논문 00-25-1A-16 한국통신학회논문지 '00-1 Vol.25 No.1A

입력단 큐잉 방식의 ATM 스위치를 위한 효율적 셀 중재 방식에 관한 연구

정회원 김 용 웅**, 원 상 연*, 박 영 근*

An Effective Cell Scheduling Algorithm for Input Queueing ATM Switch

Yong Woong Kim**, Sang Youn Won*, Young-Keun Park* *Regular Members*

요 약

본 논문에서는 광대역 종합 정보 통신망을 구현하는데 필수적인 ATM 스위치의 방식 중 입력단 큐잉 방식에 적용할 수 있는 셀 중재 기법으로 입력단 큐잉방식에서의 HOL (Head-of-Line) 블럭킹과 출력단 충돌을 개선하여 입력단과 출력단의 매칭이 최대가 되도록 하는 기법인 MUCS(Matrix Unit Cell Scheduler)를 개선한 WMUCS (Weighted Matrix Unit Cell Scheduler)를 제안한다. WMUCS는 MUCS의 장점인 단순한 알고리즘과 높은 처리율에 대한 특성은 그대로 지니면서 더 좋은 특성을 보여주었다. 그리고 MUCS의 문제점인 기근(starvation)현상을 보완하여 최대 처리율을 거의 100%로 출력단 큐잉 방식에 근접하는 뛰어난 결과를 얻어내었다. WMUCS의 성능 분석을 위해 소프트웨어로 시뮬레이션하였다. 가장 중요한 세 가지 파라미터는 최대 처리율과 평균 지연, 그리고 셀 손실률이다. 최대 처리율은 예상한대로 순수한 MUCS보다 다소 개선되었다. 평균 지연은 버스티 트래픽의 경우에 개선 효과가 뚜렷했다. 셀 손실률도 WMUCS보다 우수한 수준이다.

ABSTRACT

In this paper, we propose a cell scheduling algorithm for input queueing ATM switch. The input queueing architecture is attractive for building an ultra-high speed ATM (Asynchronous Transfer Mode) switch. We propose a WMUCS (Weighted Matrix Unit Cell Scheduler) based on the MUCS which resolves HOL blocking and output port contention. The MUCS algorithm selects an optimal set of entries as winning cells from traffic matrix (weight matrix). Our WMUCS differs from the MUCS in generating weight matrices. This change solves the starvation problem and it reduces the cell loss variance. The performance of the proposed algorithm is evaluated by the simulation program written in C++. The simulation results show that the maximum throughput, the average cell delay, and the cell loss rate are significantly improved. We can see that the performance of WMUCS is excellent and the cost-effective implementation of the ATM switch using proposed cell scheduling algorithm.

I. 서 론

광대역 종합 정보 통신망 (B-ISDN : Broadband -Integrated Service Digital Network)의 구현에 있어서 핵심적인 기술인 비동기 전송 모드 (ATM : Asynchronous Transfer Mode)는 통신망의 한정된 자원인 대역폭을 통계적으로 다중화(statistical multiplexing)하여 효과적인 대역폭 이용이 가능하고 서비스 품질(QoS : Quality of Service)을 보장하는

* 연세대학교 기계·전자공학부 (ypark@yonsei.ac.kr)
** LG정보통신 교환전송OBU전송연구소 다중통신실 (elcayong@lgic.co.kr)
논문번호 : 99293-0723 접수일자 : 1999년 7월 23일
※ 이 논문은 한국과학재단 특정기초연구(1999-2-303-005-3)지원으로 수행되었음.

데 매우 적합하여 차세대 표준으로 자리잡아 가고 있다[1][2]. ATM은 하나의 독립된 네트워크 기술로서 매우 광범위한 연구 분야를 갖고 있으며, 그 중에서도 155.52Mbps정도의 고속 데이터 전송 속도에서 빠르게 데이터를 교환해 줄 수 있는 ATM스위치는 매우 중요한 위치를 갖고 있다고 볼 수 있다. 이러한 위치의 설계에 있어서 버퍼의 위치는 매우 중요한 이슈이다.

버퍼의 관점에서 ATM 스위치는 크게 출력단 큐잉 방식, 공유 메모리 방식, 입력단 큐잉 방식으로 나눌 수 있다[1][3]. 출력단 큐잉 방식은 N개의 입력 포트에 도착한 모든 셀은 그 타임 슬롯에서 해당 목적지 출력단으로 전송된다. 이상적인 최대 처리율과 평균 지연 특성을 보여준다. 그러나 메모리의 속도를 포트 수만큼 올려주어야 하기 때문에 스위치 패브릭과 버퍼 메모리의 구현이 어렵다. 특히 같은 출력 포트로 향하는 셀이 연속으로 들어오는 버스티 트래픽(bursty traffic) 환경에서는 셀 손실률이 크다[2]. 공유 버퍼링 방식은 N개의 입출력 포트를 갖는 스위치에서 단일 메모리를 모든 출력 포트가 공유하는 구조이다. 버퍼 공유 효과(buffer sharing effect)에 의해 스위치 구현에 필요한 버퍼 메모리의 양을 줄일 수 있는 장점이 있다. 또한 처리율이 출력단 큐잉과 같이 아주 우수하다. 그러나 메모리의 속도를 $2N$배 높여야 하며, 구조상의 복잡함 때문에 최근에 중요성이 더해지고 있는 스위치의 용량 $N \times N$을 크게 하는 확장성에 문제가 있다. 입력단 큐잉 방식은 스위치 패브릭(switch fabric)의 구현이 쉽고, 구조가 간단하며, 포트나 메모리의 속도를 올릴 필요가 없고, 셀 손실률(CLR : Cell Loss Rate)이 낮은 장점이 있다. 그리고 주어진 셀 손실률을 보장하기 위해 필요한 버퍼의 크기도 다른 방식에 비해 적다. 이런 점 때문에 확장성이 다른 방식에 비해 우수하다. 또한 셀 지연에 관한 특성이 우수하여 최근에 중요성이 커지고 있는 시간에 민감한 멀티미디어 트래픽에 대한 서비스 품질 보장이 용이하다. 그러나, HOL 블럭킹(Head Of Line blocking)문제, 입력단 충돌, 출력단 충돌로 인하여 스위치의 처리율(throughput)이 다소 떨어진다는 문제점이 있다[3].

