입력큐 교환기에서의 우선순위 파이프라인 순환 스케줄링

論 文
52D-6-6

Pipelined and Prioritized Round Robin Scheduling in an Input Queueing Switch

李 相 昊* · 申 東 烈**
(Sang-Ho Lee, Dong-Ryeol Shin)

Abstract - Input queued switch is useful for high bandwidth switches and routers because of lower complexity and fewer circuits than output queued. The input queued switch, however, suffers the HOL-Blocking, which limits its throughput to 58%. To overcome HOL-Blocking problem, many input-queued switch controlled by a scheduling algorithm. Most scheduling algorithms are implemented based on a centralized scheduler which restrict the design of the switch architecture. In this paper, we propose a simple scheduler called Pipelined Round Robin (PRR) which is intrinsically distributed by each input port. We presents to show the effectiveness of the proposed scheduler.

Key Words : input-queuing, switch, scheduling

1. 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 전체시스템의 성능을 크게 저하시키는 요인으로 지적되었다[1].

HOL-블로킹에 의한 문제를 해결하기 위해 입력포트에 출력포트별 버퍼를 구성하여 패킷 또는 셀을 임시 저장하는 VOQ(Virtual Output Queue)를 사용하는 구조가 제안되었으며 VOQ를 선택하기 위한 다양한 스케줄링 기법이 제안되었다. 제안된 스케줄링 기법들은 블로킹이 발생하지 않도록 VOQ를 선택해야하며 대부분의 기법에서는 별도의 스케줄러를 사용하여 높은 처리율(throughput)을 실현하고 있다[2].

현재 제안된 스케줄링기법들은 크게 MSM(Maximum Size Matching)기법과 MWM(Maximum Weighted Matching)기법으로 나눌 수 있다[3][5]. MSM은 주로 입출력 쌍의 수가 최대가 되도록 접근하는 방법이다. MSM 기법의 대표적인 예는 PIM[2]과 Iterative round-robin matching with SLIP (iSLIP)[4]등의 방법들이 있다. PIM과 iSLIP은 입력포트 별로 각 출력에 대한 VOQ를 이용하고 병렬처리방식의 스케줄링 기법이다. 이들은 간단하면서도 높은 처리율을 보장하지만 세션 또는 패킷별 상태를 고려하여 스케줄링하지 않으므로 각 플로우별 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 $O(N^3logN)$의 높은 복잡성(Complexity)를 가지는 것이다. 이것을 보완하기 위해서 MWM에 근접하면서도 보다 낮은 복잡성을 가지는 기법들이 제안되고 있다. RPA[5]와 SIMP[6]는 복잡성이 $O(N^2)$ 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.

입력큐 교환기에서 입출력 쌍을 만들기 위한 목적이 아닌 출력큐에서 QoS를 지원하기 위한 패킷 스케줄링기법이 있다. 패킷스케줄링은 Weighted Fair Queueing(WFQ)[7], Self-Clocked Fair Queueing(SCFQ)[8]등의 패킷별 시간표시(Time-stamp)를 기본으로 하는 기법과 Weighted Round Robin(WRR)[9], Deficit Round Robin(DRR)[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. BSB-교환기[11]의 경우에도 별도의 스케줄러를 구성하여 적용하였으며 분산 및 병렬처리 형태를 가지는 iSLIP의 경우에도 실제 구현에 별도의 스케줄러를 구성하였다.

본 논문에서는 물리적으로 분산된 형태의 간단한 스케줄러

* 正 會 員 : 三星電子 DS總括 研究員
** 正 會 員 : 成均館大學 情報通信工學科 副敎授 · 工博
接受日字 : 2002年 12月 30日
最終完了 : 2003年 5月 10日

를 제시하며 제안하는 기법은 다른 스케줄러의 구성의 경우와 마찬가지로 다음과 같은 성능지표를 만족시켜야한다.

- 복잡성 및 확장성(Scalability) : 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 : 입력큐방식의 낮은 효율을 제거하여 고속 교환을 가능하게 한다.

본 논문은 입력큐 교환기에서 중앙집중방식이 아니며 부분적으로 MWM기법을 수용하고 높은 처리율을 유지하는 스케줄링 기법인 PRR(Pipelined Round Robin)이라는 기법을 제안한다. 제안하는 기법은 각 VOQ별 또는 패킷별 우선순위 부여 방법을 달리하여 성능을 확인하였다. PRR은 입력포트들에 분산된 형태로 스케줄러가 구성되어 하드웨어로의 구현이 간편하다는 장점이 있다. 본 논문의 구성은 2장에서 제안된 PRR 스케줄링 기법을 서술하며 3장에 시뮬레이션을 통하여 성능을 검증한 뒤 4장에서 결론을 맺도록 한다.

