1999 年度 大韓電子工學會 夏季綜合學術大會 論文集 第 22 卷 第 1 號 99/6

입력버퍼 윈도우 기법을 사용한 스위치 액세스 방식

문규춘*, 이우승*, 김 훈*, 박광채**
* 조선대학교 대학원 전자공학과
** 조선대학교 전자정보통신공학부
광주광역시. 동구 서석동 375번지

Switchs Access Rule Using Windowed Input Buffers

Kyu Choon Moon*, Woo Seung Lee*, Hoon Kim*, Kwang Chae Park**
* Dept. of Electronics Eng., Graduate School of Chosun Univ.
** School of Electronics, Information and Communications Eng. Chosun Univ.
375 Seoseck-dong, Dong-gu, Kwangju 501-759, Korea
E-mail : kh@candclab.chosun.ac.kr

Abstract

Hluchyj and Karlo proposed to simple alogorithm for input buffer queue service in packet switch. This paper using not multiple queue but single queue shows, in first, the improved processing rate. Next shows a few fairness as accessing, Finally shows more improved than the existed method.

Ⅰ. 서 론

급변하는 이용자들의 다양한 요구에 신속하게 대처하기 위하여 통신망은 융통성과 모듈성을 추구하면서 기존의 단순한 메시지 전송에서 음성, 데이터, 그래픽, 영상 등 멀티미디어 데이터를 하나의 디지털망에 통합하여 전송하는 광대역 종합정보통신망(B-ISDN : Broadband Integrated Services Digital Network)으로 구현되고 있다. 이러한 광대역 종합통신망에서는 미래의 고속 패킷 스위칭 시스템을 구현하기 위하여 ATM (Asynchronous Transfer Mode) 스위치를 채택하고 있다. ATM 스위치는 입력포트에 입력되는 트래픽이 일정한 레벨이 될 때까지는 셀 손실이 거의 발생하지 않는 가상적 넌 블럭킹(virtual non blocking) 스위치의 구조를 갖으며, 이러한 셀 손실은 셀 스위칭 방식과 버퍼링 방식에 따라서 효과적으로 제어될 수 있다. 입력 버퍼링은 패킷 스위치 설계에서 기본적인 선택사항 중에 하나이다. 스위치로 향하는 각각의 N개 입력 라인은 각 라인에서 도착한 패킷을 저장하기 위한 분리된 버퍼를 가지고 있고, 패킷들은 스위치 구조 안으로 바로 액세스하여 도착 할 수 없다. 왜냐하면 서로 다른 입력으로부터 동일한 출력으로 서비스를 받기 위한 셀들로 인해 발생되는 HOL(Header of Line) 블럭킹으로 인하여 최대 처리율이 58.6% 정도로 저조하기 때문이다. 이를 극복하기 위해 복잡한 제어 알고리즘들과 넌 FIFO(First In First Out) 메모리 구조들이 제안되어 향상된 처리율을 나타내었지만 각 포트를 스케닝하고, 셀 스케줄링을 하기 위해서 제어부의 속도가 N배의 속도로 동작하여야 하는 문제점을 가지게 되었다. 이와 같이 수십 Giga bit/s 또는 수십 Tera bit/s의 초고속 스위치를 구현하기 위한 다수의 우선순위와 복잡한 셀 스케줄링을 처리해야 하므로 제어계에 어려움이 많다. 본 논문에서는 초고속 ATM 스위치를 구현하기 위해 입력 버퍼형 스위치에 대한 Hluchyj-Karol 윈도우 기법보다 개선된 처리율과 공정한 특징을 볼 수가 있었다.

