論文2001-38TC-12-7

입력큐 교환기를 위한 스케줄링기법

(An Efficient Scheduling for Input Queued Switch)

李 相 昊 *, 申 東 烈 **

(Sang-Ho Lee and Dong-Ryeol Shin)

요 약

입력큐방식의 교환기는 간결하며 고속교환을 위한 효과적인 교환방법이나 입력 측의 큐에서 발생하는 HOL-블로킹(HOL-Blocking)은 패킷들의 대기시간을 크게 증가시켜 전체 시스템의 효율을 58%로 제한한다. 이를 해결하는 방법은 별도의 스케줄러(scheduler, contention controller)를 두어 블로킹의 방지 및 높은 처리율을 얻고 있다. 대부분의 스케줄러의 구현은 중앙집중방식으로 구현되는데 이는 다양한 교환기의 구성을 어렵게 한다. 본 논문에서는 입력포트별로 간단하면서 분산된 형태의 스케줄러를 소개하고 모의실험을 통하여 성능을 검증한다.

Abstract

Input queueing is useful for high bandwidth switches and routers because of lower complexity and fewer circuits than output queueing. The input queueing switch, however, suffers HOL-Blocking, which limits the throughput to 58%. To get around this low throughput, many input queueing switches have centralized scheduler, which centralized scheduler restrict the design of the switch architecture. To overcome this problem, we propose a simple scheduler called PRR(Pipelined Round Robin), which is intrinsically distributed and presents to show the effectiveness of the proposed scheduling.

Ⅰ. 서 론

패킷 교환기는 버퍼의 배치방식에 따라 입력큐방식(Input-Queued)과 출력큐방식(Output-Queued)으로 나눌 수 있다. 출력큐방식은 교환하고자하는 포트의 수(N)에 따라 각 포트의 전송속도의 N배의 속도로 동작해야 되므로 고속교환기의 실현이 어려운 방식이다. 입력큐방식은 크로스바(Crossbar)를 이용하고 교환기내의 동작속도를 전송속도로 동작시키기 위한 방식이지만 FIFO(First In First Out)를 입력큐로 사용할 경우에 발생하는 HOL-블로킹(Head Of Line Blocking)은 전체 시스템의 성능을 크게 저하시키는 요인으로 지적되었다[1].

HOL-블로킹에 의한 문제를 해결하고 높은 처리율을 얻기 위하여 입력포트에서의 여러 스케줄링 기법들이 제안되어왔으며 출력포트별 패킷을 관리할 수 있도록 VOQ(Virtual Output Queue)를 사용하는 방법들이 대표적이다[2]. 각 입력포트는 블로킹이 발생하지 않도록 VOQ를 선택해야하며 대부분의 기법에서는 별도의 스케줄러를 사용하여 높은 처리율을 실현하고 있다.

현재 제안된 입력큐에서의 스케줄링기법들은 크게 MSM(Maximum Size Matching)기법과 MWM(Maximum Weighted Matching)기법으로 나눌 수 있다[3][5]. MSM은 주로 입출력 쌍의 수가 최대가 되도록

* 正會員, 에스넷시스템㈜ 네트워크硏究所
(S Net systems Research Center)
** 正會員, 成均館大學敎 電氣電子컴퓨터工學部
(School of Electrical and Computer Engineering, Sungkyunkwan University)
接受日字:2000年4月4日, 수정완료일:2001年10月15日

접근하는 방법이다. MSM 기법의 대표적인 예는 PIM[2]과 iSLIP[4]등의 방법들이 있다. PIM과 iSLIP은 입력포트 별로 각 출력에 대한 VOQ를 이용하는 병렬처리 방식의 스케줄링이다. 이들은 간단하면서도 높은 처리율(Throughput)을 보장하지만 세션 또는 패킷별 상태를 고려하여 스케줄링하지 않으므로 *i.i.d* 프로세스형태의 트래픽이 보장되지 않는다면 starvation이 발생하여 공평성이 유지되지 못한다[5].

MWM은 VOQ별 또는 패킷별 우선순위를 부여하여 스케줄링을 진행하기 때문에 QoS(Quality of Service)를 지원할 수 있는 방법이다. MWM의 단점은 $O(N^3 logN)$의 높은 복잡도(Complexity)를 가지는 것이다. 이것을 보완하기 위해서 MWM에 근접하면서도 보다 낮은 복잡도를 가지는 기법들이 제안되고 있다. RPA[5]와 SIMP[6]는 복잡도가 $O(N^2)$ 이면서 순수 MWM기법에 근접한 성능을 나타내며, 특히 RPA는 세션(Session)별 QoS의 지원이 가능함을 보인다. 일반적으로 QoS에 관련하여 대역폭할당과 공평성을 유지하는 기법은 출력큐에서의 스케줄링기법에서 중요시되어 왔다.

출력큐에서 적용되고 있는 스케줄링기법은 WFQ (Weighted Fair Queueing)[7], SCFQ(Self-Clocked Fair Queueing)[8]등의 패킷별 시간표시(Time-stamp)를 기본으로 하는 기법과 WRR(Weighted Round Robin)[9], DRR(Deficit Round Robin)[10]등의 순환방식으로 나뉘어 지며 주로 페어큐잉(Fair Queueing)에 기초한 패킷 스케줄링 기법이 제안되었다. 이들의 우선순위 설정 방식을 입력큐에서의 스케줄링을 위한 우선순위로 활용할 수 있으나 출력큐에서와 다른 특성을 나타낼 수 있음을 고려해야한다.