2. PRR 기법

입력큐방식 교환기는 셀들이 도착하는 최소의 시간 간격인 time-slot마다 교환을 하게 된다. 즉, 한 time-slot이내에 전체 스케줄링을 완결해야한다. 간단한 스케줄링 기법으로 VOQ를 사용하고 N개의 입출력 포트를 가지는 입력큐방식의 교환기에서 각 입력포트가 순서를 정하여 출력포트를 하나씩 선택하는 순차적인 방식을 생각할 수 있다. 순차적인 방식은 각 입력포트에서 VOQ를 선택하기 위하여 N-1번의 비교단계를 거치며 이를 N개의 입력포트들이 순차적으로 반복하게 되므로 $O(N^2)$의 복잡성이 요구되며 완벽한 MWM으로 접근하는 경우 $O(N^3logN)$의 복잡성이 요구된다.

PRR은 순차적인 방식에서 복잡성을 낮추고 각 VOQ별 우선순위를 이용하기 위한 스케줄링 기법이다. 제안하는 스케줄링 기법은 입력포트의 순서를 정하는 Global Scheduling과 각 입력포트가 출력포트를 선택하는 Local Scheduling으로 나뉜다.

- Global Scheduling : Local Scheduling을 수행하는 입력포트들의 순서를 부여한다.
- Local Scheduling : 입력포트 별로 수행되며 앞서 다른 입력포트가 선택하지 않은 출력포트를 대표하는 VOQ들 중에서 가장 높은 우선순위를 가지는 VOQ를 선택한다.

N개의 입력포트가 존재하는 경우 Global Scheduling 수행 후 N번의 Local Scheduling이 발생하며 각 입력포트에서의 Local Scheduling은 앞서의 서로 다른 출력포트들을 대표하는 VOQ를 선택한다.

Global Scheduling과 Local Scheduling의 수행 방식에 따라 구현 및 수행시간의 복잡성이 달라질 수 있다. 먼저 Global Scheduling 기법을 설명하고 각 입력포트에서의 Local Scheduling을 설명한다. 일반적으로 패킷교환기는 패킷이 도착한 후 일정크기의 셀(cell)로 나누어 교환을 하므로 ATM과 같은 셀교환을 중심으로 설명하기로 한다.

2.1 Global Scheduling

Global Scheduling은 입력포트들이 출력포트를 선택하는 Local Scheduling의 수행 순서를 정하는 것이다. 먼저 Local Scheduling을 수행하는 입력포트는 출력포트 선택에 있어 우선권을 가지게 되는 것이므로 순서가 고정되어 있는 경우 특정 입력포트에 대하여 우선순위가 높게 고정되어 starvation 문제가 일어날 수 있다. starvation문제를 제거하기 위하여 Local Scheduling의 수행 순서는 변화해야한다.

Local Scheduling 수행순서는 각 입력포트 별 상태를 이용하는 방법과 고정적으로 순서를 변화하는 방법으로 나 눌 수 있다. 제안하는 PRR에서는 복잡성으로 고려하여 순환방식(Round Robin)을 이용한다.

그림 1과 같이 각 입력포트는 내부에 순환방식의 카운터를 가지고 있으며 자신의 포트번호로 초기화된다. 각 time-slot마다 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.

Global Scheduling은 각 입력포트 별 카운터로 구성되고 서로 독립적이므로 포트수 증가에 따른 복잡성은 O(1)이다.

