논문 00-25-10B-3 한국통신학회논문지 '00-10 Vol.25 No.10B

멀티캐스트 ATM망에서 ABT 블록스케줄링을 이용한 UBR 트래픽 성능개선에 관한 연구

정회원 임 동 규*, 박 용 진*

Study on Improvement of UBR Traffic Performance using ABT Block Scheduling in Multicast ATM Networks

Dong-kyu Lim*, Yong-jin Park* *Regular Members*

요 약

본 논문에서는 하나의 ATM 멀티캐스트(MVCC)세션 내에서, ATM계층 상위에 TCP와 같은 LAN 기반 망의 연동에 대해 다룬다. ATM 망에서 멀티캐스트 연결을 설정할 경우, 공유 트리 방식으로 연결 트리를 구성하고 AAL5 계층을 사용하므로 CIP(Cell Interleaving Problem)가 발생하게 된다. 이 문제를 해결하기 위해 ABT/IT(ABT with Immediate Transmission) RM(Resource Management) 셀을 이용한 블록 단위의 전송을 사용한다. 또한, ATM VC 위에 LAN 기반 망 연동서비스를 제공하기 위해 UBR 클래스를 이용하게 되는데, 그 속성상 QoS를 보장하지 못하고 망 혼잡발생 시에 우선적으로 폐기되며 단지 EPD(Early Packet Discard) 메커니즘과 같은 버퍼관리 방식만을 사용할 뿐이다. 따라서, 본 논문에서는 ABT RM 셀을 통한 블록 단위 전송을 사용하여 CIP를 해결할 뿐만 아니라, 여러 응용 트래픽 유형을 고려한 스케줄링을 통해 망에 혼잡이 발생할 경우에 UBR 트래픽이 무차별적으로 폐기되는 현상을 막고, 효율적이고 공평한 서비스를 제공한다. 특히 이를 위해 ATM 스위치 내에 하나의 멀티캐스트 세션에 대한 블록 스케줄링 알고리즘을 제안한다. 이를 적용하여 각 VC로부터 스위치에 들어오는 트래픽을 유형별로 분류하여 클래스 버퍼에 저장한 후, 유형에 따라 차별적인 전송을 해준다. 이 때, 블록 스케줄링 알고리즘을 적용하여 종단간 전송지연 및 손실률 등 UBR 트래픽의 성능이 향상되도록 하였다. 또한 C언어 및 자료구조를 이용한 모의실험을 통하여 블록 스케줄링 알고리즘의 성능을 평가하였다.

ABSTRACT

This paper treats the interworking of LAN-based networks like TCP over the ATM protocol stack, in an ATM multicast session. Multicast connection will cause CIP since multicast group members form a connection tree by some tree methods and share the connected tree. This paper solve the CIP problem through a block-by-block transmission using ABT/IT method. ABT/IT RM cell is modified, and block scheduling algorithm considering the traffic types is applied to each ATM switch using the enhanced RM cell. Block scheduling algorithm will avoid the indiscriminate discard of UBR traffic when congestion occurs, and it can provide an efficient and fair service. This paper builds a block scheduler system and suggests the block scheduling algorithm for a multicast session in an ATM switch. UBR traffics arriving at the switch through each VC is classified by the traffic type and stored at class buffer, and thereafter indiscriminately transmitted. When block scheduling algorithm is applied, it will improve the UBR traffic performance such as end-to-end delay, cell block loss ratio, etc. This paper evaluated the performance of block scheduling algorithm through the simulation using the C language and data structure.

* 한국전자통신연구원 상호운용성시험팀
** 한양대학교 전자공학과
논문번호 : 00100-0317, 접수일자 : 2000년 3월 17일

Ⅰ. 서 론

최근 망의 특징은 멀티미디어, 실시간 트래픽 등의 활성화로 인해 다양한 서비스 트래픽을 수용하는 광대역 통합서비스 망이라 할 수 있다. 뿐만 아니라, 영상회의, 가상현실 등 멀티미디어, 멀티캐스트 서비스 수요가 증가하면서 ATM 망에서는 멀티캐스트가 중요한 이슈가 되고 있다. 본 논문에서는 하나의 ATM 멀티캐스트 세션 내에서, ATM 계층 상위에 TCP와 같은 LAN 기반 망의 연동시 종단간 트래픽의 성능 향상에 대해 다룬다.