이런 한계를 극복하는 보편적인 방법으로, FIFO (First In First Out) 큐잉과 바이패스 큐잉(bypass queueing) 혹은 FIRO(First In Random Out)이 있다. 이 방식은 한 타임 슬롯에 얼마나 많은 셀을 전송할 수 있는가 하는 문제와 셀 손실률 및 평균 지연을 개선하기 위해 준최적화된 셀 중재(cell scheduling or cell arbitration)가 매우 중요하다고 볼 수 있다[2].

이런 관점에서 그 동안 여러 가지 셀 중재 기법이 발표되었고, 처리율 면에서는 이상적인 경우라고 할 수 있는 출력단 큐잉 방식에 버금가는 성능을 보여준 것도 여러 가지가 있다. 그 중에서도 MUCS (Matrix Unit Cell Scheduler)는 일종의 최대수 매칭(maximum size matching) 방법으로 아주 뛰어난 처리율을 보여준다[4][5][6]. 이 기법은 입력단 충돌과 출력단 충돌이 될 수 있는 한 적게 되도록 입력 포트와 출력 포트를 선택하는 과정을 반복하여 이 후 과정에서 경합에 참여하는 입출력단 쌍의 수가 최대가 되게 하는 것이다. 본 논문에서는 MUCS를 기초로 셀 손실률이나 평균 지연 등의 특성을 향상시킬 수 있는 개선된 알고리즘을 제안한다.

Ⅱ. 입력단 큐잉 방식의 ATM 스위치와 기존 셀 중재기법

2.1. 입력단 큐잉 방식의 ATM 스위치

ATM 교환 시스템은 가상 회선 교환 방식으로서 통계적 다중화를 기본으로 하는 전송 방식이다. 그러므로 동일한 출력 포트로 향하는 셀을 동시에 전달할 수 없다. 따라서 같은 출력 포트 주소를 갖는 셀 중에서 하나를 선택해 교환하고 나머지는 버퍼에 저장하여야 한다. 근래에는 다양한 교환기 구조가 제안되어 왔고, 여러 가지 고성능 패킷 교환기 구조를 찾아 볼 수 있다[7].

출력 포트 주소가 같은 셀이 동시에 도착하여 이들을 한꺼번에 처리할 경우 출력 포트에서 충돌이 발생하므로 입력 포트에 셀을 저장하는 버퍼를 두어 충돌이 발생할 수 있는 셀을 입력 버퍼에 저장하는 방법이 입력단 큐잉(input queueing) 방법이다. 입력 버퍼에 저장된 셀은 선입 선출(FIFO : First In First Out)에 의해 한 타임 슬롯(2.73 μsec)에 하나의 셀이 처리되므로 이전에 도착한 셀이 처리될 때까지 그 입력 포트에 도착하는 셀은 대기하였다가 다음 타임 슬롯에서 전송되어야 한다. 입력단 큐잉 방식은 출력단 충돌, HOL 블럭킹(Head Of Line blocking)문제, 입력단 충돌로 인하여 스위치의 처리율(Throughput)이 다소 떨어진다는 문제점이 있다. 그림 1은 입력단 큐잉 방식의 문제점이 나타내고 있다. 이런 문제점 때문에 최대 처리율이 $(2-\sqrt{2}) \approx 0.586$정도로 제한된다[3].

그림 1. 입력단 큐잉의 세 가지 문제점

이런 입력단 큐잉 방식의 단점을 개선하여 최대 처리율을 향상하기 위한 여러 가지 방법이 연구되어 왔다. 스위치 패브릭 구조를 개선하므로써 성능을 개선할 수 있다. 입력단을 확장하고 널리 알려진 LA(Look Ahead) 셀 중재 기법을 결합하여, 한 타임 슬롯에 복수의 셀이 경합에 참여함으로써 입력단 충돌 문제를 개선할 수 있다. 이런 기법을 입력단 확장 방식이라고 한다[8]. 스위치 패브릭은 $sN \times N$가 된다. 입력단 확장비 s를 2로 하고 무작위 트래픽을 인가하면, $N=\infty$ 일 때, 처리율이 0.764 정도이다. 이와 비슷한 접근 방식으로 출력단을 확장하여 한 타임 슬롯에 복수의 셀이 출력 포트로 전송되도록 할 수 있다. 출력단 충돌 현상을 개선하는 이런 기법을 출력단 확장 방식이라고 한다. 출력단 확장비 r을 2로 하고 무작위 트래픽을 인가하면, $N=\infty$ 일 때, 처리율이 0.885 정도이다. 또한 입력단 큐에 들어오는 셀을 비순차적으로 처리하여 HOL 블럭킹과 입력단 충돌을 해결함으로써 성능을 극대화한다[9][10][11]. 이런 입력단 큐잉을 바이패스 큐잉(bypass queueing)이라고 하며 최대 처리율을 상당히 향상시킬 수 있다. 바이패스 큐잉은 스위치 패브릭의 개선 없이 구현할 수 있는 매력이 있다. 이 경우에는 셀을 어떻게 선택하느냐가 처리율에 큰 영향을 주기 때문에 좋은 최적화된 셀 선택 집합을 찾아내는 것이 중요하다[9][11].