입력큐에서 MWM 또는 MSM의 기법들을 이용하여 실제 구현된 스케줄러는 스케줄링기법이 분산 또는 병렬처리 형태를 유지한다고 하더라도 실제 대부분의 구현에 있어서는 중앙집중방식으로 구현된다. 높은 처리율을 가지는 BSB-교환기[11]의 경우에도 별도의 스케줄러를 구성하여 적용하였으며 분산 및 병렬처리 형태로 동작하고 높은 효율을 나타내는 iSLIP의 경우에도 실제 구현에 별도의 스케줄러를 구성하였다.

본 논문에서는 물리적으로 분산된 형태의 간단한 스케줄러를 제시하며 제안하는 기법은 다른 스케줄러의 구성의 경우와 마찬가지로 다음과 같은 성능지표를 만족시켜야한다.

- 복잡성(Complexity) 및 확장성(Flexibility) : 포트 수 증가에 따라 스케줄링에 필요한 시간과 실제 구현에 따른 비용이 크게 변하지 않아야 한다.
- 대기시간(Latency) 및 지터(Jitter) : 각 입력포트에서의 패킷들의 대기시간이 적어야하며 각 flow별 또는 패킷별 대기시간의 변화가 적어야한다.
- 공평성(Fairness) : 각 포트별 또는 세션별 서비스 비율의 일정함과 각 포트들을 통과하는 패킷들의 대기시간이 일정해야한다.
- 처리율(Throughput) : 입력방식의 낮은 효율을 제거하여 고속 교환을 가능하게 한다.

본 논문은 중앙집중방식의 별도의 스케줄러를 사용하지 않으면서 부분적으로 MWM기법을 수용하고 높은 처리율을 유지하는 입력포트에서의 스케줄링 기법인 PRR(Pipelined Round Robin)이라는 기법을 제안한다. 제안하는 기법은 각 VOQ별 또는 패킷별 우선순위를 이용하며 우선순위 부여 방법을 달리하여 성능을 확인하였다. PRR은 입력포트들에 분산된 형태로 스케줄러가 구성되어 하드웨어로의 구현이 간편하다는 장점이 있다. 본 논문의 구성은 Ⅱ장에서 제안된 PRR 스케줄링 기법을 서술하며 Ⅲ장에 시뮬레이션을 통하여 성능을 검증한 뒤 Ⅳ장에서 결론을 맺도록 한다.

Ⅱ. PRR 기법

각 입력포트는 블로킹이 발생하지 않도록 VOQ를 선택해야하며 대부분의 기법에서는 별도의 스케줄러를 사용하여 이를 방지하고 있다. VOQ별 우선순위가 부여되고 별도의 스케줄러를 사용하지 않고 스케줄링하기 위해서는 각 입력포트가 능동적으로 순서를 정하여 순차적으로 전체 스케줄링을 진행하여 입출력 쌍을 만들어야한다. 순차적인 스케줄링은 먼저 VOQ를 선택하여 스케줄링을 마친 특정 입력포트가 다른 입력포트에서 같은 출력포트를 나타내는 VOQ를 선택하지 않도록 선택한 VOQ번호를 알려야한다.

순차적인 스케줄링 기법은 스케줄링에 많은 시간을 요구하게 되는데 N×N 크기의 교환기의 경우 각 입력포트에서 VOQ를 선택하기 위하여 N-1번의 비교단계를 거치며 이를 N개의 입력포트들이 순차적으로 반복하게 되므로 $O(N^2)$의 복잡도가 요구되며 완벽한 MWM으로 접근하는 경우 $O(N^3 logN)$의 복잡도가 요

구된다.

PRR기법은 순차적인 스케줄링기법에서 전체 스케줄링에 걸리는 시간을 줄이기 위하여 파이프라인(pipeline)기법을 적용하였으며 이를 바탕으로 입력포트별 분산된 형태의 스케줄러를 구현할 수가 있다. 먼저 각 입력포트들의 순차적인 스케줄링을 위한 순서결정 방법을 설명하고 각 입력포트에서의 VOQ의 선택을 위한 우선순위의 부여방법에 대해서 논의한 뒤 파이프라인 기법이 적용된 순차적 스케줄링기법의 동작을 설명한다. 보통 패킷교환기는 패킷이 도착한 후 일정크기의 셀(cell)로 나누어 교환을 하므로 ATM과 같은 셀교환을 중심으로 설명하기로 한다.

A. 스케줄링 순서의 결정

입력큐교환기는 셀들이 도착하는 최소의 시간 간격인 time-slot마다 교환을 하게된다. 즉, 한 time-slot이내에 전체 스케줄링을 완결해야한다. 순차적인 스케줄링에서 입력포트들의 스케줄링 순서가 고정되어 있는 경우 특정 입력포트에 대하여 우선순위가 높게 고정되어 starvation 문제가 일어날 수 있으므로 각 time-slot마다 스케줄링순서는 변화해야한다. PRR에서의 스케줄링 순서의 결정은 순환방식(Round Robin)을 이용한다.