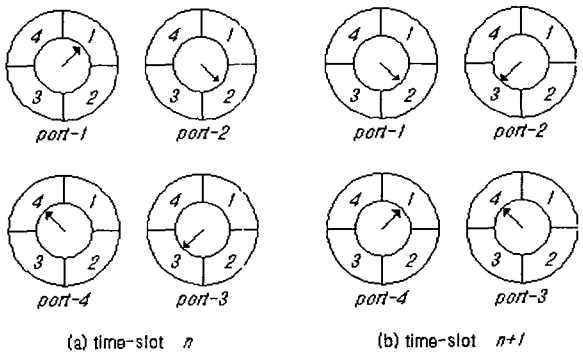

그림 1 전역 스케줄링
Fig. 1 Global Schedulingt

2.2 Local Scheduling

특정 입력포트에서의 Local Scheduling은 앞서 Local Scheduling을 수행한 입력포트들이 선택하지 않은 출력포트들을 선택한다. VOQ를 이용하는 구조에서 각 입력포트에서

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

/*** Load First VOQ's Priority ***/
 Selected_N = 0;
 Selected_P = P[0];

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

그림 2 Master로 입력포트의 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

/*** Load First Received VOQ's Priority ***/
k = Get_Receive_VoQ_Number();
Selected_P = P[k];

/*** compare Priority ***/
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

수행되는 Local Scheduling은 N개의 VOQ 우선순위를 비교한다. Global Scheduling이후 N번의 Local Scheduling이 진행되므로 포트 수 증가에 따른 VOQ 비교 횟수는 N^2 이다. 즉, 포트 수 증가에 따른 전체 스케줄링 시간의 복잡성은 O (N^2)이 된다.

PRR은 N번의 Local Scheduling에 걸리는 시간을 줄이기 위하여 Pipeline을 적용하였다. 각 입력포트에서의 Local Scheduling은 두개의 VOQ 우선순위를 순차적으로 N-1번 비교한다. 각 VOQ의 비교에서 낮은 우선순위를 가지는 VOQ를 다음 Local Scheduling을 시작하는 입력포트에 보내는 방식이다. 각 입력포트는 먼저 Local Scheduling을 수행한 입력포트로부터 선택 가능한 VOQ 리스트를 받게 된다.

Global Scheduling에서 결정된 순서에 따라 가장 먼저 스케줄링을 시작하는 Master 입력포트와 이후에 순차적으로 스케줄링을 시작하는 Slave 입력포트의 동작은 다르다. 이들의 동작은 다음과 같으며 그림 2, 그림 3에 pseudo-code 형식으로 나타내었다.

• **Master 입력포트** : 각 time-slot에서 가장 먼저 스케줄링을 시작하는 Master 입력포트는 내부에 구성된 모든 VOQ의 우선순위를 순차적으로 비교한다. N개의 VOQ를 두개씩 순차적으로 비교한다. 각 순차적인 비교에서 낮은 우선순위를 가지는 VOQ의 번호는 다음 Local Scheduling을 시작하는 Slave 입력포트에게 전달된다.

• **Slave 입력포트** : 앞서 Local Scheduling을 시작한 입력포트에서 선택되지 못한 VOQ 번호들을 받아 이들을 바탕으로 VOQ들의 우선순위를 비교한다. 각 비교에서 낮은 우선순위를 가지는 VOQ의 번호를 다음에 Local Scheduling을 시작하는 slave 입력포트에 전달한다. 직전에 Local Scheduling이 시작된 입력포트가 두 개의 VOQ를 비교한 직후 Local Scheduling을 시작한다.