2.2. 기본적인 셀 중재 기법

위에서 언급한 바와 같이 입력단 큐잉 방식은 몇몇 문제점만 보완하여 스위치의 최대 처리율을 향상시킬 수 있다면 스위치 패브릭의 단순함과 양호한 셀 손실률 등 여러 가지 장점이 있다. 스위치 패브릭을 확장하지 않고 순차적 큐잉 (FIFO queueing)의 약점을 극복하는 바이패스(by-pass) 큐잉에 관련된 많은 연구가 발표되어 왔다. 패스 큐잉에 대한 초기의 접근은 대부분 각 입력 포트에 대해 순차적인 방법으로 최적화에 가까운 블럭킹 없는 셀 집합을 찾아내는 셀 중재 알고리즘을 제시하였다[10]. 이것들은 대부분 균일(uniform) 트래픽에 대해서는 양호한 성능을 보여주었는데, 비교적 최근의 연구는 비균일(non-uniform) 트래픽, 즉 버스티 트래픽(bursty traffic)이나 핫스팟(hot spot)에 대한 분석에 대해 중요성을 강조하고 있다[2].

기본적인 큐 형태인 FIFO 큐는 버퍼에 들어온 순서대로 셀이 나갈 수 있다. 이런 큐잉 방식은 최대 처리율이 안 좋지만 평균 지연의 특성이 좋고 셀의 전송순서가 바뀌지 않으며 입력포트 간에 비슷한 정도의 대역폭을 보장해 줄 수 있다. 그리고 최근 시간에 민감한 실시간 멀티미디어 셀의 평균 지연에 대한 한계를 보장할 수 있기 때문에, 최근 기존에 입력포트의 FIFO 큐에 적용되던 알고리즘과 최대 처리율에 중점을 둔 알고리즘을 절충하여, 멀티미디어 셀의 지연 특성을 향상시키고, 데이터 셀의 셀 손실률을 개선하여 최대 처리율을 높이는 두 가지 목표를 만족시키는 방식에 대한 연구가 되고 있다[12]. 그러나 FIFO 큐잉은 그 근본적인 문제 때문에, 비록 앞에서 다룬 여러 가지 셀 중재 방식이 성능을 개선시켜 주기는 하지만, 스위치의 가장 중요한 성능평가 요소인 최대 처리율은 아무래도 출력단 큐잉 방식이나 공유 버퍼링 방식에 비해 상당히 떨어진다. FIFO 큐잉 대신에 바이패스 큐잉, 즉 FIRO (First In Random Out)을 채택하면 최대 처리율이 상당히 개선된다. 이는 HOL 블럭킹 문제가 해결되기 때문이다.

바이패스 큐잉에 적용되는 셀 중재 기법 중에서 가장 기본적인 것인 것은 윈도우 기반 셀 스케줄링 (window-based cell scheduling)이다. 이 기법은 HOL 블럭킹이 일어난 입력 포트에서 HOL 셀의 뒤에 저장되어 있는 셀들의 출력 주소를 보고 출력단 충돌이 일어나지 않는 셀을 선택하여 전송하는 것이다. 입력 버퍼는 FIRO (First In Random Out)라고도 하는 데서 알 수 있듯이 윈도우의 크기 W 내에서는 셀들이 임의의 순서로 전송이 가능하다. 단, 이 때 같은 버퍼 내에서 출력 주소가 같은 셀들을 전송 순서가 바뀌지 않도록 주의해야 한다. 입력 포트 간에는 순차적으로 W번 경합을 한다. LA (Look Ahead)기법으로도 불리는 이 방식은 윈도우의 크기가 클수록 처리율이 높다.

기존의 버퍼 선택 방법을 개선한 알고리즘으로는 무작위 선택, 순환 우선권 부여 (rotating priority), LQFA (Longest Queue First Allocation) 등이 있

다. 이 중에서 입력단 큐에 대기하고 있는 셀의 수에 비례하여 우선권을 부여하는 LQFA 방법이 상당히 좋은 특성을 보여주는데, 이는 셀 손실의 원인인 버퍼 넘침(buffer overflowing)이 일어날 가능성을 줄이는 데 적합하기 때문이다[13][14].

Ⅲ. Weighted Matrix Unit Cell Scheduler

그 동안 발표된 여러 가지 기법들은 나름대로 장단점을 가지고 있었다. 근래의 것으로는 PIM (Parallel Iterative Matching) 계열이나 2DRR(Two Dimensional Round Robin) 알고리즘이 있다[15][16]. 본 논문에서는 최근에 발표되어 HOL 블럭킹과 출력단 충돌을 해소하여 대역폭의 이용률을 높여 최대 처리율을 거의 출력단 큐잉에 가깝게 하면서 하드웨어 구현이 비교적 용이한 것으로 알려진 MUCS(Matrix Unit Cell Scheduler)를 고찰하고 이를 개선시킨 알고리즘을 제안한다.

3.1. Matrix Unit Cell Scheduler 기법

셀 중재 초기에 트래픽 행렬 A를 만든다. 스위치의 크기가 $N\times N$짜리 일 때, 행렬의 크기는 $N\times N$이 된다. 각각의 파라미터 a_{ij}는 입력 포트 i 에서 출력 포트 j 로 가는 셀이 버퍼에 있는 지의 유무를 나타낸다. 그러므로 a_{ij}는 음이 아닌 정수 값을 가지며 0 혹은 1이다. $a_{ij} > 0$ 일 때는, 입력 포트 i에서 출력 포트 j로 가는 셀이 존재함을 나타낸다. $a_{ij} = 0$ 일 때는, 입력 포트 i에서 출력 포트 j로 향하는 셀이 존재하지 않음을 나타낸다[17]. 그 다음 트래픽 행렬 A로부터 가중치 행렬(weight matrix) W를 만든다. 가중치 행렬의 크기도 역시 $N\times N$이다. 이 행렬의 파라미터 w_{ij}를 계산하고 나서, W의 요소 중 가장 큰 w_{ij}를 선택하여 차례로 반복하고(iteration) 갱신하여 행렬의 크기를 줄여 A'를 만든다. 다음 단계에서도 같은 방법으로 트래픽 행렬을 줄여나가서 입출력단을 선택하여 최적화된 셀 선택 집합을 선택한다. 이 기법의 핵심은 입력단 충돌과 출력단 충돌이 가장 작아지는 입출력단 쌍을 선택하여 그 다음 단계에서 고를 수 있는 대상이 되는 입출력단 쌍이 최대가 되도록 하는 것이다. 이런 일련의 알고리즘은 하드웨어로 구현하여 동시에 병렬적으로 동작하도록 구현이 가능하다[6]. 이제 MUCS 알고리즘의 과정과 특성을 알아보면, $N\times N$을 스위치의 크기라 하고 $M\times M$을 크기가 줄어든 행렬 A'의 크기라고 하자. 알고리즘의 단계는 다음과 같다.