그림 1과 같이 각 입력포트는 내부에 순환방식의 카운터를 가지고 있으며 처음 time-slot에 자신의 포트번호로 초기화된다. 각 time-slot 마다 모든 입력포트의 카운터는 시계방향으로 하나씩 증가하며 이 때의 값이 해당 입력포트의 스케줄링 순서를 나타낸다. 각 time-slot마다 카운터의 값이 1인 입력포트가 스케줄링을 시작하며 이때의 입력포트를 Master 입력포트라고 정의한다. Master 입력포트로 정의된 이외의 입력포트들을 Slave 입력포트로 정의한다.

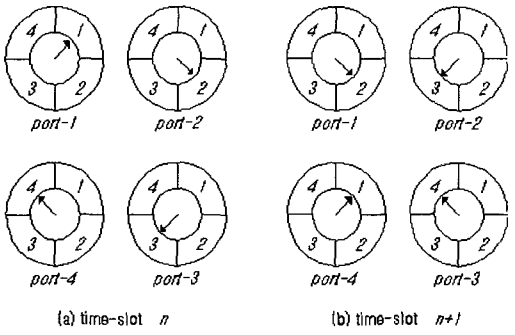


그림 1. 각 입력포트의 스케줄링순서의 결정
Fig. 1. Scheduling sequence of each input-port.

B. 각 입력포트에서의 VOQ의 선택기법

각 입력포트에 도착한 셀들은 VOQ로 나누어 저장되므로 스케줄링은 VOQ를 선택하는 것으로 생각할 수 있다. MWM기법의 스케줄링은 각 VOQ에 동적 우선순위를 부여함으로써 time-slot 마다 비교하여 선택하는데 i 번째 VOQ에 부여된 동적 우선순위를 P(i)라고 하면 가장 큰 P(i)를 선택하게 된다.

먼저 동적 우선순위를 부여하는 간단한 방법은 LQF (longest queue first)와 OCF(oldest cell first)가 있다[12]. 각 입력포트에서 L(i)를 i 번째 VOQ의 길이로 가정하고 W(i)를 i 번째 VOQ의 HOL에 위치한 셀의 대기시간으로 가정하면 LQF의 경우 P(i)=L(i)가 되며 OCF의 경우 P(i)=W(i)가 된다. LQF 기법을 사용하는 경우 i.i.d process를 엄격하게 따르지 않는 트래픽에 대해서는 starvation의 문제가 발생한다. OCF에서는 starvation의 문제를 극복하였다[12].

다른 기법으로는 QoS를 위하여 제시된 패킷 스케줄링 기법을 적용할 수 있다. 즉, 출력포트에서 사용되는 스케줄링 기법들을 입력포트에서 VOQ별 스케줄링 기법으로 적용하는 것이며 여러 기법들 중에서 WRR기법과 WFQ기법을 사용해보았다. WFQ기법은 가장 일반화된 패킷 스케줄링기법이며 WRR은 간결하고 패킷의 길이가 일정한 경우 높은 성능을 보여주는 기법이다.

본 논문에서는 입력포트에서의 각 VOQ는 같은 대역폭을 가지는 것으로 가정하며 WFQ, WRR에서의 가중치(weight)는 모두 같은 것으로 적용한다.

WRR기법은 각 세션별 가중치와 우선순위를 두게된다. 서비스를 받지 못한 플로우의 우선순위를 가중치만큼 더해 우선순위를 높여주며 서비스를 받은 세션의 우선순위를 0으로 하여 각 스케줄링마다 세션별 우선순위를 조정하게 된다.

WRR기법을 입력포트로 적용하는 경우 모든 VOQ의 대역폭이 같으므로 각 VOQ별 부여되는 가중치는 모두 1로 하였으며 우선순위는 일반적인 카운터로 구현 할 수 있다. 각 time-slot마다 서비스를 받은 VOQ의 카운터는 0으로 리셋(reset)되며 서비스 받지 못한 VOQ의 카운터는 1씩 증가한다.

VOQ의 카운터는 서비스를 받기 위해 기다린 time-slot을 나타낸다. 이는 WRR기법에서 모든 가중치 또는 우선순위의 증가율을 같게 설정한 기법과 동일하다. 각 VOQ의 우선순위를 나타내는 카운터의 값을 C(i)라

고 하면 P(i)=C(i) 이다.

WFQ기법은 모든 패킷별 우선순위를 부여하는 방법이다. 패킷별 우선순위는 해당 패킷이 전송을 마치게 되는 시간이며 이를 time-stamp라 한다. 실제 WFQ에서 세션-i에서 k번째로 도착한 패킷에 대한 time-stamp의 결정은 다음과 같다.

$$TS_i^k = \max\{ TS_i^{k-1},\ V(t)\} + \frac{l_i^k}{r_i} \qquad (1)$$

l_i^k는 세션-i에서 k번째로 도착한 패킷의 길이를 나타낸다 r_i는 세션-i가 할당받은 대역폭의 비율을 나타내며 가중치라고 할 수 있다. $V(t)$는 가상시간(virtual time)으로 전체 세션들이 평균적으로 서비스 받아야할 서비스 비율을 나타내며 세션의 수를 n이라고 한다면 $\frac{t}{n}$로 표현된다.