Ⅱ. 기존의 윈도우 기법

N×N 입력 버퍼형 넌 블록킹 타임슬롯 패킷 스위치 구조가 그림 1에 나타나 있다.[1~3]
스위치로부터의 패킷의 도착과 출발들은 서로 다른 시간에 일어나게 된다. 각 포트에서 입력 버퍼는 입력포트로 향하는 패킷들의 도착을 기다린다. 각 타임슬롯에서는 많아야 하나의 패킷들이 도착하게 된다. 본 논문은 단일화된 트래픽을 가정하였고, 그래서 도착하는 패킷은 동등한 확률을 가지고 있는 N개의 출력포트들 중 임의의 목적지 주소로 전송할 수 있다. 여기에는 출력포트를 위한 버퍼는 있지 않으므로 각 입력포트에서 각각의 타임슬롯은 어떠한 하나의 입력포트로부터 많아야 하나의 패킷을 받을 수 있다.

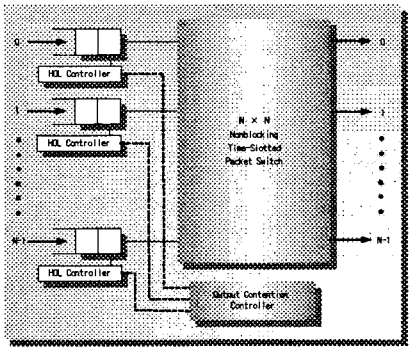

그림 1. N×N 입력버퍼형 스위치

각 타임슬롯에서 각 입력 큐의 HOL 패킷은 먼저 이 패킷의 출력 목적지 주소를 조사하고, 계속해서 포트 1로부터 포트 N까지 이 과정을 반복하게 된다. 패킷은 만약 요청된 출력포트가 아직 다른 입력포트의 HOL 패킷에 할당되어 있지 않는다면 스위치 구조로 액세스 되는 것을 허락받게 된다. 만약 어떠한 입력 큐에서 HOL 패킷이 현재의 슬롯에서 액세스가 거부되면 FIFO(First In First Out) 규칙에 의하여 이 큐 안에서의 모든 패킷들은 거부된 패킷 뒤에 있게 된다. 포트 1부터 포트 N까지의 조사된 순서에 의해 입력포트 중에서는 우선적으로 조사받은 것부터 자연스럽게 감소하게 된다. Hluchyj와 Karol는 분석과 시뮬레이션에 의하여 FIFO 규칙 하에서 커다란 N에 대하여 도달할 수 있는 가장 근사적인 처리율이 약 0.586이라는 것을 보였다. 그러므로 패킷 스위치 셀계에 대한 입력 버퍼링은 제한적인 선택된 처리율을 가지고 있다.

Hluchyj와 Karol는 처리율의 한계를 넘어 우회할 수 있는 논제에 관하여 발표하였고, FIFO 규칙을 극복해야 하는 다음과 같은 윈도우 액세스 프로토콜을 제안하였다.[4] 하나의 타임슬롯에서 각각의 N 입력 큐들 중에서 w를 맨 처음의 패킷이라 가정한다. N 입력 큐들에서 처음 위치한 패킷들은 위에서 설명한 것과 같이 출력포트들에 액세스되기 위하여 경쟁을 하게 된다. 그 다음 큐들 중에서 스위치의 액세스를 위해 선택되지 못한 HOL 패킷들은 두 번째 위치의 패킷들에 대해 액세스되기 위하여 다시 경쟁하게 된다. 그 다음에는 첫 번째나 두 번째 위치에서 액세스를 승낙받지 못한 패킷들이 세 번째 위치의 패킷들과 경쟁하게 된다. 이러한 과정은 어느 쪽이 빨리 완료되던 간에 경쟁과정으로 w 주기가 끝나거나 모든 출력포트들에서 선택한 패킷들을 가질 때까지 계속된다. 비록 윈도우 크기를 적절히 조절해야 하나 처리율 면에서 매우 개선효과를 얻을 수 있다.