1) 초기에는 $M=N$이고 $A' = A$ 이다.

2) 각각의 반복 과정(iteration round)에서 가중치 행렬의 요소인 w_{ij}를 매 과정마다 a_{ij}로부터 다음의 식을 이용하여 계산한다.

$$w_{ij} = \frac{1}{wr_i} + \frac{1}{wr_j}, \quad \text{if } a_{ij} = 1 \tag{1}$$

$$w_{ij} = 0, \quad \text{if } a_{ij} = 0 \tag{2}$$

이때,

$$wr_{ij} = \sum_{n=1}^{M} a_{ij} \tag{3}$$

$$wc_{ij} = \sum_{n=1}^{M} a_{ij} \tag{4}$$

3) M^2 개의 w_{ij} 중에서 큰 제일 것을 선택한다. 이때 제일 큰 값이 복수개일 경우 i와 j가 겹치지 않는다면 여러 개를 동시에 선택해도 된다. i가 겹치지 않도록 하는 것은 한 입력 포트에서 하나만 선택하기 위함이다. j가 겹치지 않도록 하는 것은 출력단 충돌을 방지하기 위함이다. 일단 이렇게 선택된 a_{ij}에 대하여 i행과 j열을 모두 제거하고 크기가 줄어든 새로운 트래픽 행렬 A'를 만든다. 그리고 나서 역시 크기가 줄어든 A'에 대한 가중치 행렬 M'를 만든다. 이제는 같은 과정을 반복하는데 N개의 a_{ij}가 선택되면 중단한다.

4) 만일 가중치가 같고 i 와 j가 달라 선택 가능한 것이 여러 개 있다면 무작위로(randomly) 선택한다.

그림 2에서 예로 4×4 스위치에 MUCS 기법을 적용한 것을 보였다. 첫 번째 가중치 행렬에서 3행 4열의 w_{34}의 값이 2로 가장 크기 때문에 이것이 선택된다. 그리고 나서 3행과 4열의 값을 소거한 두 번째 트래픽 행렬을 만든다. 이것을 바탕으로 두 번째 가중치 행렬을 만들면 1행 2열과 2행 1열의 값이 1.5로 같은데 서로 입출력단이 겹치지 않으므로 w_{12}와 w_{21}이 선택된다. 이제 1행과 2열을 모두 소거한 트래픽 행렬에서 세 번째 가중치 행렬을 만든다. 남은 값이 4행 3열에 있는 w_{43}이 선택된다. 이런 세 단계를 거쳐서 (3,4), (1,2), (2,1), (4,3)의 입출력단 쌍이 선택되었다.

MUCS는 셀 도착률 1.0을 인가했을 때, 균일 무작위 트래픽일 때 1.00에 가까운 이상적인 결과를

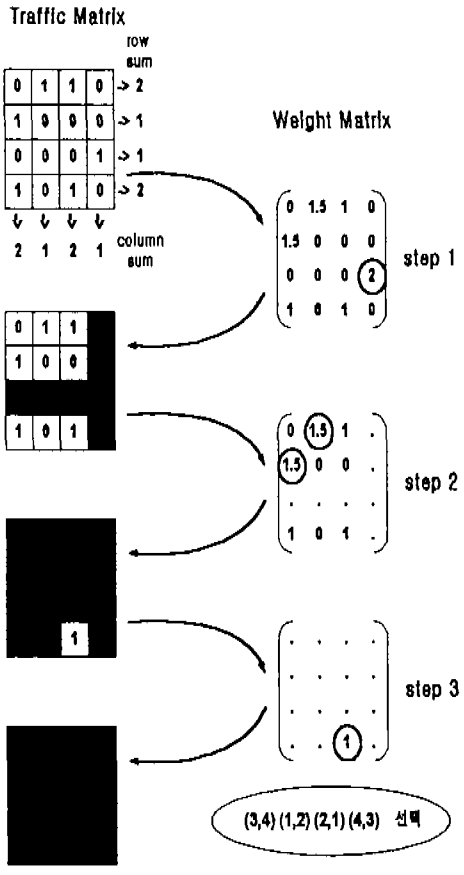

그림 2. MUCS 기법의 적용 예

보여준다. 버스티 길이가 5와 10일 경우 각각 0.91과 0.83 정도를 나타낸다. 이 때의 평균 큐 길이는 각각 40과 140 정도이다. 이 정도의 값은 버스티 트래픽 환경에서 MUCS의 성능이 별로 저하되지 않는다는 것을 알 수 있다. 특히 중요한 것은 이런 우수한 성능이 포트 수에 거의 제한을 받지 않고 유지된다는 것이다. 이는 MUCS의 확장성이 다른 알고리즘에 비해 매우 우수하다는 것을 보여준다[6].

따라서 대용량 ATM 스위치의 구현에 응용하기에 적합하다. 그러나 이 MUCS 알고리즘은 다음과 같은 문제점을 가지고 있다.

ATM 셀 중재 기법에서, 입력단의 제일 중요한 관심사는 패킷의 전송이고 출력단의 제일 중요한 관심사는 셀을 받는 것이다. 이런 관점에서 볼 때 입력단과 출력단의 짝짓기를 최대로 하는데 중점을 둔 MUCS는 매우 이상적인 처리율을 보여준다는 장점이 있다. 그런데 MUCS에서 트래픽 행렬을 만드는 과정을 보면 각 입력단에서 출력단으로 가는 패킷의 유무만을 따지기 때문에 각 입력 포트의 길이를 고려하지 않는다. 이는 큐 길이의 변이가 커져서 버퍼 넘침으로 인한 셀 손실률의 특성이 나빠질 위험이 크다.