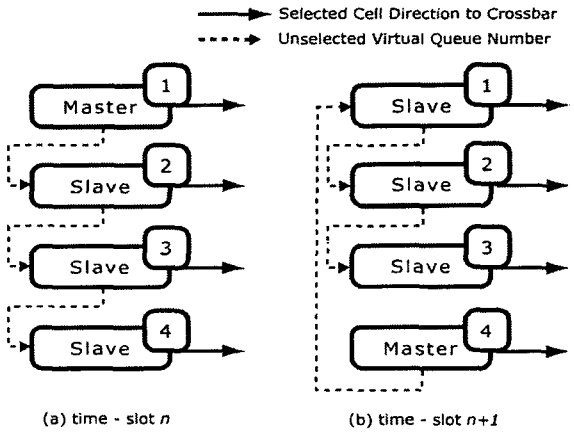

그림 4 선택하지 않은 VOQ번호의 전달.
Fig. 4 Transfer unselected VOQ's number

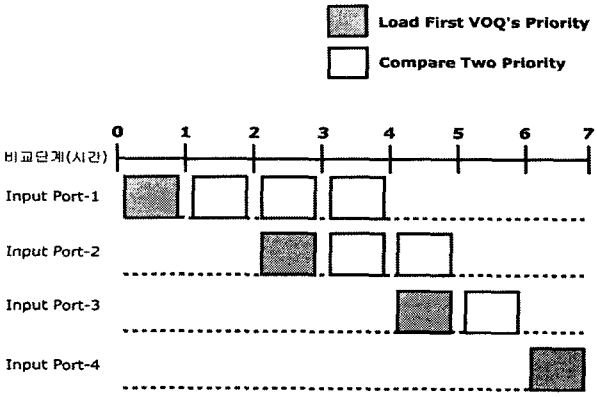

그림 5 전체 스케줄링의 과정
Fig. 5 Overall scheduling process

그림 1와 같이 Global Scheduling이 수행됨에 따라 각 입력포트들이 Local Scheduling을 수행하는 경우 선택하지 않은 VOQ 번호의 전달방향을 그림 4에 나타내었으며 그림 5는 그림 4-a와 같이 Local Scheduling이 수행되는 경우 각 입력포트의 Local Scheduling 진행을 나타내었다.

그림 5에서 Master 입력포트로 지정된 1번 입력포트는 내

부에 구성된 전체 VOQ에 대해서 순차적인 비교를 시작하며 매번 비교에서 낮은 우선순위를 가지는 VOQ번호를 다음 스케줄링을 시작하는 2번 입력포트로 전송한다. 2번 입력포트는 1번 입력포트로부터 전송받은 VOQ번호를 받아 VOQ 우선순위를 비교한다. 가장 마지막에 Local Scheduling을 수행하는 4번 입력포트는 앞서 3개의 입력포트에서 선택하지 않은 하나의 VOQ번호를 받게 되어 이를 선택하게 된다.

Slave 입력포트에서 Local Scheduling은 앞서 Local Scheduling에서 최초 VOQ 비교이후 즉시 시작되므로 전체 N번의 Local Scheduling에 2N-1번의 VOQ 비교시간이 소모되며 복잡성은 *O(N)*이다. 또한 PRR은 각 입력포트가 Local Scheduler와 Global Scheduler를 내장하고 있으므로 포트수 증가에 따른 전체 스케줄러 구현에 대한 복잡성은 *O(N)*이다.

2.3 VOQ별 우선순위 부여방법

Local Scheduling은 우선순위가 가장 높은 VOQ를 선택하는 것이다. 우선순위는 모든 VOQ가 공평한 서비스 율(service rate)를 얻도록 동적으로 각 VOQ의 상태를 반영해야한다. 우선순위는 LQF(Longest Queue First)와 OCF (Oldest Cell First)와 같은 단순한 기법과 QoS를 위하여 제안된 패킷 스케줄링 기법을 적용 할 수 있다.

LQF는 가장 긴 VOQ를 선택하는 것이며 OCF는 HOL에 위치한 셀의 대기시간이 가장 높은 VOQ를 선택하는 기법이다. 이용한다. LQF는 많은 셀이 집중되는 VOQ를 계속 선택하는 starvation이 발생한다. 즉, 각 플로우별 *i.i.d process*를 엄격하게 따르지 않는 트래픽에 대해서는 문제가 발생한다. OCF에서는 starvation의 문제를 극복하였다[12].

QoS를 위하여 제안된 패킷 스케줄링 기법으로는 WFQ 기법과 WRR 기법이 대표적이다. WRR기법은 각 세션별 가중치를 이용하여 플로우별 서비스율을 제어 할 수 있다. 서비스를 받지 못한 플로우의 우선순위를 가중치만큼 더해 우선순위를 높여주며 서비스를 받은 세션의 우선순위를 0으로 하여 각 스케줄링마다 세션별 우선순위를 조정하게 된다.

WRR기법을 PRR에 적용하는 경우 모든 VOQ의 서비스율이 같아야 하므로 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.

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.

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

l_i^k는 *세션-i*에서 k번째로 도착한 패킷의 길이를 나타낸다. w_i는 *세션-i*가 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.

WRR의 적용과 같이 각 VOQ는 같은 서비스율을 가져야 한다. 각 VOQ는 같은 대역폭 비율을 할당받고 셀단위의 교환을 하므로 식(1)은 식(2)로 나타낼 수 있다. 또한 V(t)는 $\frac{t}{N}$이 된다. 이 때 i는 VOQ의 번호 이며 k는 해당 VOQ에 도착한 셀의 순서로 나타낸다. C는 Constant 이다.

$$TS_i^k = \max\{ TS_i^{k-1}, V(t)\} + C \qquad (2)$$

본 논문에서는 OCF, WRR, WFQ 기법을 적용하여 성능을 분석해 보았으며 각 각 OCF-PRR, RR-PRR, FQ-PRR로 표시하며 시뮬레이션을 통하여 각 각의 성능을 평가하였다.