각 입력포트 스케줄링으로의 적용은 세션별 스케줄링을 VOQ별 스케줄링으로 적용한 것이다. 모든 VOQ가 서비스 받는 비율이 일정하고 도착하는 패킷의 길이가 일정함을 가정하면 모든 l_i^k/r_i는 같은 값이므로 1로 가정할 수 있으며 각 입력포트에 N개의 VOQ를 가지고 있는 경우 V(t)는 $\frac{t}{N}$이 된다.

N×N 교환기의 입력포트 j의 i번째 VOQ에 k번째로 도착하는 패킷의 time-stamp는 다음 식(2)와 같으며 특정 입력포트에서 P(i)= $\frac{1}{TS_i^k(j)}$ 이다.

$$TS_i^k(j) = \max\left\{ TS_i^{k-1}(j),\ \frac{t}{N} \right\} + 1 \qquad (2)$$

본 논문에서는 OCF, WRR, WFQ 기법을 적용하여 성능을 분석해 보았으며 WRR 기법을 사용한 PRR을 RR-PRR로 표시하고 WFQ의 경우 FQ-PRR로 표시하였다. 그리고 OCF를 적용한 PRR은 OCF-PRR로 표시하였다.

C. 파이프라인을 적용한 순차적 스케줄링

N×N 크기의 교환기에서 입력포트들은 내부의 VOQ의 우선순위를 순차적으로 비교하게되는데 이 경우 전체 스케줄링 시간의 증가에 대한 복잡도가 $O(N^2)$으로 정의된다. PRR은 전체 스케줄링시간에 대한 복잡도를 줄이기 위해 파이프라인 기법을 적용하여 복잡도를 O(N)으로 줄였다.

PRR 기법에서는 가장 먼저 스케줄링을 시작하는 Master 입력포트와 Master 입력포트의 스케줄링 시작 이후에 순차적으로 스케줄링을 시작하는 Slave 입력포트로 나뉜다. 이들의 동작은 다음과 같으며 그림 2, 그림 3에 pseudo-code 형식으로 나타내었다.

- Master 입력포트 : 각 time-slot에서 가장먼저 스케줄링을 시작하는 Master 입력포트는 내부에 구성된 모든 VOQ의 우선순위를 순차적으로 비교한다. 각 순차적인 비교에서 높은 우선 순위를 가지는 VOQ의 번호는 저장되고 낮은 우선순위를 가지는 VOQ의 번호는 다음 스케줄링을 시작하는 Slave 입력포트에게 전달된다.

- Slave 입력포트 : 직전에 스케줄링을 시작한 입력포트에서 선택되지 못한 VOQ 번호들을 받아 이들을 바탕으로 VOQ들의 우선순위를 비교한다. 각 비교에서 낮은 우선순위를 가지는 VOQ의 번호를 다음에 스케줄링을 시작하는 입력포트에 전달한다.

$I_i(j)$를 입력포트 i가 j번째로 스케줄링을 시작하는 것으로 표시하고 특정 time-slot n 에서 그림 1-a와 같이 스케줄링의 순서가 $\{I_1(1), I_2(2), I_3(3), I^4(4)\}$로 정해진다면 1번 입력포트부터 시작하여 4번째 입력포트까지 순차적으로 스케줄링 함을 나타낸다. 이때 $I_1(1)$이 Master 입력포트가 된다.

가장 먼저 스케줄링을 시작하는 입력포트 $I_1(1)$는 내부에 구성된 전체 VOQ에 대해서 순차적인 비교를 시작하며 매번 비교에서 낮은 우선순위를 가지는 VOQ의 번호를 다음 스케줄링을 시작하는 $I_2(2)$로 전송한다. $I_2(2)$는 $I_1(1)$에서 전송 받은 VOQ번호들에 대하여 순차적인 비교를 시작하며 $I_3(3)$는 $I_2(2)$에서 전송 받은 VOQ번호를 바탕으로 순차적인 비교를 진행한다. 가장 나중에 스케줄링을 시작하는 $I_4(4)$는 이전에 스케줄링을 시작한 모든 입력포트들이 VOQ를 선택한 경우 선택되지 못한 VOQ 하나를 선택하게된다.

그림 1에 관련하여 각 입력포트들의 VOQ 번호 전달 방향을 그림 4에 나타내었다. $I_1(1)$이외의 입력포트들은 직전에 스케줄링을 시작한 입력포트가 두 개의 VOQ를 비교한 직후 스케줄링을 시작한다. 전체 스케줄링에서의 각 입력포트에서의 비교와 이에 걸리는 시간을 그림 5에 나타내었다.

```
/*** Priority and Length of each VOQ ***/
P[Number_of_Ports];   //각 VOQ의 우선순위 배열
L[Number_of_Ports];   //각 VOQ의 길이 배열

/*** Initialize at each time-slot ***/
Selected_N = 0;       //선택된 VOQ의 번호 저장
Selected_P = 0;       //선택된 VOQ의 우선순위

/*** Master Scheduling ***/
For( k=1; k<=Number_of_Ports ; k++ ) {
    if ( ( P[k] > Selected_P ) && (L[k] > 0) ) {
        Put_to_NextPort( Selected_N );
        Selected_P = P[k];   Selected_N = k;
    } else Put_to_NextPort( k );
}
```

