다중 입력버퍼 방식의 고속 패킷 스위치

김학용

광주광역시 북구 오룡동 1, 광주과학기술원 정보통신공학과
Tel: 062-970-2265, Fax: 062-970-2204
Web: http://charly.kjist.ac.kr/~hykim/
E-mail: hykim@ieee.org *or* hykim@kjist.ac.kr

요약

인터넷을 필두로 한 폭발적인 데이터 트래픽의 증가는 광대역 전송뿐만 아니라 고속 스위칭 기능을 요구하고 있다. 고속 특성을 가지고 있는 입력버퍼형 패킷 스위치는 입력포트에 있는 FIFO 큐에서 발생하는 Head-of-Line (HOL) 블록킹으로 인해 그 수율이 58%로 제한되는 문제점이 있다. 이러한 문제를 해결하기 위해 개발된 것이 다중 입력버퍼 방식(MIQ)의 고속 패킷 스위치이다. MIQ 스위치는 각 입력포트에 m 개의 FIFO 메모리를 두고, 각 메모리를 특정 출력포트 혹은 일단의 출력포트들로 하여금 공유하도록 하는 방식이다. 이렇게 함으로써, 입력버퍼형 스위치의 고속 동작 특성을 유지하게 되고 스위치의 수율을 향상시킬 수 있다. 본고에서는 다중 입력버퍼 방식의 특징에 대해 살펴보고 관련 스케쥴링 알고리즘에 대해 살펴 보겠다.

1 머릿말

인터넷 사용자의 폭발적인 증가와 다양한 멀티미디어 서비스의 등장은 전송되는 트래픽 양을 폭발적으로 증가시켰으며, 전송망의 성능향상을 요구하고 있다. 전송망이 광케이블과 같은 전송의 부분과 스위치 및 라우터와 같은 교환의 부분으로 구성된다고 할 수 있으며[1], 전송의 부문에 있어서는, 광케이블이 넓은 대역폭을 제공하고 있고 교환 부문에서는 고속 ATM 스위치 및 기가비트 라우터 등이 그 역할을 다하고 있다.

고속 ATM 스위치 및 고속 라우터는 시스템적인 측면에서 볼 때, 그림 1에서 보이는 것처럼, 실제로 ATM cell이나 데이터 패킷의 교환이 이루어지는 스위칭 패브릭과 교환이 이루어지기 전 혹은 후에 ATM cell이나 데이터 패킷을 잠시 저장해 두는 메모리 부분, 그리고 메모리 내의 데이타의 효율적인 전송을 위한 제어 부분(control unit)으로 크게 나눌 수 있다[2].

스위치 패브릭은 일반적으로 크로스바(crossbar) 스위치와 같은 공간스위치(space switch)가 사용되며, 동시에 입력 혹은 출력포트 수만큼의 연결을 제공할 수 있다. 제어 부분은 스위치 패브릭의 동작 및 메모리들 사이의 스케줄링과 같은 기능을 지원하는 일을 담당하고 있다. 그림에서처럼 하나의 제어부가 존재할 수도 있으며, 기능에 따라 스위칭 패브릭 및 입/출력포트들에 흩어져서 존재할 수도 있다.

메모리는 그림 1에서처럼 스위치 패브릭의 앞, 뒤, 혹은 그림에는 보이지 않지만, 패브릭 사이에 위치할 수가 있으며, 각각 입력버퍼, 출력버퍼, 공유버퍼 방식이라 불린다. 이중에서 출력버퍼 방식이 성능 측면에서 가장 효율적인 것으로 알려져 있으나, 입력 혹은 출력포트 수만큼 빨리 동작해야 하는 단점이 있다. 공유버퍼 방식은 입력 및 출력포트가 하나의 커다란 메모리를 공유(share)하는 방식으로 적은 메모리의 양으로도 출력버퍼 방식의 스위치와 비슷한 성능을 제공할 수 있으며, 외부 링크 속도보다 두 배 빨리 동작해야 한다. 마지막으로, 본고에서 다룰 (단일)입력버퍼 방식은 외부 링크와 똑같은 속도로 동작하므로 고속 스위치를 만들기에 적합하

[1] 본고에서는 특별한 언급이 없는한 스위치 및 라우터를 같은 개념으로 사용할 것이며, 가능한한 스위치 라는 말을 사용할 것이다.
[2] 본고에서는 ATM cell과 데이터 패킷을 같은 개념으로 사용할 것이다.

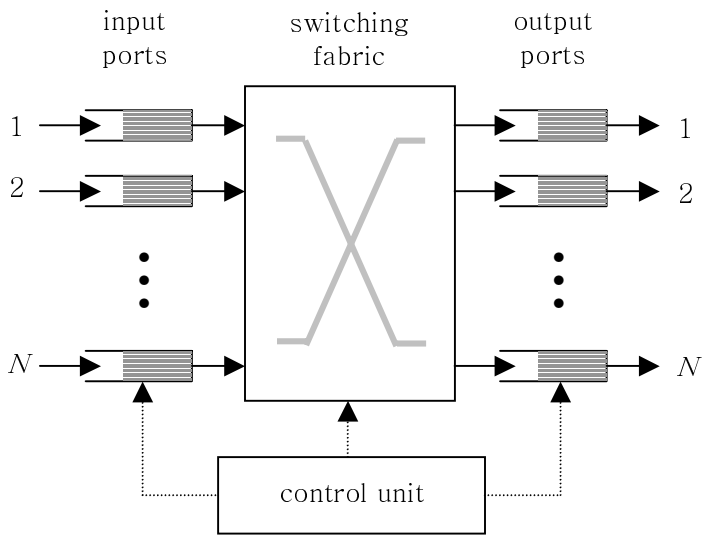

Fig. 1 고속 ATM 스위치 및 라우터의 구조.

고 하드웨어 구현이 간단한 것으로 알려져 있다. 문제는 HOL 블록킹 현상으로 인해 수율이 58%로 제한된다는 것이다 [1].