1. 관련연구 및 문제점

최근 ATM 망은 점대점 연결뿐만 아니라, 점대다점 연결의 특징을 나타내고 있다. 이와 같은 멀티캐스트 ATM에서 트리를 구성하는 방법으로 망사구조 방식, 서버 기반 방식 등 다양하지만, 단일한 VCI/VPI로 트리가 구성되므로 확장성이 높고, 트리 변경에 따른 시그널링 오버헤드도 훨씬 적고 효율적인 공유 트리 방식이 멀티캐스트 ATM에서 큰 장점을 갖는다.

그러나, 이렇게 공유 트리 방식에 의해 멀티캐스트 ATM 연결을 설정하게 될 경우, 여러 VC들이 스위치에서 하나의 VC로 병합되는 과정에서 몇가지 문제점이 제기된다. 그 중에서도 무엇보다 중요한 문제점은 CIP이다. 즉, AAL5 계층을 통해 전송할 경우, 각 소스로부터 셀들이 병합 및 혼합되어 어느 소스로부터의 트래픽인지 구분할 방법이 없기 때문에, 수신측에서 원래의 메시지로 재조립할 수 없게 되는 문제점이 발생된다[2].

CIP를 해결하기 위해, 여러가지 방식이 제안되었다[2]. 첫째, 앞서와 같이 멀티캐스트 서버를 통해 셀을 재조립한 후 다시 전송하는 방식이 있고, 둘째로 하나의 멀티캐스트 세션을 VP로 구별하고 세션 내의 각 연결을 VC로 구별하는 방식이 있다. 그 외에도 ATM 셀 헤더 내에 GFC(General Flow Control) 필드를 두어 각 연결을 구별하는 방식 등이 있다. 하지만, 그 중에서도 여러 셀들의 집합인 메시지 블록을 두어, 각 연결에 대해 블록 단위 전송을 하는 방법이 가장 적합한 방식으로 제안되고 있다[3].

그러나 블록 단위 전송방식 중 대표적인 SEAM은 각각의 트래픽 유형에 따른 차별화된 전송을 제공할 수 없고 CBR이나 rt-VBR 서비스 클래스의 QoS(CDV)를 제공하기에 부적합하다[3]. 따라서 엄격한 QoS를 요구하지 않는 ABR이나 UBR 서비스를 제공하게 되는데 이 또한 트래픽 성능의 향상을 기대하기는 어렵다.

특히 ABR 서비스의 경우, 전송지연의 증가 등 여러 이유로 인해 UBR을 통한 연동 서비스 보다 성능이 떨어진다[4]. 따라서, LAN 기반 망의 연동 서비스는 UBR 서비스 클래스를 통해 제공하며, UBR의 성능 향상을 위해 EPD라는 혼잡제어를 위한 버퍼 관리 메커니즘을 도입할 수 있다[1][5]. 그러나 이 방식만으로는 UBR 트래픽의 성능 향상에 한계가 있다. 특히 멀티미디어 트래픽을 UBR 클래스를 통해 전송할 경우, 각 트래픽의 유형에 따른 망 자원의 할당 등이 트래픽 성능 향상을 위한 중요한 이슈가 될 수 있다. 이를 위해 망 자원관리 방식중 하나로 큐 관리 메카니즘이 고려되어야 한다.

최근 들어 인터넷과 같은 LAN기반 망은 멀티미디어, 실시간 트래픽 등 다양한 형태의 서비스를 제공하고 있고, 이에 따라 각기 다른 트래픽 유형이 존재한다. 하지만, 공유 트리 방식으로 구성된 멀티캐스트 ATM의 경우에 공유 트리상의 모든 연결들이 하나의 VCI/VPI로 병합되어 전송되므로 동일한 멀티캐스트 세션내에 존재하는 여러 유형의 인터넷 트래픽이 ATM 레벨에서는 구별되지 않는다.