Ⅲ. 개선된 스위치 액세스 방식

H-K 윈도우 기법에서는 각 타임슬롯마다 경쟁을 하였고, 전혀 제공되지 않는 입력 큐의 수를 최소화하기 위한 것을 나타내고 있다. 이것은 경쟁 과정에서 더 이상 제공되지 않는 입력 큐들에 접속되는 기회를 제한하는 것이다. 이 방법은 모든 큐들에 대하여 공정하게 시도할 수 있는 장점을 가지고 있으나 가끔씩 가득 찬 윈도우의 종단 부분에서 출력포트를 할당받을 수 없고, 윈도우의 주소 내에 있는 패킷들도 포트를 할당받을 수 없게 된다. 하지만 제안한 방식은 두 개의 명백한 결함을 막을 수 있다.

큐의 HOL 패킷에서 출력포트 j로 주소를 할당받은 패킷은 경쟁과정의 첫 번째 주기에서 선택되어진다. 그리고 나서 현재의 타임슬롯에서 출력포트 j를 다시 지정하기 위하여 출력포트 j에 할당되었던 패킷들은 제거되어 남아있지 않게 된다. 즉, 출력포트 j는 실제로 현재의 타임슬롯에서는 비록 패킷들이 이 주소를 할당받았어도 계속해서 비어있게 된다. 이와 다른 유사한 경우에도 스위치의 효율은 실제로 행하여 진 것보다 낮아지게 된다.

개선된 윈도우 방식은 H-K 윈도우 방식에 대하여 다른 방안으로 제안한 것이다. 비록 같은 입력 큐로부터 다중 선택을 허용하더라도 각 타임슬롯에서 할당된 출력포트의 수를 최대화하는 것이다. 개개의 입력 라인에 대하여 어떠한 불공평성은 아마 N개의 입력버퍼를 초과하여 도착하는 패킷들을 임의적으로 간단하게 수정할 수 있다. 그러나 현재의 윈도우 내에서 입력 패킷들이 출력포트를 할당받을 때 출력포트가 비어있는 상태로 되는 것을 허락하지 않으므로서 우리는 개선된 처리율을 얻을 수 있다.

식 1은 N=4와 윈도우 크기 w=3인 경우에 대한 알고리즘의 동작을 나타내었다. 아래와 같이 4×3의 가득 찬 윈도우로 시작한다. 숫자들은 목적지 주소들을 나타내고, 별표는 비어 있는 위치를 나타낸다. 현재의 윈도우 내에서 라인 1에 대한 입력버퍼는 3개의 주소를 가지고 있고, 그들의 도착 순서에 의하여 포트 2, 3 그리고 2로 된다. 즉, 가장 위의 열은 오른쪽에서 왼쪽으로 읽어간다. 스위치는 위에서부터 아래까지 이러한 입력들을 조사하고, 가장 오른쪽 행부터 시작한다.

$$[\text{입력윈도우}] = \begin{bmatrix} 2 & 3 & 2 \\ 4 & 4 & 3 \\ 1 & 2 & 1 \\ 1 & 3 & 3 \end{bmatrix}$$

$$[\text{출력포트}] = [\ * \ * \ * \ * \ ] \qquad \text{(식 1)}$$

가장 오른쪽 행을 조사하고, 마지막에 첫 번째 3개의 HOL 패킷들을 선택하는데 출력포트 2, 3 그리고 1에 대하여 각각의 주소를 할당하게 된다. 라인 4에서의 HOL 패킷은 확실하게 차단된다. 이것은 현재의 단계에서 할당받지 못한 출력포트 4만을 버리게 된다. 결국 스위치 상태의 행렬과 출력포트 벡터는 다음과 같다.

$$[\text{입력윈도우}] = \begin{bmatrix} 2 & 3 & * \\ 4 & 4 & * \\ 1 & 2 & * \\ 1 & 3 & 3 \end{bmatrix}$$

$$[\text{출력포트}] = [\ 1\ 2\ 3\ *\ ] \qquad \text{(식 2)}$$

다음의 두 주기에서는 H-K 알고리즘 HOL 블록킹 상태에서 가장 끝 부분을 찾는데 식 3과 같이 나타낸다.

$$[\text{입력윈도우}] = \begin{bmatrix} 2 & 3 & * \\ 4 & * & * \\ 1 & 2 & * \\ 1 & 3 & 3 \end{bmatrix}$$

[출력포트] = [1 2 3 *] (식 3)

여기에서는 포트 4로 어떠한 패킷이 주소가 지정되었는지 알 수 없다. 그러므로 출력포트 4는 비어있는 상태로 되고, 이 슬롯에서는 단지 75%의 이용률만 얻을 수 있다.