3. 시뮬레이션

3.1 시뮬레이션 모델 및 수행 방법

PRR을 적용하는 대상은 크기가 16× 16 이고 내부에 버퍼를 가지지 않는 크로스바 스위치이다. 크로스바 스위치는 일정한 시간간격인 time-slot마다 특정 입력과 출력의 물리적 연결이 유지되며 이때 일정길이의 셀들이 교환된다.

PRR은 OCF, WRR, WFQ를 적용하여 iSLIP과 비교하였다. iSLIP은 병렬 스케줄링의 반복횟수, i,에 따라 처리율이 높아진다. i 의 적정값은 N× N 크기의 스위치에서 식(3)와 같이 표현된다[4].

$$E[I] \leq log_2 N \qquad (3)$$

시뮬레이션에서 N의 크기는 16 이므로 제안된 기법과의 비교대상의 iSLIP은 4회 반복으로 동작한다.

트래픽 모델은 일정 길이의 셀이 유입되는 교환과 가변적인 길이의 패킷이 유입되는 두 가지 상황을 표현하기 위하여 Bernoulli 트래픽과 Bursty 트래픽의 두 가지 모델을 적용하였다.

- **Bernoulli 트래픽** : 각 입력포트에 도착하는 셀의 간격은 Poisson Arrival을 따르며 각 셀의 출력포트 방향은 Uniform Distribution을 가진다.
- **Bursty 트래픽** : ON-OFF모델에 기초하며 ON 구간에서 셀은 연속적으로 발생한다. OFF 구간에서는 발생하지 않는다. 하나의 ON 구간에서 발생되는 셀의 출력포트 방향은 모두 일정하며 하나의 패킷으로 설명될 수 있다. 본 논문에서는 각 구간의 길이가 Poisson Distribution을 따르는 Packet Train Model을 사용한다[13]. 평균 길이가 1/α, off구간의 평균 길이를 1/β라고 정의하면 부하 ρ에 관련하여 ρ= α/(α+β)로 정의된다. 본 논문에서는 1/α을 16과 32로 고정하고 1/β을 변화시켜 부하를 조절하였다.

3.2 시뮬레이션 결과

3.2.1 Bernoulli 트래픽에서의 처리성능

그림 6에 평균대기시간의 결과를 나타내었다. 평균 대기시간의 비교에서 RR-PRR를 제외한 PRR은 iSLIP과 비슷한 성능을 가지나 iSLIP이 전반적으로 가장 좋은 성능을 나타내고 있다. 이는 모든 VOQ에 동일한 비율로 셀들이 도착하는 상황에서 입출력쌍을 최대한 많이 생성하는 MSM 기법이 유리함을 보여준다.

RR-PRR은 97% 이상의 높은 부하에서 다른 기법들과 나타낸다. FQ-PRR은 높은 부하일수록 상대적으로 낮은 성능을 나타낸다. OCF-PRR의 경우 전반적으로 iSLIP에 근접한 성능을 나타낸다.

그림 7에 평균지터를 나타내었다. 모두 비슷한 특성을 가지나 각 셀의 도착시간을 이용하는 OCF-PRR의 성능이 전반적으로 가장 앞서고 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 8에 각 기법의 처리율을 나타내었다. 제안된 기법과 iSLIP 모두 99%이상의 처리율을 가지는 것을 확인할 수 있다. 97%이상의 부하에서만 낮은 평균대기시간을 가지는 RR-PRR이 높은 처리율을 가진다. FQ-PRR의 경우 높은 대기시간과 지터에서와 같이 높은 부하에서의 처리율이 상대적으로 떨어진다.

3.2.2 Bursty 트래픽에서의 처리성능

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

Bursty 트래픽의 적용에서는 평균대기시간에 따른 성능은 FQ-PRR이 iSLIP보다 높은 성능을 가진다. 지터는 OCF-PRR과 FQ-PRR이 iSLIP보다 높은 성능을 가진다. Bursty 트래픽 환경에서는 예상 전송시간과 플로우의 상태를 함께 고려하는 우선순위 부여기법이 적절함을 알 수 있다.