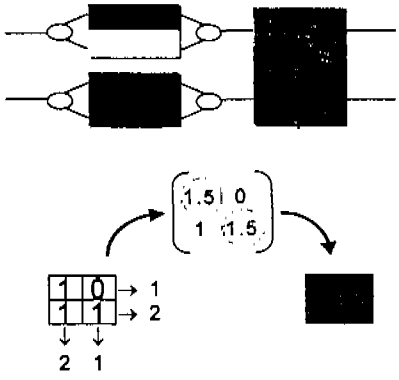

그림 3. MUCS의 기근 문제

이는 기근(starvation)이 발생하기 때문이다. 무작위로 선택하는 알고리즘에서는 기근이 발생하지 않지만 MUCS는 최대 수 매칭을 지향하기 때문에 기근이 발생한다. 그림 3에 기근에 대한 개념을 도시하였다. 그림과 같이 입력 포트 2에서 출력 포트 1로 나가는 셀은 선택될 기회가 좀처럼 없다. 이는 큐간의 공평성에 심각한 문제를 야기할 뿐더러 입력단 큐잉의 장점인 셀 손실률에도 나쁜 영향을 미친다. 본 논문에서는 이런 여러 가지 단점을 보완하고 장점을 살릴 수 있도록 MUCS를 개선한 새로운 알고리즘을 제안한다.

3.2. 제안하는 Weighted MUCS 기법

입력단과 출력단을 짝짓는 수를 최대로 하는 데 중점을 둔 MUCS는 기근 문제와 입력 포트간의 공평성이 지켜지지 않는 문제가 있다. 이런 문제를 해결하기 위해 본 논문에서는 트래픽 행렬을 생성하는 과정에 변화를 주었다.

MUCS의 트래픽 행렬에서 각각의 파라미터 a_{ij}는 입력 포트 i에서 출력 포트 j로 가는 셀이 버퍼에 있는 지의 유무를 나타낸다. 그러므로 a_{ij}는 음이 아닌 정수 값을 가지며 0 혹은 1이다. 이 부분에서, 셀이 없을 때 0으로 하는 것은 같지만 셀이 있을 경우에 단순히 유무를 1로 표시하는 것이 아니라, 그 개수를 세서 음이 아닌 정수로 표현한다.

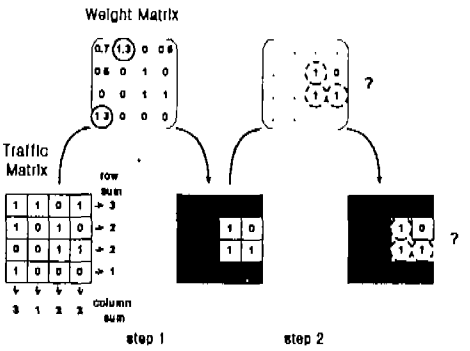

그림 4. MUCS 알고리즘 적용

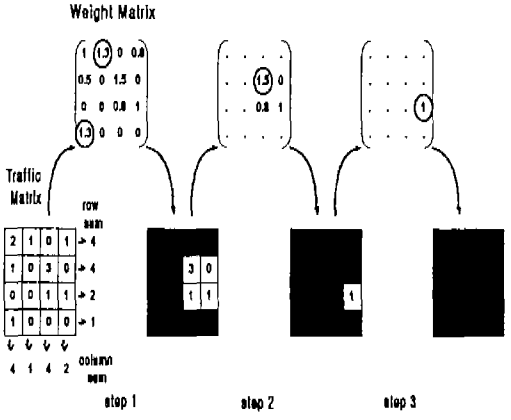

그림 5. WMUCS 알고리즘 적용

즉, 위의 식 (1)과 (2)에서 w_{ij}의 계산이 달라진다.

$$w_{ij} = \frac{a_{ij}}{wr_i} + \frac{a_{ij}}{wr_j}, \quad \text{if } a_{ij} > 1 \tag{6}$$

$$w_{ij} = 0, \quad \text{if } a_{ij} = 0 \tag{7}$$

이런 개선이 구체적으로 최대수 매칭에 어떻게 기여하는지 알아보자.

트래픽 행렬을 만드는 방법으로 그림 4는 순수한 MUCS를 적용하였고 그림 5는 개선된 MUCS를 적용한 것이다. MUCS를 적용하면 2단계에서 가중치가 1인 파라미터가 3개가 나온다. 이런 경우에는 무작위로 선택해야 하는데, 운 좋게도 w_{23}과 w_{32}를 선택할 경우에는 3단계에서 w_{33}을 선택하여 4개의 입출력단 쌍을 선택할 수 있다. 그러나 w_{33}을 선택하면 더 이상 진행하지 못하여 3개의 입출력단 쌍을 고를 수밖에 없다. 개선된 MUCS는 각 입력 포트에서 기다리는 셀의 수를 트래픽 행렬에 반영하기 때문에 가중치 행렬이 약간 다르게 나온다. 이 경우에 가중치 행렬의 파라미터 중에서 같은 값을 가질 확률은 버퍼가 안정되어 있는 한 거의 없다. 2단계의 가중치 행렬을 보면 MUCS에서 같은 값을 가졌던 w_{23}, w_{32}, w_{33}이 비로소 다른 값을 가진다. 이제 2 단계에서 w_{23}을 선택하고 나서 3단계에서 w_{32}를 선택하면 4개의 입출력단 쌍을 고를 수 있다. 이와 같이 WMUCS(Weighted MUCS)라 이름 붙인 알고리즘은 가중치 행렬에서 같은 값이 나와 선택할 수 있는 기회를 놓칠 수 있는 가능성을 없앴다. 또한 앞에서 언급한 기근문제에 대해 개선 효과를 기대할 수 있다. 왜냐하면, 기근으로 인해 선택받지 못한 입력 버퍼는 그만큼 셀이 쌓일 것이므로 타임 슬롯이 경과할수록 가중치 행렬에서 값이 높아질 것이기 때문이다. 물론 기존의 이산화된 (즉, 0 또는 1) 수치를 쓴 알고리즘보다 복잡도가 다소 늘어나기는 하지만 MUCS 알고리즘이 본래 단순하고 계산이 적은 편이기 때문에 최근의 하드웨어 속도라면 별 무리 없이 구현 가능하다고 예측할 수 있다.