개선된 알고리즘은 2행의 모든 요소들을 이용할 수 있게 스위치를 허용하고, 마찬가지로 포트 4에 할당되어진 하나의 패킷을 발견할 수 있어 결과적으로 현재의 슬롯에서 100%를 이용할 수 있다. 결과적으로 현재 타임슬롯의 마지막에서 윈도우 상태는 위에서와 같이 변화되지 않는 것을 남긴다.

윈도우는 이 시점에서 다음 타임슬롯에 대한 추가적인 입력들이 다시 채워지는 것을 필요로 한다. 각 큐의 오른쪽으로 채워지고, 가득 채워진 윈도우에서 각 큐의 왼쪽 끝 부분은 추가적인 입력들로 덧붙여진다. 이런 경우에 큐 1에서 HOL 위치는 출력포트 3으로 주소가 할당되어진 패킷에 의해서 차지하게 된다. 다음 타임슬롯에서 이 패킷은 큐 4에서 포트 3으로 주소가 지정되어 이미 대기중인 HOL 패킷을 넘어서 선택되어질 것이다. 왜냐하면, 뒤의 패킷이 명백하게 먼저 도착하였고, 이것은 확실히 불공평하기 때문이다.

제안한 방식은 이것에 대해 다음과 같은 방법으로 재배치하므로서 공정한 측정을 할 수 있다. 행 1의 남은 부분에 있는 패킷인 가장오른쪽 행을 가장 윗 부분으로 이동시키고, 그 아래 부분부터 비어 있는 위치인 별표로 된 부분을 채운다. 행 1에서 비어있는 부분은 이제 아래쪽으로 내려가게 된다. 그 다음에는 만약 행 1의 아래 부분에 비어있는 곳에서 오버플로우가 발생하면 행 2 위쪽의 나머지 부분 패킷을 이동시킨다. 행 3 등에 대하여도 마찬가지로 하는데 다음과 같다.

$$[\text{입력윈도우}] = \begin{bmatrix} 2 & 3 & * \\ \uparrow & \uparrow & \uparrow \\ 4 & * & * \\ \uparrow & \uparrow & \uparrow \\ 1 & 2 & * \\ \uparrow & \uparrow & \uparrow \\ 1 & 3 & 3 \end{bmatrix}$$

[출력포트] = [* * * *] (식 4)

마지막으로 결과는 다음과 같다.

$$[\text{입력윈도우}] = \begin{bmatrix} * & 2 & 3 \\ * & 4 & 3 \\ * & 1 & 2 \\ * & 1 & 3 \end{bmatrix}$$

[출력포트] = [* * * *] (식 5)

네 개 이상의 입력패킷들 또는 필요한 만큼의 수가 더해지므로서 다음 타임슬롯은 윈도우를 채우기 시작한다. 이 윈도우 재충전 방법은 위에서 나타낸 것과 같이 액세스할 때 어느 정도 공정성을 보증할 뿐만 아니라 하드웨어와 소프트웨어에서도 아주 간단하게 실행할 수 있다.

Ⅳ. 구현 및 결과

N개의 병렬 큐들을 사용하는 H-K 알고리즘 대신에 제안한 알고리즘은 하나의 선형 배열이나 스택에서 수행을 할 수 있다. 그림 2에서는 변형된 알고리즘의 매우 간단한 수행 기법을 나타내고 있다.