실제 가변적인 길이를 가지는 패킷은 연속된 셀들로 변환되며 연속적으로 처리되어야한다. 비슷한 시기에 들어온 셀들이 연속적으로 전송되기 위해서 VOQ별 서비스 율을 조절하는 방식보다 각 셀의 대기시간을 고려하는 것이 유리하다.

그림 13, 14에 처리율을 나타내었다. FQ-PRR과 OCF-PRR이 대기시간과 지터특성의 결과와 마찬가지로 상대적으로 높은 성능을 가진다.

4. 결 론

본 논문은 입력큐교환기에서 분산형태 스케줄러인 PRR을 제안하였다. PRR은 순차적인 스케줄링기법을 파이프라인으로 동작시켜 스케줄링에 필요한 시간을 단축한 기법이다. 스

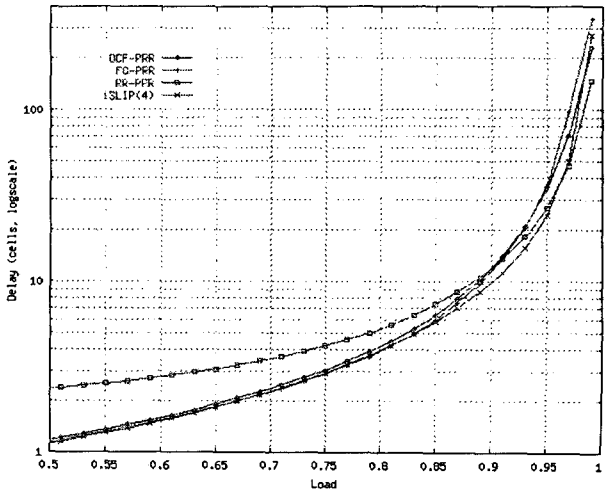

그림 6 Bernoulli 트래픽에서의 평균대기시간
Fig. 6 The average delay under Bernoulli arrivals

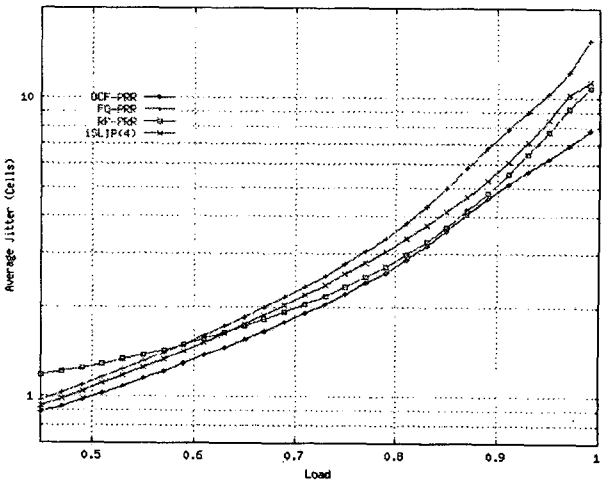

그림 7 Bernoulli 트래픽에서의 평균 지터
Fig. 7 The average jitter under Bernoulli arrivals

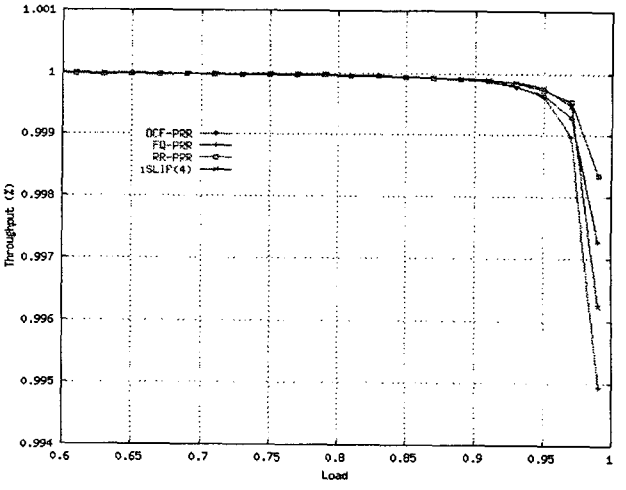

그림 8 Bernoulli 트래픽에서의 처리율
Fig. 8 The throughput under Bernoulli arrivals

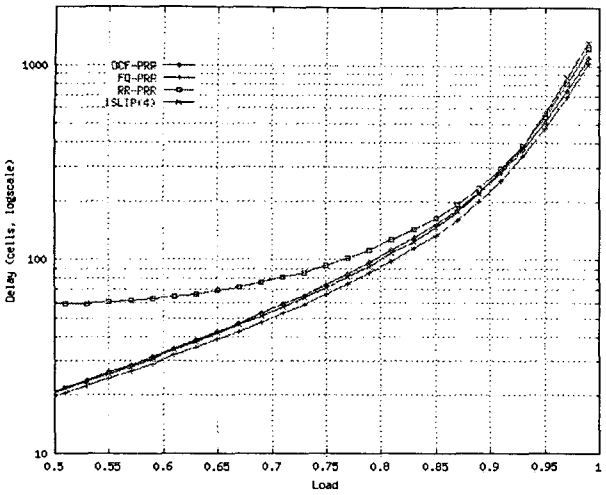

그림 9 1/α=16 인 경우의 평균 대기시간
Fig. 9 The average delay for 1/α=16

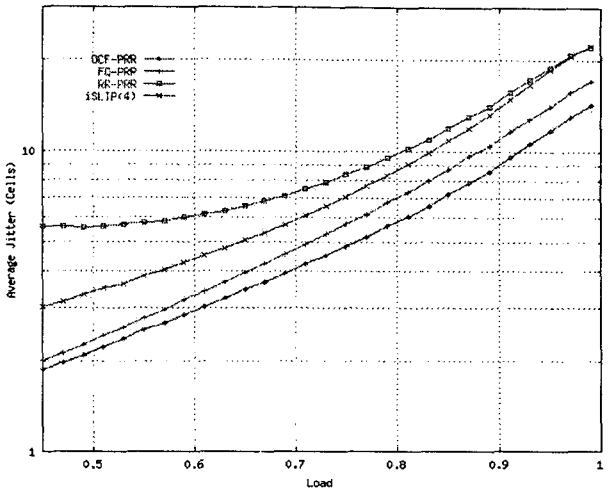

