에서 인피니밴드의 하위 우선순위 엔트리의 처리에 있어 패킷의 전송 대기 시간이 길어진다는 문제점을 파악하고, 이를 보안하는 방법을 제시했으며 기존의 방식에 비해서, 상위 우선순위의 데이터 패킷 처리에 거의 영향을 주지 않고 하위 우선순위의 패킷 처리량을 향상시켰다. 그러나, 아직까지 각 SL 에 대한 좀더 정확한 분류나 VL 의 제한된 수로 인해 정확한 결과를 얻는데 약간의 미비한 점이 있다.

인피니밴드에서 패킷의 처리량을 향상하고 QoS 를 보장하기 위해서는 중재 테이블의 구성이 매우 중요하다. 본 논문에서는 단지 중재 테이블을 구성하기 위해 필요한 필드를 첨가했지만 다음에는 실제 중재 테이블의 각 엔트리의 배치를 최적의 상태로 구성하여 보다 정확하고 향상된 패킷 처리량을 얻도록 해야 할 것이다.

참고 문헌

[1] F. Alfaro, J. Sanchez, and J.Duato. " AStrategy to Compute the InfiniBand Arbitration Table". Technical Report DIAB-01-02-20, Oct. 2001. http://raap.info-ab.uclm.es/diab-01-02-20.pdf

[2] F. Alfaro, J. Sanchez, and J. Duato. "A Strategy to Manage Time Sensitive Traffic in InfiniBand". In *proceedings of Workshop on Communication Architecture for Clusters*, Apr. 2002, Florida.

[3] J. Pelissier. "Providing Quality of Service over InfiniBand Architecture fabrics". In *Proceedings of the 8th Symposium on Hot Interconnects*, Aug. 2000.

[4] G. Pfister. High Performance Mass Storage and Parallel I/O, chapter 42: *An Introduction to the InfiniBand Architecture*, pages 617-632. *IEEE Press and Wiley Press*, 2001

[5] P. by ISSG Technology Communications. InfiniBand Architectural Technology. Technical Report TC000702TB, Compaq Computer Corporation, July 2000.

[6] InfiniBand Trade Association. Infiniabnd Architecture Specification Volume 1. Release 1.0, Oct. 2000.

[7] InfiniBridge MT21108. Product overview, Mellanox Technologies Inc., 2001. http://www.mellanox.com/products/whitepaper.html

[8] M. Schwartz and D. Beaumount. "Quality of Service requirements for audio-visual multimedia services". ATM Forum, ATM94-0640, July 1994.

[9] K.H.Yum, E.J.Kim, and C.R.Das. "QoS Provisioning in Cluster: An Investigation of Router and NIC Design". In *International Symposium on Computer Architecture (ISCA)*, July 2001.

[10] S. Blake, D. Black, M. Carlson, E. Davies, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.