도착한 버퍼는 매 타임슬롯마다 많아야 N의 비율인 모든 N 라인에 도착하는 패킷들을 위한 메모리 저장 영역이다. 주어진 라인의 주어진 슬롯에서 패킷 도착에 대한 확률은 $p < 1$이고, Np는 평균 비율이다. 본 논문의 시뮬레이션 결과에서 최대 처리율을 결정하기 위하여 과부하의 경우 $p = 1$로 가정하였다.

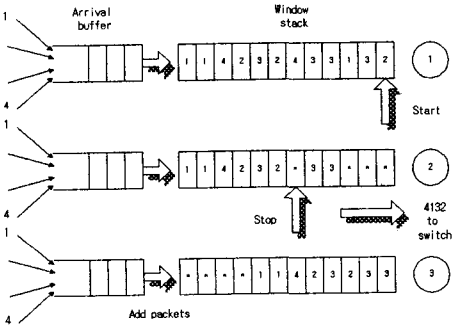

그림 2. 스위치 구성도

하나의 타임슬롯에서 많아야 오직 N 패킷이 출발할 수 있고, 각 타임슬롯에서 N 패킷들이 도착하고, 도착한 버퍼의 요구로 항상 패킷들을 이용할 수 있다. 패킷들은 슬롯 위치에서 FIFO 순서에 의해 도착한 버퍼로부터 밖으로 제공되어진다. 여기에서 타임슬롯 i에서 도착한 N 패킷들은 타임슬로 i+1이나 그 이후에 패킷들이 도착하기 전에 모두 밖으로 나가게 된다. 그러나 각 타임슬롯에서 N 패킷들의 출발 순서는 무작위화되고, 모든 입력 라인들 중에서 오른쪽부터의 액세스는 공정성을 유지하기 한다. 하나의 타임슬롯에서 세 가지의 스위치 동작을 보여주고 있다. 윈도우 스택은 Nw 요소들이 선형적으로 구성된 메모리 공간이다. 처음에는 Nw 패킷들이 가득 찬 스택에서부터 시작한다. 오른쪽에서 왼쪽으로 하나씩 이동하는 스위치 포인터는 윈도우 내에서 모든 적합한 패킷들을 가리키고, 그것들이 스위치 구조로 액세스되는 것을 승인한다. 이렇게 하나씩 이동하면서 작동하는 마지막 부분에서 윈도우 스택은 오른쪽으로 가득 차게 옮겨지며, 그 결과 왼쪽 끝 부분에 비어있는 곳은 도착한 버퍼로부터 새로운 패킷들로 채워진다. 이렇게 하나씩 이동하면서

모든 단계를 수행한다. 제안한 알고리즘은 하나의 선형 배열과 매 타임슬롯마다 하나씩 이동하면서 동작을 하는 기간에 매우 간단한 수행을 제공한다. 그러므로 논리적인 배열이 하드웨어에서나 소프트웨어에서 수행 동작을 용이하게 하는 증거이다. 시뮬레이션 프로그램은 그림 2에 나타나 있는 데이터 구조와 동작을 이용하였다. 제안한 알고라즘의 수행을 시뮬레이션에 의해서 조사하였고, 시뮬레이션 결과들은 표1에 나와있는 H-K 알고리즘과 비교하여 개선된 처리율을 표 2에 나타내었다.

N	윈도우 크기 w							
	1	2	3	4	5	6	7	8
2	.75	.84	.89	.92	.93	.94	.95	.96
4	.66	.76	.81	.85	.87	.89	.91	.92
8	.62	.72	784	.82	.85	.87	.88	.89
16	.60	.71	.77	.81	.84	.86	.87	.88
32	.59	.70	.76	.80	.83	.85	.87	.88
64	.59	.70	.76	.80	.83	.85	.87	.88
128	.59	70	.76	.80	.83	.85	.86	.88