최근에는 인터넷이 보편적인 서비스로 확산되어감에 따라, 교환되는 데이터 트래픽의 양이 폭발적으로 증가하고 있고, 이러한 추세는 네트웍의 고속화를 요구하고 있다. 따라서, 여러 버퍼링 방식 중에서 입력버퍼 방식에 많은 관심이 모아지고 있다. 특히, 입력버퍼형 스위치의 문제점인 HOL 블록킹으로 인한 낮은 수율을 향상시키기 위한 버퍼링 및 그와 관련된 스켈쥴링 방식들에 대한 연구가 집중적으로 이루어지고 있다. 따라서, 본고에서는 다중입력버퍼(MIQ: multiple input-queue) 방식의 고속 패킷 스위치/라우터의 특성에 대해 살펴볼 것이다.

본고의 구성은 다음과 같다. 2 장에서는 MIQ 스위치를 정의하고 관련된 연구들을 소개할 것이다. 3 장에서는 MIQ 스위치의 성능을 분석할 것이다. 4 장에서는 MIQ 스위치에 사용되어지는 셀 단위(cell-based)의 스케쥴링 알고리즘에 대해 살펴볼 것이다. PIM 및 PIM에 기반한 알고리즘을 특징 위주로 설명할 것이다. 5 장에서는 VOQ 스위치의 문제점을 살펴볼 것이다.

2 MIQ 스위치의 정의 및 분류

다중 입력버퍼(MIQ: multiple input-queued) 방식의 스위치는 이름이 나타내는 것처럼 각 입력포트마다 여러 개의 FIFO 메모리를 가지고 있는 스위치를 말한다. 스위치의 포트 수가 N이고 한 입력포트내의 메모리의 수를 m이라 할 때, $1 \leq m \leq N$ 과 같은 관계가 있다. 그림 2는 MIQ 스위치의 입력포트 i의 구조를 보여주고 있다. $m = 1$ 일 때는 하나의 큐가 모든 출력으로 향하는 셀들을 저장하게 되므로 기존의 단일 입력버퍼(SIQ: single input-queued) 방식의 스위치와 동일하게 된다. $m = 2$ 일 때는 일반적으로 특정 입력포트 내의 하나의 FIFO 메모리는 짝수의 출력포트들로 향하는 셀 혹은 패킷들을 저장하고 다른 하나의 FIFO 메모리는 홀수의 출력

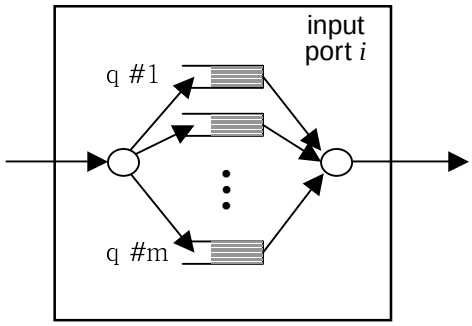

Fig. 2 MIQ 스위치의 입력 버퍼의 구성

출력포트들로 향하는 패킷들을 저장하도록 약속을 하게 되며, 이런 이유로 Odd-Even 버퍼링 방식이라 불린다 [2, 3]. 또한, $m = N$ 일 때는 각 FIFO 메모리가 특정한 하나의 출력포트에 대해 할당되어서 마치 출력버퍼 방

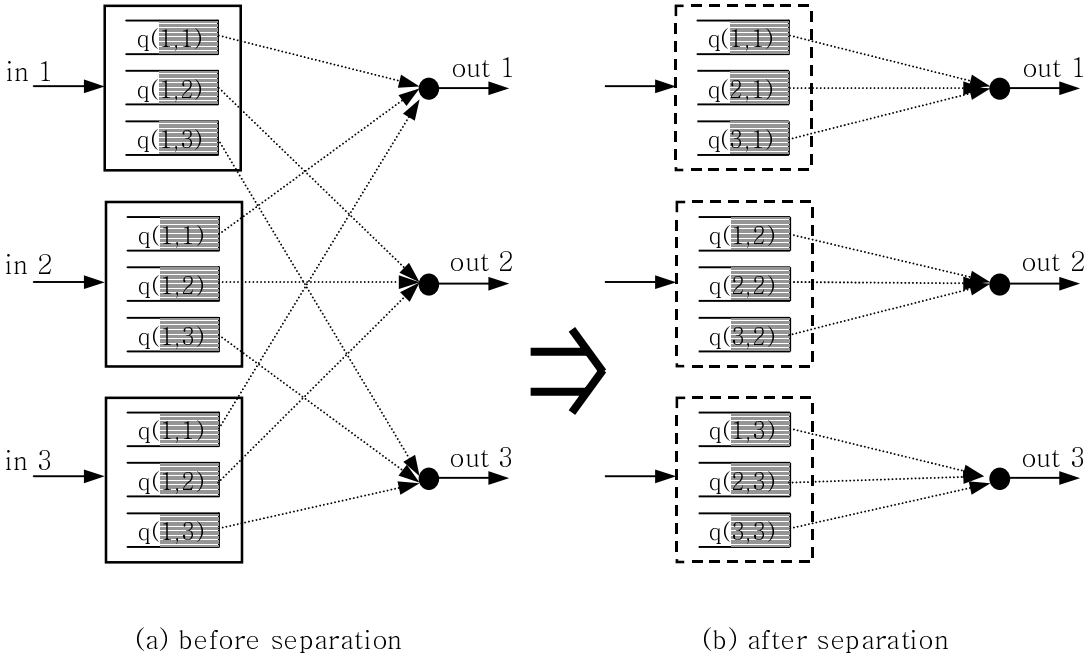

Fig. 3 3×3 VOQ 스위치의 분리.

식의 스위치 처럼 사용되기 때문에 VOQ(virtual output-queued) 스위치라고 불리운다.