이상으로 Weighted MUCS에 대한 이론적인 검토를 마치고 다음 장에서는 컴퓨터 시뮬레이션을 통해 다양한 트래픽 유형이나 파라미터의 변화에 따른 WMUCS의 성능을 평가하도록 하겠다.

Ⅳ. WMUCS의 성능 고찰

4.1. 시뮬레이션 환경

본 논문에서는 제안한 WMUCS 기법의 성능을 C++ 언어로 작성한 사건 발생 (Event-Driven) 시뮬레이션 프로그램을 이용하여 검증하였으며, 스위치에 인가하는 트래픽은 크게 균일 무작위 트래픽(uniform random traffic)과 버스티 트래픽(bursty traffic)이다. 또 확장성에 대한 성능을 측정하기 위해 스위치의 크기에도 변화를 주었다. 이런 각 요소에 대하여 셀 도착률을 변화시켜가면서 중요 측정 파라미터의 결과를 살펴보았다. 성능 측정 파라미터로는 최대 처리율, 셀 손실률과 평균 셀 지연이다.

시뮬레이션 프로그램은 C++ 언어로 작성되었으며, 실행은 펜티엄 II 컴퓨터에서 하였다. 인가하는 모든 트래픽에 대하여 다음의 조건을 공통적으로 부가하였다.

• 입력단 큐의 안정화 이후에 성능 파라미터를 측정하기 위하여 초기 1000 타임 슬롯은 계산에 넣

지 않았다.

• 전체 시뮬레이션 횟수는 10^6이상으로 하였다.

• 단순화를 위하여 입력 포트와 출력 포트의 수는 같다고 놓았다. (포트 수는 N 이고 스위치 크기는 $N \times N$으로 하였다)

• 입력 포트와 출력 포트의 처리 능력은 동일하게 하였다.

• 한 타임 슬롯은 셀이 스위칭을 거쳐 출력 포트로 나갈 때까지로 하였다.

시뮬레이션 프로그램의 기본 구조는 포트 크기, 셀 도착률, 트래픽 유형, 버스티 길이 등의 값을 입력받는 초기화 입력 부분, 셀을 균일 무작위로 혹은 버스티하게 생성하는 부분, 생성된 셀을 버퍼에 넣는 부분, 전송할 셀 집합을 선택하는 부분, 셀을 전송하는 부분, 결과를 파일로 출력하는 부분 등으로 구성되어 있다.

4.2. 시뮬레이션 결과 및 분석

먼저 스위치의 가장 기본적인 성능 파라미터인 최대 처리율에 대하여 살펴보자. 그림 6은 포트 수에 따른 최대 처리율을 보여주고 있다.

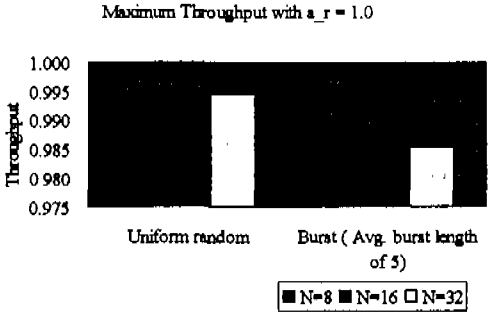

그림 6. 포트 수 N에 따른 최대 처리율

포트 수는 N=8, N=16, N=32로 하였다. 왼쪽은 균일 무작위 트래픽을 인가한 것이고 오른쪽은 평균 버스티 길이가 5인 버스티 트래픽을 인가한 것이다. 그래프에서 보듯이 균일 무작위 트래픽일 때는 포트 수 N에 상관없이 0.990을 상회하는 높은 처리율을 나타내었다. 버스트 트래픽을 때에는 균일 무작위 트래픽에 비해 다소 낮은 값이 나오지만 역시 0.985를 상회한다. 주목할 것은 버스티 트래픽의 경우에는 포트 수 N이 커짐에 따라 처리율이 영향을 받는다는 것이다.

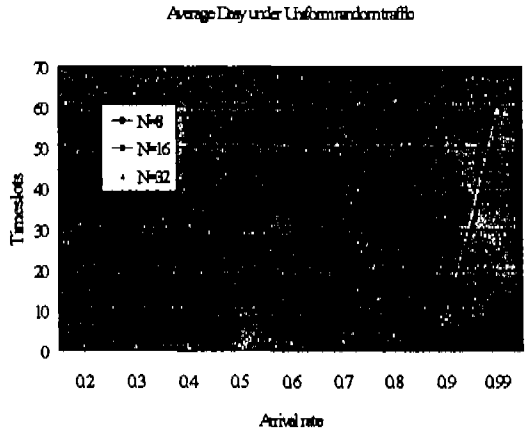

그림 7. 셀 도착률에 대한 평균 지연[균일 무작위 트래픽]

그림 7는 셀 도착률에 따른 평균 지연을 나타내고 있다. 균일 무작위 트래픽을 인가한 것이다. 셀 도착률이 0.7 이하일 경우에는 지연이 거의 완만하게 유지되는데 0.7을 넘어서면 지연이 커진다. 그림 8은 버스티 트래픽에 대한 평균 지연에 대한 것이다. 예측했던 대로 균일 무작위 트래픽보다는 버스티 트래픽에서 지연이 큰데, 0.7 이하에서는 대략 2배 정도이지만 0.7을 넘어서면 지연이 3배정도의 차이가 난다. 그림 7와 10에서 보면 스위치 포트 수의 증가가 평균 지연에 미치는 영향은 미미하다. 이는 WMUCS의 확장성을 확인할 수 있는 부분이다.