그림 2. Master로 동작하는 입력포트에서의 스케줄링
Fig. 2. Scheduling at master input-port.

```
/*** Priority and Length of each VOQ ***/
P[Number_of_Ports];   //각 VOQ의 우선순위 배열
L[Number_of_Ports];   //각 VOQ의 길이 배열

/*** Initialize at each time-slot ***/
Selected_N = 0;       //선택된 VOQ의 번호 저장
Selected_P = 0;       //선택된 VOQ의 우선순위 저장

/*** Slave Scheduling ***/
do {
   if( Check_Receive_from_Previous_Port() == True ) {
        k = Get_Receive_VoQ_Number();
        if ( ( P[k] > Selected_P ) && ( L[k] > 0 ) ){
            Put_to_NextPort( Selected_N );
            Selected_P = P[k];  Selected_N = k;
        } else Put_to_NextPort( k );
   }
} while( Check_Previous_Port_Scheduling_Complete() == NO)
```

그림 3. Slave로 동작하는 입력포트에서의 스케줄링
Fig. 3. Scheduling at slave input-port.

제안된 기법은 입력포트의 수가 N인 경우 2N-1번의 비교시간이 소모되므로 포트수 증가에 따라 요구되는 스케줄링 시간에 대한 복잡도는 $O(N)$이다. PRR은 각 입력포트가 스케줄러를 내장하고 있으므로 입력포트 수의 증가에 따라 스케줄러를 구성하는 회로가 증가하므로 포트수 증가에 따른 구현의 비용에 대한 복잡도는 $O(N)$이다.

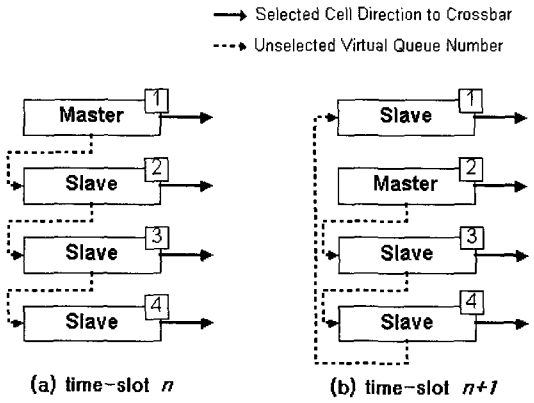


그림 4. 선택하지 않은 VOQ번호의 전달
Fig. 4. Transfer unselected VOQ's number.

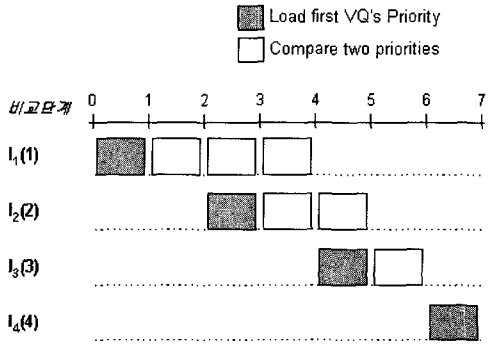


그림 5. 전체 스케줄링의 과정
Fig. 5. Overall scheduling process.

Ⅲ. 성능평가

성능평가는 시뮬레이션을 통하여 수행하였으며 크로스바를 사용하는 16×16 교환기를 대상으로 하였다. 시뮬레이터는 C++를 이용하여 제작하였고 Event-driven 시뮬레이션 방법을 이용하였다. 교환기에 입력되는 트래픽 모델은 Bernoulli 트래픽과 Bursty 트래픽을 이용하여 성능평가를 실시하였다.

PRR은 OCF, WRR, WFQ의 우선순위 부여방법에 따른 성능의 변화를 확인하였으며 4회 반복의 iSLIP을 함께 시뮬레이션을 하여 성능을 비교하였다. PRR은 Ⅱ-b에 설명한 우선순위 부여 방식에 따라 OCF를 적용한 경우 OCF-PRR로 표시하였고 WRR, WFQ를 적용한 경우를 각 각 RR-PRR, FQ-PRR로 표시한다.

성능은 평균대기시간과 지터 그리고 처리율에 대하여 비교하였다.

A. Bernoulli 트래픽에 대한 성능평가

성능평가는 16×16 교환기를 대상으로 하였으며 각 입력포트에 도착하는 패킷들의 출력포트로의 방향은 1/16의 일정한 확률로 발생하게 하였다. 제안된 기법들의 평균대기시간, 지터 그리고 처리율을 현재 가장 높은 성능을 가지는 iSLIP을 대상으로 비교하였다. 그림 6에 평균대기시간의 결과를 나타내었다.

평균 대기시간의 비교에서 RR-PRR를 제외하고 비슷한 성능을 확인 할 수 있다. 4회 반복회수를 가지는 iSLIP이 전반적으로 가장 좋은 성능을 나타내고 있으며 RR-PRR은 97%이상의 부하에서만 가장 좋은 성능을 나타낸다. FQ-PRR은 높은 부하일수록 상대적으로 낮은 성능을 나타낸다. 이는 출력큐 구조에서의 스케줄링기법이 입력큐 구조에서 사용되었을 때 문제점이 발생할 수 있음을 추측할 수 있다. OCF-PRR의 경우 전반적으로 iSLIP에 근접한 성능을 나타낸다.

그림 7에 평균지터를 나타내었다. 지터는 OCF-PRR의 성능이 전반적으로 가장 앞서고 있다. RR-PRR의 경우 평균대기시간에서와 마찬가지로 낮은 부하에서 상대적으로 좋지 않은 성능을 나타내고 있다.