다음 장에서 살펴보겠지만, MIQ 스위치는 각 입력포트 내의 FIFO 메모리의 개수가 증가할수록 높은 수율을 내는 장점을 가지고 있지만, 메모리를 여러 개 관리해야 하는 문제가 발생한다. 즉, SIQ 스위치에서는 N 개의 큐에 대해 동시에 스케쥴링을 하게 되지만, Odd-Even 스위치에서는 두 개의 부류에 대해서 스케쥴링을 해야만 한다. 문제는, 특정 타임슬롯 동안 한 입력포트에서는 하나의 셀만을 서비스하여야 하므로 서로 다른 큐들의 스케쥴링을 조화시키는 문제가 발생하게 되는 것이다. 이러한 스케쥴링의 문제는 4장에서 자세히 다룰 것이다.

3 MIQ 스위치의 성능 분석

이 장에서는 참고문헌 [4]에서 분석된 uniform 트래픽에 대한 MIQ 스위치의 성능분석 결과를 제시한다. 앞 장에서도 지적된 것처럼, MIQ 스위치는 하나의 입력포트가 m 개의 메모리를 가지고 있으므로, 스케쥴링 방식에 따라 특정 입력포트는 최대 m 개의 셀을 서비스 할 수도 있다. 따라서, 이 장에서는 메모리들의 상호관계를 고려한 경우와 고려하지 않은 경우 두 가지로 나누어 성능 분석 결과를 제시한다.

3.1 메모리의 상호관계를 고려하지 않은 경우

입력포트 내부의 메모리 사이의 상관관계를 고려하지 않는 경우의 해석은 무척 간단해 진다. 그 이유는 모든 메모리가 서로 동일하고 독립적이라고 가정할 수 있기 때문이다. 따라서, 이 경우에는 한 입력포트에서 최대 m개의 셀을 서비스 할 수 있게 된다. 즉, 각 메모리가 다른 메모리의 서비스 여부와 상관없이 셀을 서비스 하게 되는 것이다.

이 경우에는 각 메모리가 특정 출력포트들로 향하는 셀만을 저장하므로 전체 스위치를 작은 부스위치(sub-switch)로 나눌 수 있다. VOQ 스위치를 예를 들어보자. VOQ 스위치는 입력포트 내의 각 메모리가 특정한 하나의 출력포트로 향하는 셀들만을 저장하게 되고 그 출력포트에 대한 스케쥴링에만 참여를 할 수 있다. 따라서,전체 스위치는 $m = N$ 개의 부스위치들로 나뉘어질 수 있다.

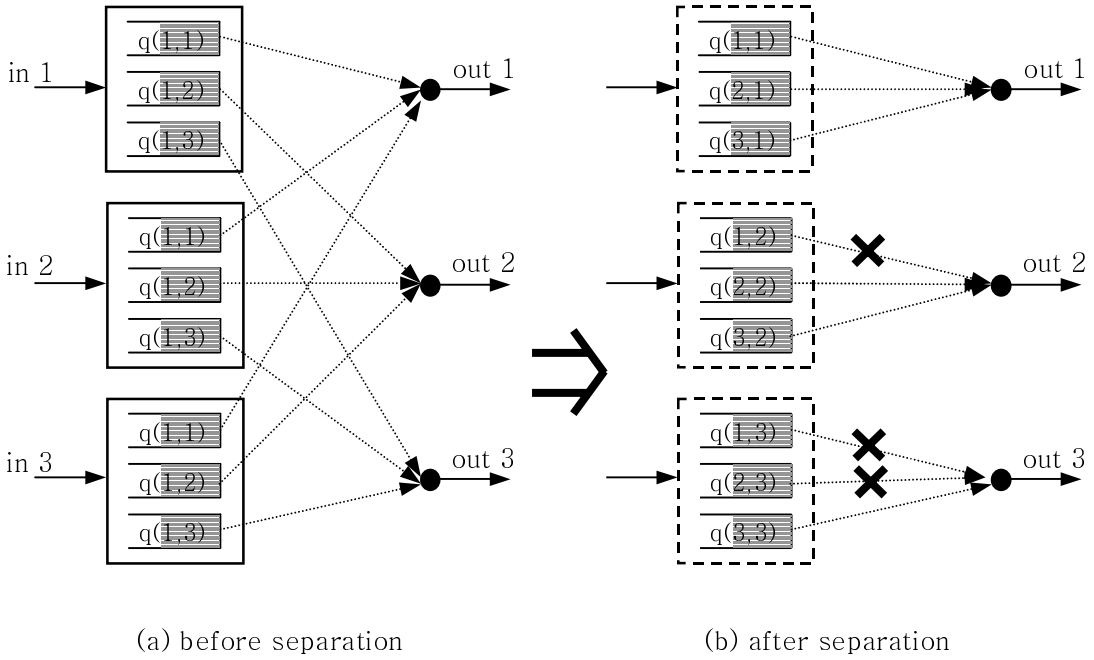

Fig. 4 메모리들 사이의 상관관계를 고려했을 때의 MIQ 스위치의 분리.

메모리의 상관관계를 고려하지 않은 경우, 모든 입력포트 및 모든 메모리들이 독립적이고 uniform한 트래픽이 부과된다는 가정하에서는 전체 스위치를 해석하는 대신 부스위치를 해석해도 동일한 결과를 얻을 수 있으며 해석도 간단해진다. 그림 3은 3×3 VOQ 스위치를 출력포트 별로 분리하는 과정을 보여주고 있다. 그림 3.(b)에서처럼, 분리된 후의 모습이 마치 출력버퍼 방식의 스위치의 모습과 비슷한 형태가 되기 때문에 가상의 (virtual) 출력버퍼형 스위치 라는 이름을 얻게 된 것이다.