표 1. H-K 알고리즘 수행 결과

N	윈도우 크기 w							
	1	2	3	4	5	6	7	8
2	.75	.86	.90	.94	.94	.95	.97	.97
4	.66	.82	.87	.91	.93	.94	.96	.97
8	.62	.79	.84	.89	.92	.92	.94	.95
16	.60	.77	.85	.89	.91	.93	.93	.94
32	.59	.77	.84	.89	.91	.92	.94	.94
64	.59	.77.	.84	.87	.91	.92	.93	.94
128	.59	77	.84	.87	.90	.92	.93	.94

표 2. 제안한 알고리즘 수행 결과

그림 3과 4는 두 개의 알고리즘 수행을 그래프적으로 비교하였다.

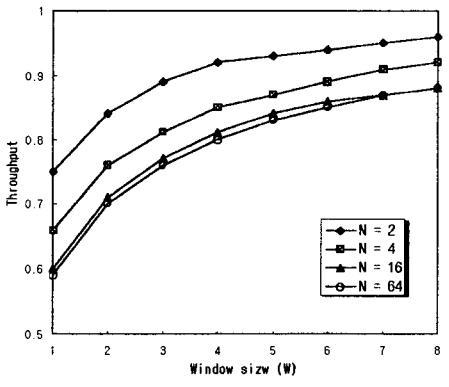

그림 3. 기존 방식의 알고리즘 수행

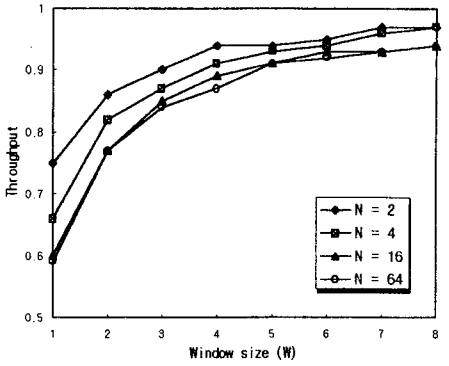

그림 4. 제안한 방식의 알고리즘 수행

V. 결 론

본 논문에서는 입력 버퍼형 스위치에서 패킷들의 윈도우 서비스에 대한 Hluchyj-Karol 윈도우 기법보다 개선된 처리율과 공정한 특징을 나타내었다. 하나의 타임슬롯에 제공되지 않는 큐들의 수를 최소화하고, 하나의 타임슬롯에서 이동되어지는 패킷들의 수를 최대화하는 방식을 선택하였다. 그러므로 입력 버퍼형 스위치에 대한 Hluchyj-Karol 윈도우 방식보다 개선된 시뮬레이션 결과를 보였다. 특히 적절한 크기의 윈도우에 대하여 제안한 방식은 공정한 특징을 보여주었고, 또한 소프트웨어와 하드웨어에서 비교적 간단한 수행을 하는데 제한을 받지 않는 것을 나타내었다. 처리율의 비교는 기존의 방식과 비교하여 10% 이상 개선되었음을 보여주었고, 더욱 실질적인 개선은 수행과정의 복잡성을 낮추고, 훨씬 쉽게 하도록 윈도우 크기를 작게 한 부분이다.

참 고 문 헌

[1] G. Thomas and J. Man, "Improved Windowing Rule for Input Buffered Packet Switches," *Electronics Letters, 18th Vol.* 29, No. 4, pp.393~395. Feb. 1993.

[2] J. J. Li, "Improving the Input-queuing Switch Under Bursty Traffic," *Electronics Letters, 25th Vol. 31,* No. 11, pp.854~855. May, 1995.

[3] N. K. Sharma and M. R. Pinnu, "A Simple Bypass Queue for Bursty Traffic," *Australian Telecommunication Networks & Application Conference,* melbourne, Australia. Dec. 1996.

[4] M. G.. HLUCHYJ and M. J. KAROL, "Queuing in High-Performance Packet Switching", *IEEE Journal on Selected Areas in Communications, vol.5,* no.9, pp. 1587-1597. 1988,