90%이상의 높은 부하에서는 iSLIP의 경우 상대적으로 큰 평균지터 값을 가지게 되는데 셀 또는 VOQ의 상태를 고려하지 않기 때문인 것으로 추론 할 수 있다. FQ-PRR의 경우 대기시간에서와 마찬가지로 부하가 높아질수록 성능이 낮아진다.

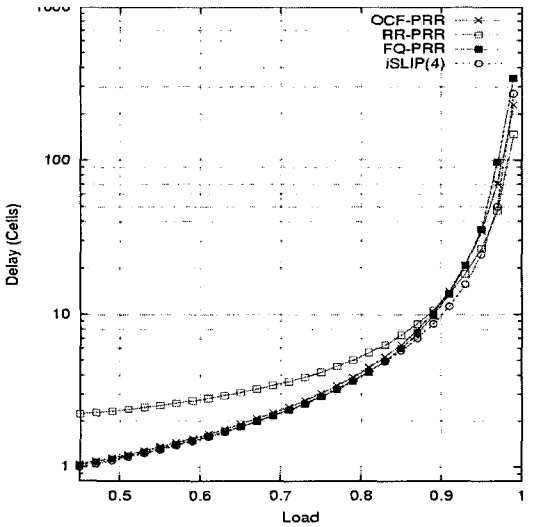


그림 6. Bernoulli 트래픽에서의 평균대기시간
Fig. 6. The average delay under Bernoulli arrivals.

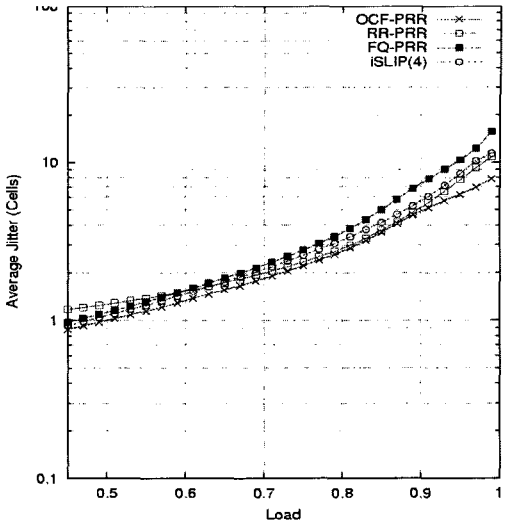


그림 7. Bernoulli 트래픽에서의 평균 지터
Fig. 7. The average jitter under Bernoulli arrivals.

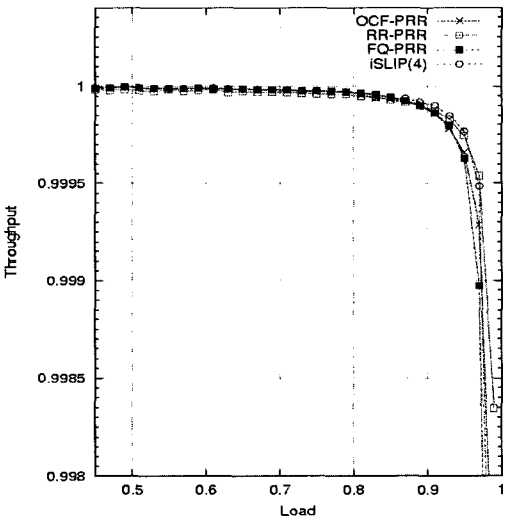


그림 8. Bernoulli 트래픽에서의 처리율
Fig. 8. The throughput under Bernoulli arrivals.

처리율은 제안된 기법과 iSLIP 모두 99%이상의 처리율을 가지는 것을 확인할 수 있다. FQ-PRR의 경우 높은 부하에서의 처리율이 상대적으로 떨어진다. 그림 8에 각 기법의 처리율을 나타내었다.

B. Bursty 트래픽에 대한 성능평가

실제 네트웍에서의 트래픽은 i.i.d 프로세스 형태를 유지하지 않으므로 bursty 트래픽에 대한 성능평가는 매우 중요하다. 트래픽의 생성은 on/off 트래픽에 기초하였으며 한번의 on구간에서는 셀들이 연속적으로 생성된다. on 구간에서의 연속적인 셀들은 같은 출력포트

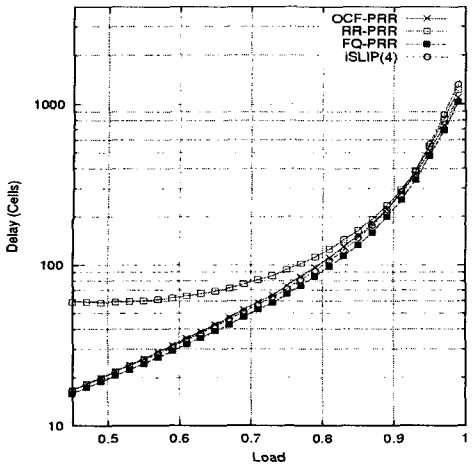


그림 9. $1/a$=16 인 경우의 평균 대기시간
Fig. 9. The average delay for $1/a$=16.