그림에서 $q(i, j)$는 입력 i에 있는 j번째 큐를 의미한다. 메모리의 상관관계를 고려하지 않는 경우는, $q(1, 1)$, $q(1, 2)$, $q(1, 3)$ 처럼 하나의 입력포트에서 여러 개의 셀이 선택될 수가 있다. 이는 입·출력 사이의 매칭의 수를 늘림으로써 성능의 향상을 가져올 수는 있으나 내부 동작 속도의 증가 라는 문제를 가져오게 되며, 결국 출력버퍼 방식의 스위치와 동일하게 된다. 물론, 입력포트 내의 메모리들이 모두 물리적으로 구분되는 경우는 내부 동작 속도의 증가 문제는 없어지지만, 이런 경우에는 각 메모리들을 출력 포트와 개별적으로 연결해줄 필요가 발생하며 이는 스위치 패브릭의 사이즈가 커져야 함을 의미한다. 따라서, 다음 절에 소개되는 것처럼 메모리들 사이의 상관관계를 고려해 줄 필요가 있게 된다.

이 경우의 스위치의 성능은 많은 논문에서 다양한 방법으로 해석되고 있으며, 공통적으로 얻어지는 saturation throughput T_f는

$$T_f = m + 1 - \sqrt{m^2 + 1} \tag{1}$$

과 같다. 그림 5는 무한 크기를 갖는 스위치의 saturation throughput을 메모리의 개수(m: bifurcation parameter)에 대해 나타내고 있다. 실선으로 나타내어진 것이 T_f에 해당한다. m의 값이 점점 커질수록 saturation throughput은 1로 수렴하는 것을 알 수 있으며, $m = 4$ 일 때 90% 정도의 throughput을 얻을 수 있음을 보여주고 있다.

3.2 메모리의 상호관계를 고려한 경우

입력포트 내부의 버퍼들 사이의 상관관계를 고려하게 되면, 스위치를 분석하는 것은 더욱 복잡해진다. 즉, 이 경우에는 한 입력포트가 한 개의 셀만을 서비스 할 수 있게 되므로, 셀을 서비스 하기로 되어 있는 입력포트는 다른 출력포트들에 대한 스케쥴링에 참여하면 안 된다. 실제로 대부분의 스케쥴링은 동시에 이루어지고 있지만, 해석을 하는 과정에서는 스케쥴링이 순차적으로 일어난다고 가정함으로써 이러한 문제를 해결할 수 있다.

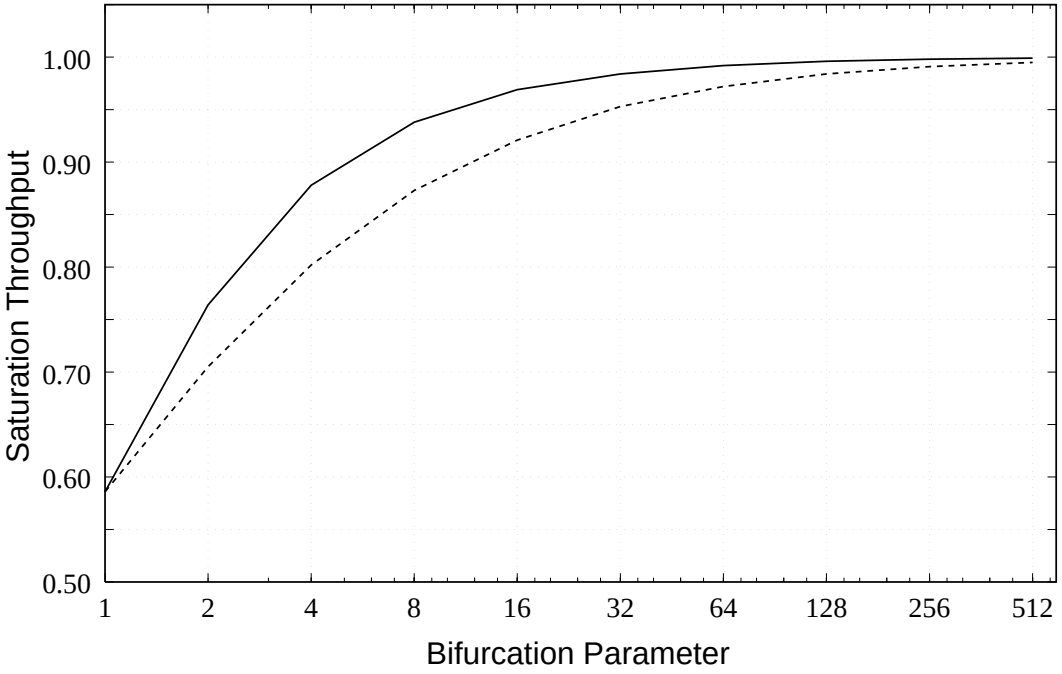

Fig. 5 MIQ 스위치의 saturation throughput.

즉, 그림 4에 보이는 것처럼 첫번째 출력포트에 대한 스케쥴링에는 모든 큐들이 참여를 할 수 있지만, 두번째 출력포트에 대한 스케쥴링에는 첫번째 스케쥴링에서 선택된 입력포트에서 오는 request는 배제를 시키는 것과 같은 방법으로 해석을 할 수 있다. 그림 4에서는 $q(1,1)$이 처음에 선택이 되었다라고 가정을 하였으며, 그런 이유로 1번 입력포트로부터의 request가 두번째 및 세번째 스케쥴링에는 참여할 수 없음을 나타내고 있다.

이 경우의 saturation throughput T_r은

$$T_r = \frac{1}{m}\sum_{i=1}^{m} T_i = \frac{1}{m}\delta_m \tag{2}$$

과 같이 복잡하게 나타내지며, 이때 T_i와 δ_i는 다음과 같은 관계를 가지고 있다.

$$T_i = m - \delta_{i-1} + 1 - \sqrt{(m-\delta_{i-1})^2 + 1}\,, \tag{3}$$

$$\delta_i = \begin{cases} 0, & i = 0 \\ \sum_{j=1}^{i} T_j\,, & i \geq 1. \end{cases} \tag{4}$$

식 (2)에 대한 결과 그래프는 그림 5에 m의 함수로서 점선으로 표시되고 있다. 메모리들의 상관관계를 고려하지 않은 경우처럼 m의 값이 증가함에 따라 saturation throughput은 1로 수렴하고 있지만, $m = 1$인 경우를 제외하고는 모든 경우에 있어 메모리들의 상관관계를 고려하지 않은 경우보다 조금 작은 값을 나타내고 있다. T_f에 비해 throughput이 적은 이유는 특정 입력포트가 오직 한 개의 셀만을 서비스 할 수 있기 때문이다.