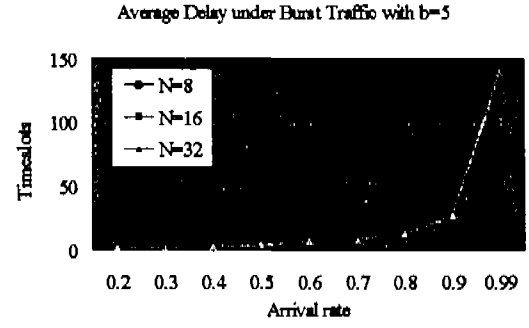

그림 8. 셀 도착률에 대한 평균 지연[버스티 트래픽]

그림 9는 균일 무작위 트래픽에 대한 셀 손실률을 보여준다. 10^{-6} 이하의 셀 손실률을 유지하기 위해서 N=8일 경우에는 큐 길이가 11정도 되어야 하고 N=16일 경우에는 큐 길이가 15정도 되어야 한

다. 포트 수가 두 배인데 추가로 필요한 큐 길이가 4라면 그리 손해보는 것은 아니다.

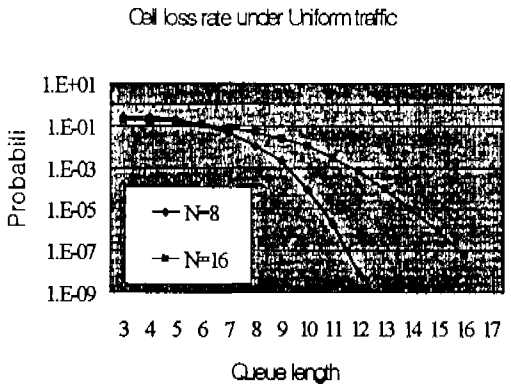

그림 9. 균일 무작위 트래픽에 대한 셀 손실률

그림 10에는 버스티 트래픽에 대한 셀 손실률을 나타내었다. 10^{-6} 이하의 N=8일 경우에는 큐 길이가 23정도 되어야 하고 N=16일 경우에는 큐 길이가 28정도 되어야 한다. 최근에는 10^{-7}정도의 셀 손실률을 기준으로 제시하는 경우도 종종 있는데, 버스티 트래픽에서 N=16 인 경우 필요한 큐의 길이가 약 30 정도 된다.

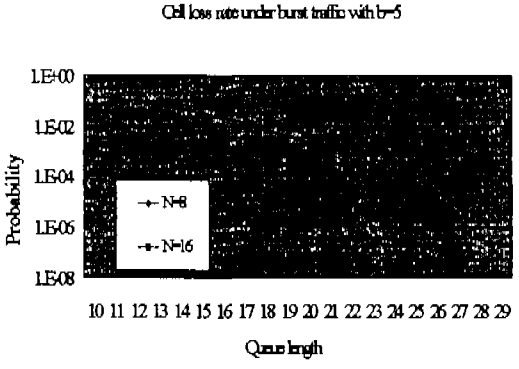

그림 10. 버스티 트래픽에 대한 셀 손실률

그림 11에서 평균 지연에 대하여 Weighted MUCS와 MUCS를 비교해 보았다. 셀 도착률이 0.6 이하일 때는 별 차이가 없는데 0.7 근처에서부터 차이가 나기 시작해서 0.99의 셀 도착률에서는 20 슬롯 정도의 차이가 난다. 이로부터 트래픽 부하가 높으면 Weighted MUCS의 평균 지연 특성이 MUCS 보다 뚜렷하게 양호한 결과를 보인다. 이는 기근 문제의 해소와 공평성 문제를 해소했기 때문인 것으로 보인다.

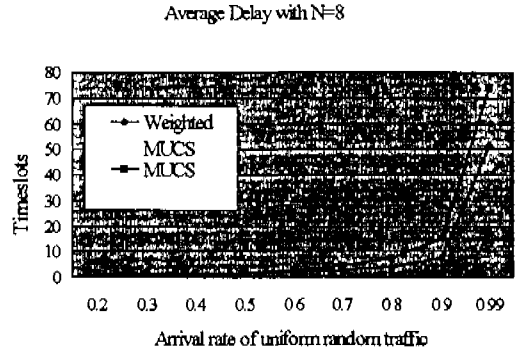

그림 11. 평균 지연으로 본 WMUCS와 MUCS

그림 12에서는 최대 처리율의 측면에서 WMUCS와 MUCS를 비교하였다. 균일 랜덤 트래픽일 때보다 버스티 트래픽일 때 처리율의 개선폭이 더 크다. 즉, 버스티 환경에서는 WMUCS가 MUCS에 비해서 특히 최대 처리율이 더 우수하다.

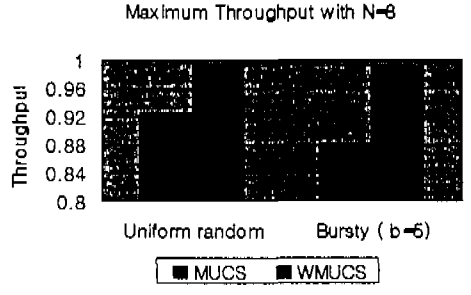

그림 12. 최대 처리율로 본 WMUCS와 MUCS

이상으로 제안된 WMUCS 기법의 성능을 시뮬레이션을 통해 분석해 보고 MUCS와 비교해 보기도 했다. 스위치 평가의 중요 파라미터인 최대 처리율, 평균 셀 지연, 셀 손실률을 평가해 본 결과 만족할 만한 성능을 보여주었으며 기존의 MUCS보다 개선된 면도 있었다. WMUCS에서 특히 주목할 것은 버스티 환경에서 우수한 최대 처리율과 포트 크기에 크게 영향을 받지 않는 성질이다. WMUCS는 높은 처리율을 보이는 여러 셀 중재 알고리즘 중에서 그 과정이 간결한 편에 속하기 때문에, 확장성이 매우 우수하다는 이점이 있다.