그림 12 1/α= 32인 경우의 평균 지터
Fig. 12 The average jitter for 1/α = 32

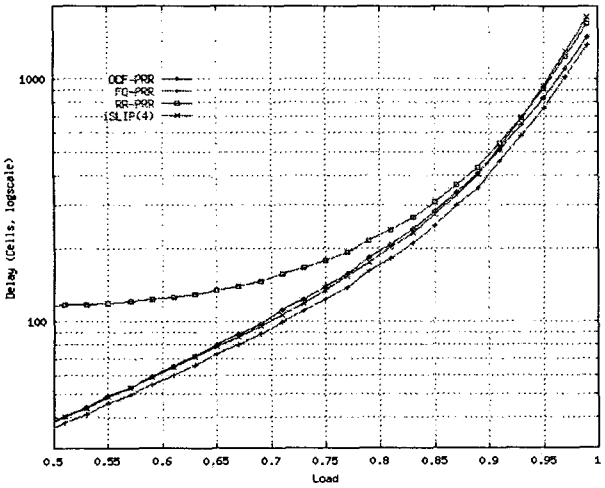

그림 10 1/α= 32인 경우의 평균 대기시간
Fig. 10 The average delay for 1/α = 32

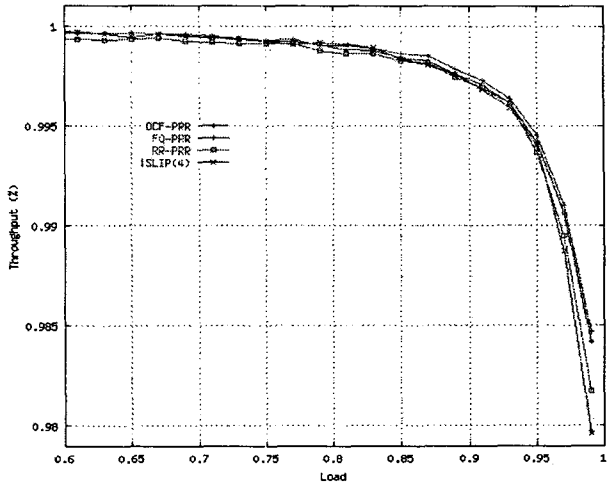

그림 13 1/α= 16인 경우의 처리율
Fig. 13 The throughput for 1/α = 16

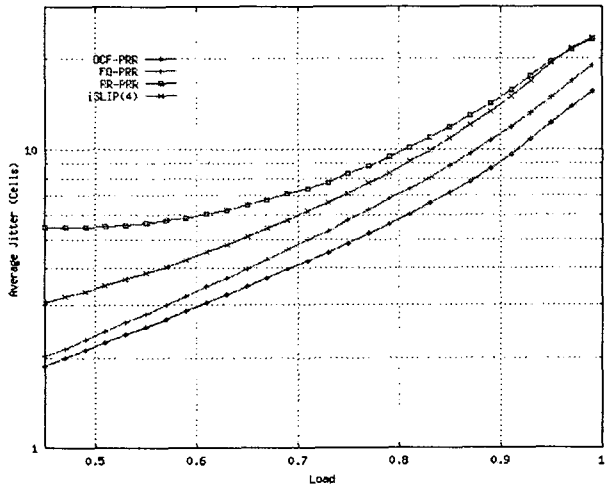

그림 11 1/α= 16인 경우의 평균지터
Fig. 11 The average jitter for 1/α = 16

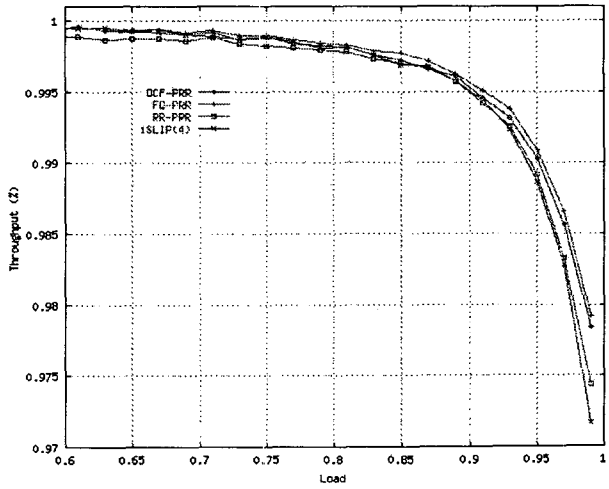

그림 14 1/α= 32인 경우의 처리율
Fig. 14 The throughput for 1/α = 32

케줄링 시간에 관계한 복잡성은 $O(N)$으로 MSM과 MWM 사이의 복잡성을 가진다. 스케줄러는 입력포트별로 모듈화가 가능하다. 이는 실제 하드웨어 구현에 있어 사용되는 게이트(gate) 수의 복잡성 $O(N)$을 따르게 한다. 이는 현재 구현되어 적용되고 있는 iSLIP이 $O(N^2)$이상인데 비하여 낮은 복잡성을 나타내며 실제 구현에 있어 보다 유리하다[4].