메모리들의 상관관계를 고려한 경우의 saturation throughpug을 나타내는 식(2) 및 식 (3)에 사용된 δ의 의미에 주목할 필요가 있다. 식(1)과 식(3)을 비교해 보면 알겠지만, δ_{i-1}은 $i-1$ 번째 까지의 스케쥴링 과정에서 선택된 부분을 제거해 주는 역할을 하고 있다. 만약, 메모리들의 상관관계를 고려하지 않게 된다면 δ_i는 0이 되어 결국 식(3)은 식(1)와 같은 형태가 된다.

4 MIQ 스위치를 위한 스케쥴링 알고리즘

2장에서 지적된 것처럼, MIQ 스위치의 각 입력포트는 여러 개의 메모리를 가지고 있다. 하지만, 복수 개의 셀을 서비스 하게 되는 경우는 내부 동작 속도를 증가시켜야 하고 출력포트에도 메모리를 요구하게 되어 진정한

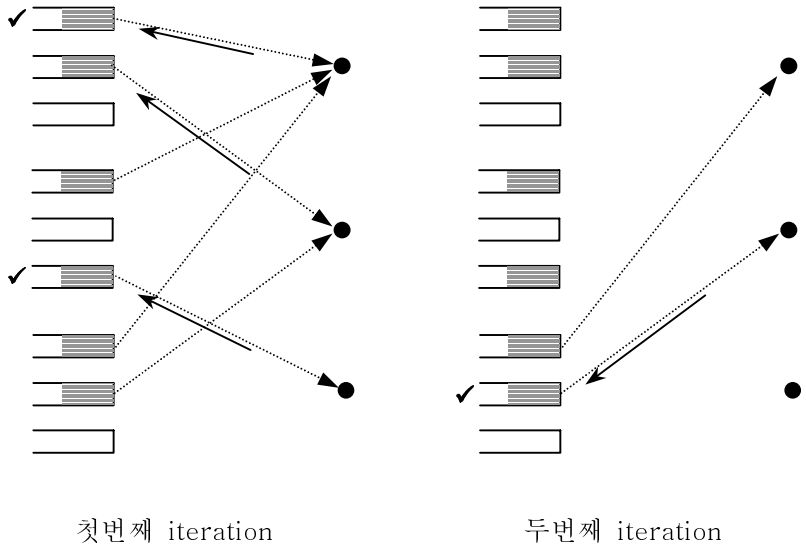

Fig. 6 PIM 동작의 예: 3×3 VOQ 스위치.

의미에서의 입력버퍼 방식의 스위치가 갖는 장점들을 잃어버리게 된다. 따라서, 각 입력포트가 특정 타임슬롯 동안에 최대 한 개의 셀만을 서비스 하도록 적절히 제어를 해 줄 필요가 있으며, 그러한 역할을 하는것이 스케쥴러이다. 참고 문헌 [5]과 [6]에 제시된 스케쥴링 알고리즘은 버퍼의 개수에 상관없이 모든 유형의 MIQ 스위치에 대해 적용 가능한 알고리즘이지만, 대부분의 MIQ 스위치를 위해 개발된 알고리즘이 VOQ 스위치 용으로 개발되었으므로 이장에서는 MIQ 스위치의 특수한 경우인 VOQ 스위치에 대한 스케쥴링 알고리즘들에 대해 살펴보기로 한다.

여러 가지 방식의 VOQ 스위치를 위한 스케쥴링 알고리즘이 이미 개발되어 발표되었고, 일부는 실제 상용 스위치나 라우터에 사용되기도 하였다 [7]. 그러한 알고리즘들의 상당수는 Anderson에 의해 제안된 PIM (parallel iterative matching) 알고리즘의 동작 원리에 기반을 두고 있다. 따라서, 4.1 절에서 PIM 알고리즘의 동작 원리에 대해 살펴본 후, 4.2 절에서 PIM에 기반을 둔 알고리즘들의 동작 원리를 알아보도록 한다. 그리고, SLIP과 같은 대표적인 알고리즘을 간략히 서술하였다.

4.1 Parallel Iterative Matching 알고리즘

Anderson에 의해 제안된 PIM 알고리즘은 입력단과 출력단 사이에서 Request, Grant, 그리고 Accept 라는 세 단계를 반복적으로 행함으로써 입력과 출력 사이의 최대 매칭(maximal matching)을 찾아내는 방법이다[3] [8]. PIM의 동작원리는 다음과 같다.

Request phase 매치가 이루어지지 않은 입력포트들의 리퀘스트 정보를 해당 출력포트로 보낸다.

Grant phase 매치가 이루어지지 않은 출력포트들이 Request 단계에서 받은 request 중에서 랜덤하게 하나를 선택한 후 해당 입력포트에 grant를 보낸다.

Accept phase 매치가 이루어지지 않은 입력이 랜덤하게 하나의 grant를 선택한다.

이와 같은 세 단계의 동작을 필요한 만큼 반복하는 것이 PIM의 기본 동작 원리이다.