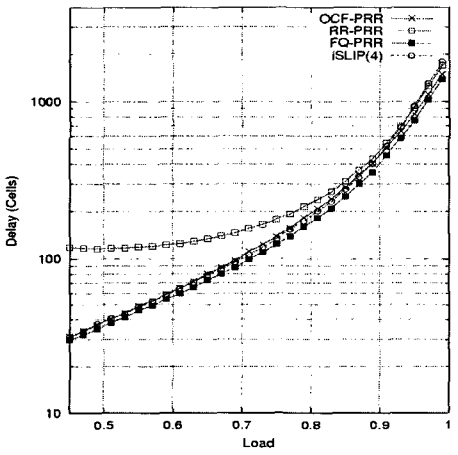


그림 10. $1/a$=32인 경우의 평균 대기시간
Fig. 10. The average delay for $1/a$=32.

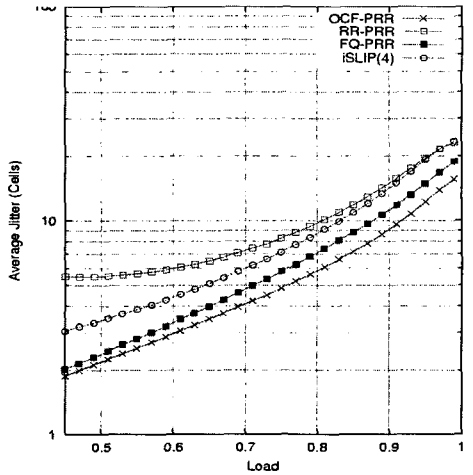


그림 11. $1/a$=16인 경우의 평균지터
Fig. 11. The average jitter for $1/a$=16.

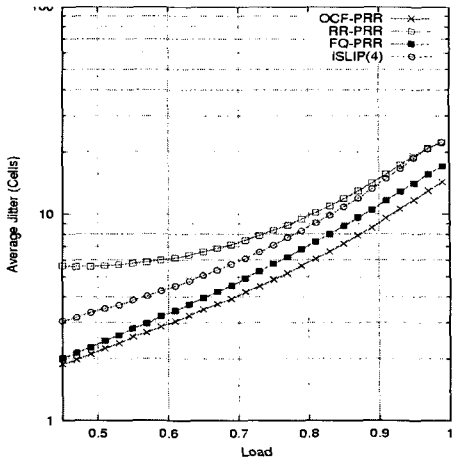


그림 12. $1/a$=32인 경우의 평균지터
Fig. 12. The average jitter for $1/a$=32.

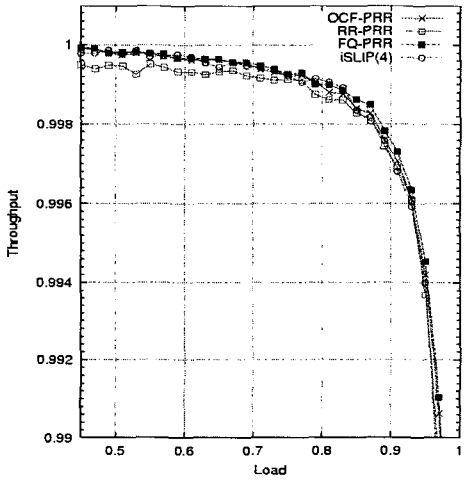


그림 13. $1/a$=16인 경우의 처리율
Fig. 13. The throughput for $1/a$=16.

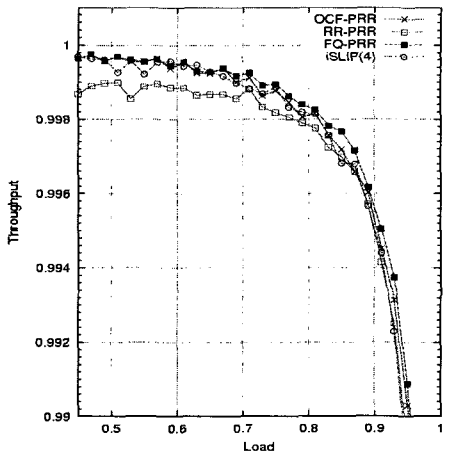


그림 12. $1/a$=32인 경우이 처리율
Fig. 12. The throughput for $1/a$=32.

를 향하는 packet train 모델을 사용하였다[13].

on 구간의 평균 길이를 1/α, off구간의 평균 길이를 1/β라고 정의하면 부하 ρ에 관련하여 ρ= α/(α+β)로 정의된다. 본 논문에서는 1/α을 16과 32 time-slot으로 고정하고 부하를 변화시켜 이에 따른 트래픽을 이용하였다.

Bernoulli 트래픽에서와 마찬가지로 셀들의 평균대기시간과 평균 지터 그리고 처리율을 중심으로 비교하였다. 그림 9, 10에 평균대기시간을 나타내었으며 그림 11, 12에 평균지터를 나타내었다.

Bursty 트래픽의 적용에서도 평균대기시간에 따른 성능은 FQ-PRR과 OCF-PRR이 상대적으로 높은 성능을 나타내며 지터특성에서도 상대적으로 좋은 결과를 보인다. RR-PRR은 상대적으로 가장 낮은 성능을 보이고 있다. 그림11, 12에 나타난 지터 특성은 큰 차이를 보이지 않고 있다.

그림 13, 14에 처리율을 나타내었으며 99%이상의 처리율을 확인 할 수 있다.