PRR은 VOQ별 우선순위 부여방법에 따라 성능의 차이를 가진다. 본 논문에서는 OCF, WFQ, WRR을 적용하여 MSM 기법을 대표하는 iSLIP과 비교 평가하였다. 성능은 Bernoulli 트래픽에서 iSLIP이 보다 우수한 성능을 가지게 되나 OCF와 WFQ를 적용한 PRR은 실제 패킷환경과 같은 Bursty 트래픽에서 iSLIP보다 높은 성능을 가진다. 특히 OCF를 적용한 경우 트래픽변화에 따른 성능의 변화가 가장 적었으며 좋은 지터특성을 가지고 있다. WFQ를 적용한 경우 대기시간과 처리율에 있어 좋은 성능을 가진다.

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. Courcousbetis, "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

이 상 호(李 相 昊)

1974년 6월 8일생. 1997년 성균관대학교 제어계측공학과 졸업. 2001년 동 대학원 전기전자 및 컴퓨터 공학과 박사과정 수료. 2001년~2002년 에스넷시스템 네트워크 연구소. 2003년~현재 삼성전자 DS총괄 SYSTEM LSI 사업부 선임연구원
Tel : 031-209-8482, Fax : 031-209-6585
E-mail : sangho74.lee@samsung.com

신 동 렬(申 東 烈)

1957년 6월 25일생. 1980년 성균관대학교 전자공학과 졸업. 1982년 한국과학기술원 전기 및 전자공학과 졸업(공학석사). 1992년 Georgia Tech. 전기공학과 졸업(공학박사). 1992년~1994년 삼성 SDS 수석연구원. 1994년~현재 성균관대학교 부교수
Tel : 031-290-7125, Fax : 031-290-7231
E-mail : drshin@ece.skku.ac.kr