그림 6는 3×3 VOQ 스위치에서 PIM이 동작하는 원리를 보여주는 하나의 예이다. 그림에서 보이는 것처럼, 점선으로 표시된 Request 단계에서는 입력포트들이 가지고 있는 큐들의 HOL 위치에 있는 셀들의 목적지로

[3] 우리 말로 최대 매칭은 maximal matching과 maximum matching 두 가지를 모두 나타내기 때문에 혼란을 가져올 수도 있다. 하지만, 이 두 가지 개념의 차이를 분명히 할 필요가 있는데, 간단히 말하면 maximum이 더 커다란 개념으로 생각하면 된다. 즉, maximum은 maximal이 될 수 있지만, maximal은 maximum이 되지 못한다.

request 신호를 보낸다. request 신호를 받은 각 출력포트는 랜덤하게 하나의 request를 선택해서 해당 입력에 grant 신호를 보내며, 이러한 Grant 단계는 실선으로 표시되어 있다. 1번 및 2번 출력포트는 request들 중에서 모두 1번 입력에서 온 request를 선택했고, 3번 출력포트는 오직 하나의 request 만을 받았으므로 2번 입력에서 온 request를 선택했다. 1번 입력포트는 Grant 단계에서 두 개의 grant를 받았으므로 그 중 하나의 grant를 Accept 단계에서 랜덤하게 선택한다. 그림에서는 1번 출력에서 온 grant를 선택했음을 나타내고 있다. 2번 입력은 3번 출력포트에서 온 하나의 grant 만을 받았으므로 그 grant를 선택했고, 3번 입력은 선택할 grant를 받지 못했다. 두번째 iteration에서는 이전 iteration에서 선택되지 않은 입력포트들만이 request 신호를 보내므로 3번 입력만이 request 신호를 보내고 있으며, 이전 iteration에서 선택되지 않은 출력포트들만이 request를 선택해서 grant 신호를 보낼 수 있으므로 2번 출력만이 grant를 보내고 있고 그 grant가 3번 입력에서 선택되는 과정을 보여주고 있다. 이와 같이 세 개의 단계를 몇 차례 반복하므로써 매칭의 수를 증가시키고 스위치의 성능을 향상시키게 되는 것이다.

PIM의 동작에 있어서의 두드러진 특징은 iteration, randomness, 그리고 parallelism으로 설명될 수 있다. 즉, Request, Grant, 그리고 Accept의 기본 동작들을 몇 차례 반복하며, Grant 및 Accept 단계에서 request나 grant를 랜덤하게 선택한다. 또한, 이 모든 동작은 다른 입·출력포트에 상관없이 패러럴하게 이루어져서 제어에 필요한 시간을 최소화 시킬 수 있다. 이 절의 서두에서 이야기된 maximul matching을 찾기 위해서는 $\log N$만큼 iteration 시켜야 하지만 (N은 스위치의 크기), 연구 결과에 의하면 일반적인 사이즈의 스위치에 대해 3회 정도의 iteration만으로도 만족스러운 성능을 나타내고 있는 것으로 알려지고 있다.

PIM 방식의 단점으로는 첫째, 랜덤 선택으로 인한 회로의 복잡성을 들 수 있다. 일반적으로 랜덤 선택은 개념적으로는 매우 간단하지만, 회로를 구현하는데 있어서는 뒤에서 소개될 round-robin 방식 등에 비해 복잡한 것으로 알려져 있다. 또한, 랜덤 선택으로 인해, 최악의 경우에는 특정 입력의 특정 큐가 영원히 서비스 되지 않는 starvation 현상이 발생할 수도 있어 service의 delay requirement를 만족시키지 못할 수도 있다.

4.2 PIM 기반의 알고리즘

PIM에 기반을 둔 스케쥴링 알고리즘들은 입력들과 출력들 사이에서 서비스할 request들을 선택하는 일종의 동작을 반복하는 알고리즘들을 의미한다. 그림 7은 4.1 절에서 설명된 PIM의 기본적인 동작 원리를 나타내는 프로토콜을 보여주고 있다. 프로토콜의 일반적인 동작은 다음과 같다:

1. 입력들은 출력들에게 requests를 방송(broadcast)한다.
2. 각 출력은 하나의 request를 선택해서 해당 입력포트에게 grant를 보내어 선택여부를 알린다.
3. 각 입력은 grant 중 하나를 선택한다.

이러한 일련의 동작이 몇 번 반복된 후에 스케쥴링의 결과에 따라 최종적으로 데이터 패킷이 전송되게 된다.

PIM의 기본 동작에 기반한 스케쥴링 알고리즘들 사이의 기능적인 차이는 두번째와 세번째에서 각각 request와 grant를 선택하는 방식에 따라 결정된다. 4.1 절에서 설명된 것과의 차이를 비교하면, 두번째 및 세번째에서 랜덤하게 하나의 request와 grant를 선택한다는 말이 빠져 있을 뿐이다. 이러한 기능적인 차이는 중대한 성능의 차이를 유발하게 된다. 4.3 절부터는 PIM의 기본적인 동작에 기반을 둔 스케쥴링 알고리즘들을 간단히 기술할 것이다.

4.3 RRM: Round-Robin Matching

RRM[9]은 round-robin 동작에 기반하여 Grant 및 Accept 신호를 발생함으로써 PIM의 random selection에 의한 복잡성 및 unfairness의 문제를 해결하는 것을 목적으로 개발되었다. 기본 동작을 살펴보면, Grant 단계에서 각 출력은 임의의 고정된 round-robin 순서의 다음에 오는 request에 grant를 수여하며, 이와 동일한 방식으로 Accept 단계에서 각 입력은 어떤 grant를 받아들일지를 선택하게 된다. 문제는 입력 혹은 출력포트들에 있는 round-robin 포인터들 사이의 동기화(synchronization)로 인해 PIM 보다 나쁜 성능을 낸다는 것이며, 특정 트래픽 환경에서는 상당히 큰 지연(delay)을 유발하기도 한다. 좀더 구체적으로 설명하면, 출력포트들에 있는 포인터들이 동기화되어 움직이기 때문에, 특정 순간에 특정 입력에서 온 request들이 동시에 선택되어지게 된다. 이렇게 됨으로써 전체적인 성능이 저하된다.