V. 결 론

본 논문은 처리율이 떨어나고 알고리즘이 간단한 Matrix Unit Cell Scheduler에 주목하고, 이를 분석하여 문제점을 개선한 Weighted MUCS를 제안하였다. MUCS 기법에 저장된 셀의 길이를 고려한 가중치를 두어 입력단과 출력단의 쌍이 최대로 선택되는 가능성을 더욱 높이는데 중점을 두었다. 시뮬레이션을 통한 다양한 성능 분석 결과는 양호한 결과를 보여준다. 최대 처리율은 이상적인 경우라 할 수 있는 출력단 큐잉 방식과 같은 결과가 나왔다. 이런 처리율의 특성은 트래픽 부하가 100%인 경우에도 유지됨을 알 수 있었다. 또 스위치의 크기가 증가하더라도 최대 처리율은 여전히 우수한 수준을 나타내는 결과를 보여주었다. 평균 지연에 대한 결과는 균일 무작위 트래픽에 비해 버스티 트래픽에서 다소 높은 값이 나온다. 이는 다른 알고리즘에서도 비슷하며 셀 손실률은 입력단 큐잉 방식에서와 같은 결과를 보여주었다.

시뮬레이션 결과에서 특히 주목할 것은 스위치의 크기가 커져도 성능 인자에는 영향을 주지 않는다는 사실이다. MUCS 알고리즘이 하드웨어적으로 구현이 용이한 것으로 알려져 있는데, WMUCS 또한 간결한 알고리즘에 속하기 때문에 하드웨어적으로 구현하는데 어려움이 없을 것으로 보인다.

참 고 문 헌

[1] Perter Newman, "ATM technology for corporate networks", *IEEE Communications Magazines*, pp. 90-101, April, 1992

[2] Young-Keun Park, Gyunho Lee, "Neural network-based ATM cell scheduling with queue length-based priority scheme", *IEEE Journal on Selected Areas in Communications-Computational and Artificial Intelligence in High Speed Networks*, Vol.12 No.2 1997.2.

[3] Michael G. Hluchyj, Mark J. Karol, "Queueing in high-performance packet switching", *IEEE JSAC in communications*, vol.6, December 1988

[4] H. Duan, "Design and development of cell queueing, processing, and scheduling modules for iPOINT input-buffered ATM testbed", *Ph.D. Dissertation*, Univ. Illinois at Urbana-Champaign, 1997.

[5] H. Duan, J.W. Lockwood, S.M. Kang, "A High-performance OC-12/ OC-48 queue design prototype for input-buffered ATM switch", *IEEE Infocom'97*, pp.20-28, 1997.

[6] H. Duan, J.W. Lockwood and S.M. Kang, "Matrix Unit Cell Scheduler (MUCS) for Input-Buffered ATM Switches", *IEEE Communication Letters*, vol.2, no.1, pp.20-23

[7] R. Rooholamini, V. Cherkassky, and M. Garver, "Finding the right ATM switch for the market", *IEEE Computer Magazine*, pp. 16-28, April 1994.

[8] R.Y. Awdeh, H.T. Mouflah, "Survey of ATM Switch Architecture", *Computer Networks and ISDN Systems*, Vol. 27, pp. 1567-1613, 1995.

[9] M.G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching", *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1587-1597, Dec. 1988.

[10] T.X. Brown and K. H. Liu, "Neural network design of a Banyan network controller", *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 8, pp. 1428-1438, Oct. 1990.

[11] K.W. Sarkies, "The bypass queue in fast packet switching", *IEEE Trans. Communications*, vol. 39, no. 5, pp. 766-774, May 1991.

[12] Bin Li, Mounir Ham, Xi-Ren Cao, "An efficient scheduling algorithm for input-queuing ATM switches", *IEEE*, 1997.

[13] M.J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch", *IEEE Trans. Communications*, vol. com-35, no. 12, pp. 1347-1356, Dec. 1987.

[14] Y. Tamir and H. C. Chi, "Symmetric crossbar arbiters for VLSI communication switches", *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 1, pp. 13-27, Jan. 1993.

[15] T.E. Anderson, "High Speep Switch Scheduling for Area Networks", *ACM Trans.*

on computer systems, Vol/ 11. NO.4, pp. 319-352, Nov. 1993

[16] R.O. Lamaire, D.N. Serpanos, "Two-Demensional Round-Robin Schedulers for Packet Switches with Multiple Input Queues", *IEEE/ACM Trans. on Networking*, Vol. 2, No.3, pp. 471-482, Oct. 1994.

[17] H. Duan, J.W. Lockwood, S.M. Kang, "A 3-dimensional queue (3DQ) for practical ultra-broadband input buffered ATM switches", *IEEE Tran. VLSI Syst*, submitted for publication.

김 용 웅(Yong Woong Kim)

1997년 2월 : 연세대학교 전기공학과(학사)
1999년 2월 : 연세대학교 대학원 전기컴퓨터공학과(석사)
1999년 2월~현재 : LG정보통신 (연구원)

원 상 연(Sang Youn Won)

1998년 2월 : 연세대학교 전기공학과(학사)
1998년 8월~현재 : 연세대학교 대학원 전기컴 퓨터공학과 석사과정

박 영 근(Young-keun Park) 정회원

1986년 2월 : 연세대학교 전기공학과(학사)
1990년 8월 : University of Minnesota 전기공학과 (석사)
1993년 12월 : University of Minnesota 전기공학과 (박사)
1994년 1월~1996년 2월 : University of Minnesota 전기공학과(Post Doc.)
1996년 3월~현재 : 연세대학교 기계전자공학부 조교수
<주관심 분야> 컴퓨터 네트워크, 멀티미디어 통신망, ATM 교환기, 신경회로망 응용