Ⅳ. 결 론

본 논문은 입력큐교환기에서 분산형태 스케줄러인 PRR을 제안하였다. PRR은 순차적인 스케줄링기법을 파이프라인으로 동작시켜 스케줄링에 필요한 시간을 단축시킨 기법이다. 스케줄링 시간에 관계한 복잡도는 $O(N)$으로 MSM과 MWM 사이의 복잡도를 가진다. 실제 하드웨어 구현에 따른 복잡도는 O(N)이며 이는 다른 스케줄러들이 $O(N^2)$이상인데 비하여 낮은 복잡도를 나타낸다.

PRR은 VOQ별 우선순위 부여방법으로 OCF, WFQ, WRR을 사용하였으며 나뉘며 OCF를 적용한 경우 트래픽변화에 따른 성능의 변화가 가장 적었다. OCF-PRR은 현재 최고의 성능을 가지는 MSM기법인 iSLIP과의 비교에서 약간 높은 대기시간을 가지게 되나 보다 좋은 지터특성을 가지며 FQ-PRR은 bursty 트래픽 환경에서의 대기시간과 처리율의 비교에서 가장 높은 성능을 나타낸다. WRR-PRR은 상대적으로 낮은 성능을 나타내었다.

PRR은 100%에 가까운 처리율을 가지며 스케줄러가 입력포트들에 분산되어 있으므로 크로스바를 사용하는 교환기에 폭넓게 적용 가능하다.

참 고 문 헌

[1] M. J. Karol, M. G. HLUCHYJ and S. P. Morgan, ,"Input versus Output Queueing Switch", IEEE Journal on Selected Areas in Communications, Vol. 9, No. 7, Sep. 1991, pp. 1347~1355.

[2] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High speed switch scheduling for local-area networks", ACM Transaction on Computer Systems, Nov. 1993, pp. 319~352.

[3] A. Mekkittikul and Nick McKeown, "Achieving 100% throughput in an input-queued switch", Proceedings of IEEE INFOCOM'96, 1996, pp. 296~302.

[4] A. Mekkittikul and Nick McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches", IEEE/ACM Transaction on Networking, Vol. 7, No. 2, April 1999, pp. 188~201.

[5] M. A. Marsan, A. Bianco, E. Leonardi, and L. Mila, "RPA : A Flexible Scheduling Argorithm for Input Buffered Switches", IEEE Transactions on Communications, Vol. 47, No. 12, December 1999, pp. 1921~1933

[6] R. Schoen, G. Post, and G. Sander, "Weighted arbitration algorithms with priorities for input-queued switches with 100% throughput", Proceedings of IEEE Broadband Switching Systems, 1999

[7] A. Demer, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm", Proceedings of ACM SIGCOMM 1989, pp. 1~12

[8] S. Golestani, "A self-clocked fair queueing scheme for broadband applications", Proceedings of IEEE INFOCOM'94, pp. 636~646. April 1994, pp. 636~646

[9] M. Katevenis, S. Sidiropouls, and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general purpose ATM switch chip," IEEE Journal of Selected Areas in Communications,

vol. 9, Oct. 1991, pp. 1265~79

[10] M. Shreedhar and George, "Varghese Efficient fair queueing using deficit round robin", Proceedings of ACM SIGCOMM 1995, pp.231~242

[11] H. Obara, S. Okamoto and Y. Mamazumi, "INOUT AND OUTPUT QUEUEING ATM SWITCH ARCHITECTURE WITH SPATIAL AND TEMPORAL SLOT RESERVATION CONTROL", IEE Electronics Letters, Vol. 28, No. 1, Jan. 1992, pp. 22~24.

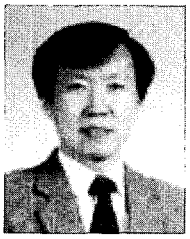
[12] A. Mekkittikul and Nick McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", Proceedings of IEEE INFOCOM'98, 1998, pp. 792~797.

[13] Raj Jain and Shawn A. Routher, "Packet Trains-Measurements and a New Model for Computer Network Traffic", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 6, September 1986, pp. 986~995

저 자 소 개

李 相 昊(正會員)

1997년 : 성균관대학교 제어계측공학과 졸업. 1999년 : 성균관대학교 대학원 전기전자컴퓨터공학부 졸업(공학석사, 네트워크). 2001년 : 성균관대학교 대학원 전기전자컴퓨터공학부 박사수료. 2001년~현재 : 에스넷시스템㈜ 네트워크연구소. <관심분야> 네트워크(트래픽관리, 네트워크관리, QoS)



申 東 烈(正會員)

1980년 : 성균관대학교 전자공학과 졸업. 1982년 : 한국과학기술원 전기 및 전자공학과 졸업(공학석사, 제어공학). 1992년 : Gerogia Inst. of Tech. 전기공학과(공학박사, 통신공학). 1982년 3월~1986년 7월 : 대우중공업 기술연구소, 주임연구원. 1992년 7월~1994년 2월 : 삼성SDS. 수석연구원. 1994년~현재 : 성균관대학교 전기전자컴퓨터공학부 부교수. 1999년~현재 : 중소기업 신기술, 장영실상, 및 KT 마크 심사위원. TR57 표준화 분과위원, 공조학회 smart home 분과위원. <관심분야> 컴퓨터네트워크(트래픽관리, 네트워크관리, VoIP), 자동화네트워크(MMS, ICCP), 실시간 통신, 큐잉네트워크(성능분석 및 모델링)