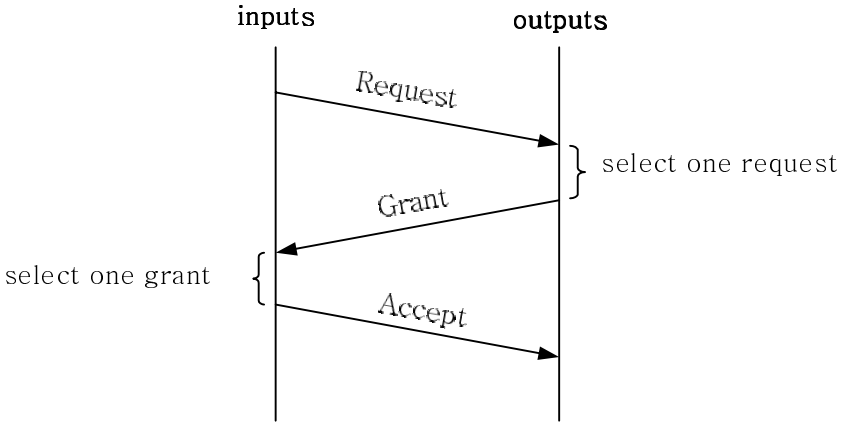

Fig. 7 PIM 동작 원리를 나타낸 프로토콜.

4.4 *i*SLIP

*i*SLIP[10]은 앞 절에서 지적된 RRM의 한계(즉, round-robin 포인터들 사이의 동기화로 인한 성능의 제한)를 해결하기 위해 개발되었다. 즉, RRM에서는 입력과 출력에 있는 round-robin 포인터들이 동기가 되어 움직였지만, *i*SLIP에서는 선택된 grant에 해당하는 포인터들만을 하나씩 증가시킴으로써 포인터들 사이의 동기화 문제를 해결하였다. 예를 들어, 2번 입력과 3번 출력만 선택되었다면 2번 입력과 3번 출력포트에 있는 포인터만 1씩 증가하게 되며, 나머지 입·출력포트에 있는 포인터는 변하지 않는다. 비록 RRM에 사소한 변화를 준 것이지만, *i*SLIP은 단 한번의 회전(iteration)만으로도 100%의 포화 수율(saturation throughput)을 얻을 수가 있다. *i*SLIP은 또한 service guarantee, 즉 fairness도 제공한다. 이는 임의의 request가 특정 시간 (bounded time) 안에 서비스됨을 의미한다.

4.5 그 외의 스케쥴링 알고리즘

PIM의 기본 동작에 바탕을 두지 않고 VOQ 스위치를 위해 개발된 스케쥴링 알고리즘도 다수 있다. 예를 들면, LaMaire의 2DRR (2-Dimensional Round-Robin) 방법[11]과 Kim의 Chessboard 알고리즘 [5] 등이 있다. 이러한 방법들과 PIM 기반 스케쥴링 알고리즘들과의 가장 두드러진 차이는 동작이 순차적(sequential)으로 이루어진다는 것이다. 어떤 출력포트가 하나의 request를 선택하는 과정을 보통 arbitration round 라고 말하는데, 이러한 round가 순차적으로 구성되어 있다는 것이다. 주목할만한 점은, 먼저 수행된 arbitration round의 결과가 나중에 수행되어질 round에 반영된다는 것이다. 따라서 한 번의 iteration 만으로 모든 일이 끝나지만, 한 번의 iteration이 parallel 알고리즘을 N 번 수행한는 것과 같으므로 콘트롤 시간이 비교적 길다는 단점이 있으며, 따라서 스위치의 크기가 큰 경우에는 사용할 수 없게 된다. 하지만, 대부분의 순차적인 알고리즘들은 maximal matching 보다는 maximum matching을 찾기 때문에 항상 최적의 성능을 내게 된다.

5 VOQ 스위치가 가장 좋은가?

VOQ 스위치에 대한 많은 스케쥴링 알고리즘들이 내부 속도 증가 없이 간단한 방식에 의해 출력버퍼 방식의 스위치에서와 같은 성능을 제시하자 그런 부류의 스케쥴링 알고리즘이 최적의 방식인 것으로 이해되고 있다. 이런 이유는 그동안 VOQ 스위치용 스케쥴링 알고리즘들에 대한 단점이 전혀 지적되지 않았기 때문이다. 따라서, 이장에서는 VOQ 스위치용 알고리즘들의 일반적인 단점 혹은 문제점들을 간단히 지적하고자 한다.

VOQ 스위치용 알고리즘들의 가장 대표적인 문제는 버퍼의 개수로부터 나온다. 첫번째 문제는, VOQ 스위치가 N^2의 메모리를 필요로 한다는 것이다. 비록, 하나의 커다란 메모리를 여러 개의 논리적인 메모리로 나누어 사용할 수도 있지만, 스위치의 크기가 커질 때는 이러한 방법에도 문제가 발생한다. 더욱이, 메모리의 개수

가 4개 이상일 때는 메모리의 개수가 그 이상으로 증가해도 아주 적은 성능 향상만이 있다는 연구결과는 VOQ 스위치가 최선의 방법만은 아님을 의미한다 하겠다 [12, 13, 14]. 두번째는, 메모리의 개수에 따라 제어시간도 증가한다는 것이다. 특히, 대규모의 스위치의 경우 N^2의 메모리에 대해 아주 짧은 시간에 복잡한 스케쥴링 알고리즘을 수행하는 것이 불가능할 수도 있다.

6 결론 및 맺음말

인터넷의 급속한 보급 및 관련 응용 프로그램들의 개발로 인해 다양한 형태의 멀티미디어 서비스가 보편화되기 시작하면서 네트웍을 지나가는 데이터 트래픽의 양은 기하급수적으로 증가하고 있다. 이러한 트래픽 양의 증가는 네트웍이 보다 더 빨라지기를 요구하고 있으며, 그러한 요구에 부응하기 위한 많은 연구가 진행중이다. 이미 전송 분야에 있어서는 광케이블 및 WDM 전송 방식과 같은 전송기술들이 개발되어 수백 기가 혹은 수 테라급의 대역폭을 제공하고 있다. 교환 영역에서도 수십 기가에서 수 테라급에 이르는 고속 교환기 및 라우터들이 이미 개발되었거나 개발중에 있다.

본고에서는 고속 스위치 및 라우터에 사용되는 효율적인 버퍼링 방식인 다중입력버퍼링(MIQ) 방식에 대해서 살펴보았다. MIQ 방식은 각 입력포트에 여러 개의 FIFO 메모리를 위치시킴으로써 HOL 블록킹을 줄이거나 완전히 없앰으로써 입력버퍼 방식의 스위치/라우터의 성능을 향상시키는 방식이다. 각 입력포트의 FIFO 메모리의 개수가 스위치의 크기와 같을 때 MIQ 스위치는 VOQ 스위치라 불리며, 내부 동작속도의 증가 없이 출력버퍼 방식의 스위치와 같은 성능을 제공하는 특징을 가지고 있다.

MIQ 스위치가 입력포트마다 여러 개의 메모리를 가지고 있기 때문에, 효율적인 스위칭을 위해서는 메모리들 사이의 조화로운 제어가 반드시 필요하다. 이를 위해 많은 스케쥴링 알고리즘들이 개발되었으나 대부분이 VOQ 스위치에 제한된 것들이다. 가장 대표적인 스케쥴링 알고리즘은 PIM으로 기본 동작은 Request, Grant, 그리고 Accept의 세 단계로 이루어져 있으며, iteration, randomness, 그리고 parallelism의 특성을 가지고 있다. 나머지 스케쥴링 알고리즘들의 대부분도 PIM에 기반한 것들이며, 주로 Request 및 Accept 단계에서 request나 grant를 선택하는 방식의 차이에 의해 구분이 된다.

References

[1] M.J. Karol, M.G. Hluchyj, and S.P. Morgan, "Input versus output queueing in a space division switch," *IEEE Trans. Commun.*, vol.COM-35, pp. 1347-1356, Dec. 1987

[2] C. Kolias and L. Kleinrock, "The Odd-Even input-queueing ATM switch: performance evaluation," *Proc. IEEE ICC '96*, vol.3, June 1996, pp. 1674-1679.

[3] C. Kolias and L. Kleinrock, "The Odd-Even ATM switch," *IEICE Tr. Communications*, vol.E-81-B, no.2, Feb. 1998, pp. 244-250.

[4] Hakyong Kim, K.S. Kim, and Y.T. Lee, "Performance analysis of the multiple input-queued ATM switch using the restricted rule," submitted to *IEEE/ACM Tr. on Networking.*

[5] Hakyong Kim, K. Kim, Y. Lee, H. Yoon, and C. Oh, "A simple and efficient cell selection algorithm for the multiple input-queued ATM switch," *Proc. IEEE ATM Workshop'99*, Kochi, Japan, 24-27 May 1999, pp.259-264.

[6] Hakyong Kim, K.S. Kim, H.H. Yoon, and Y.T. Lee, "A Throughput-Enhanced Parallel Scheduling Algorithm for the MIQ Switch with a Moderate Number of Queues," *IEE Electronics Letters*, 36(**5**), 2nd March 2000, pp. 477-478.

[7] "Cisco 12000 Series - Gigabit Switch Routers," http:// www.cisco.com/univercd/cc/td/doc/pcat/12000.pdf

[8] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Tr. on Computer Systems*, Nov. 1993, pp. 319-352.

[9] N. McKeown, "Scheduling cells in an input-queued switches," *Ph.D Thesis*, University of California at Berkeley, May 1995.

[10] N. McKeown, "White Paper: A fast switch backplane for a gigabit switched router," *Business Communications Review*, 27(**12**), December 1997.

[11] R.O. LaMaire and D.N. Serpanos, "Two-dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Tr. Networking*, vol.2, no.5, Oct. 1994, pp. 471-482.

[12] G. Thomas, "Bifurcated Queueing for Throughput Enhancement in Input-Queued Switches," *IEEE Commun., Letters*, vol.1, no.2, March 1997, pp. 56-57.

[13] G. Thomas and V. Veludandi, "ATM switches with bifurcated input queueing," *Proc. ICCCN '97*, Las Vegas, NV, USA, 22-25 Sept. 1997, pp. 504-507.

[14] Hakyong Kim, C. Oh, Y. Lee, and K. Kim, "Throughput analysis of the bifurcated input-queued ATM switch," *IEICE Tr. Communi*, vol.E82-B, no.5, May 1999, pp.768-772.

김학용(金學龍) 1995년 충남대 전자공학과 졸업(공학사). 1997년 광주과학기술원(K-JIST) 정보통신공학과 졸업(이학석사)했으며 현재 광주과학기술원 정보통신공학과에서 박사과정에 있다. 2000년 일본 우정성 산하 통신종합연구소(CRL)의 방문연구원으로 있었다. 1999년 호남대학교 전자공학과 강사를 했으며, 2000년 국립목포대학교 전자공학과 강사를 하고 있다. 1995년부터 IEEE 학생회원이었으며 현재는 IEEE Kwang-Ju Section의 student representative로 활동하고 있다. 1999년부터 IEE 학생회원이었으며 현재는 IEE Korea Center를 준비중에 있다. 1999년부터 JCN의 학생회원이며 IEICE의 정규회원이며, 2000년부터 네트워크매니아즈의 웹서퍼로 활동하고 있다. 1995년과 1994년에는 한국통신에서 주관하는 정보통신 논문공모에서 각각 대상과 장려상을 수상하였다. 주관심분야는 다중 입력 버퍼 방식(Multiple Iinput Qqueueing)의 고속 스위치 및 그의 성능분석 및 스케쥴링 알고리즘이며, 최근에는 고속 라우터 및 IPoW, MPoW 등에도 관심을 가지고 있다.