ATM 레벨에서 UBR 트래픽은 연결 설정시 어떠한 트래픽 변수 및 QoS 변수도 설정하지 않는다. 따라서 스위치에 혼잡이 발생하는 경우에, ATM 레벨에서는 CBR/VBR 트래픽 대신에 UBR 트래픽의 셀들을 우선적으로 폐기하는 등 UBR 트래픽의 전송 대역폭을 제한하는 방식으로 트래픽 관리를 할 것이다. 따라서 트래픽의 유형에 따라 스케쥴링을 통해 차별화하여 전송하지 않는다면, 실제 종단간의 응용 레벨에서는 전송 지연 및 셀 손실률 증가 등 서비스의 유형에 따라 매우 다른 트래픽 성능을 나타낼 것이고 결국, 트래픽 성능이 저하되는 문제가 발생한다. 따라서 전체적인 트래픽의 성능 향상을 위해서는 트래픽의 유형에 따른 스위치 내에서의 스케쥴링이 요구된다. 또한 CIP문제를 해결하면서 동시에 스케쥴링을 통한 UBR 트래픽의 성능 향상을 얻을 수 있는 새로운 방법이 필요하다.

2. 접근방법

각 서비스 클래스에 따른 QoS 보장을 통해 트래픽 성능을 향상시키기 위해서 라우터나 스위치와 같은 망의 중간 노드들은 여러 유형의 트래픽 관리

메카니즘을 제공하는데 그 중 중요한 기능이 효율적인 큐 스케줄링 기능이다. 큐 관리 알고리즘으로는 도착한 순서에 따라 처리하는 FIFO큐 알고리즘, 우선순위가 높은 큐부터 서비스하는 우선순위 큐 알고리즘, 각 큐에 대역폭을 할당하여 라운드 로빈 방식으로 서비스하는 클래스 기반 큐 알고리즘 등이 있다[8]. 하지만 이상적인 GPS (Generalized Processor Sharing) 큐 모델에 가장 근사한 WFQ (Weighted Fair Queuing) 알고리즘[9]이 가장 개선된 알고리즘으로 제안되고 있어 본 논문에서는 스케줄링을 위한 큐 관리 알고리즘으로 이를 사용한다.

또한 CIP를 해결하면서 동시에 스케줄링을 통해 UBR 트래픽의 성능 향상을 얻을 수 있는 새로운 방법이 필요하다. 그래서 제안되는 것이 블록 단위 전송인 ABT(ATM Block Transfer) 방식이다. ITU-T는 ATM에서 보내고자 하는 트래픽의 성능 향상 등의 트래픽 관리를 위해 망 자원을 예약하는 방법으로 ABT 방식을 제공하고 있다[6],[7].

ABT는 보내고자 하는 셀들을 블록 단위로 묶고, 각 블록의 앞뒤에 RM 셀을 두어 수락 제어를 하고, 망으로부터 허가된 블록에 대하여 망 자원을 예약한 후, 보장된 QoS를 통해 전송하는 방식이다. ABT에는 두가지 동작모드가 존재하는데 본 논문에서 접근하는 방법으로 전송 지연이 상대적으로 낮은 ABT/IT 를 사용하여 블록 단위 전송을 한다. 이와같이 ABT의 속성인 RM 셀을 통한 블록 단위 전송을 이용할 경우, 첫째, CIP를 해결하고, 둘째, 각 종단간 연결 트래픽의 유효 처리율을 향상시키며, 셋째, 소스 트래픽 서비스 유형에 따라 클래스로 분류하고, 각 스위치에서 해당 클래스에 기반하여 블록 스케줄링을 수행함으로써, 서비스 유형별로 종단간 전송지연이나 트래픽 손실률 등 트래픽 성능을 향상시킬 수 있다.

따라서 본 논문에서는 이를 위해 ABT 전송시 사용되는 ABT RM 셀의 예비영역에 '송수신측 주소'와 'Classification ID' 필드를 추가하는 등 RM셀을 수정·보완한 후 이를 이용해 망 상의 각 스위치에 블록 스케줄링 알고리즘을 제안하고 적용한다. 또한 이를 구현하기 위해 ATM 스위치에서 블록 스케줄러 시스템을 구성하고 블록 스케줄러에 적용할 블록 스케줄링 알고리즘을 설계하였다. 더불어 블록 스케줄링 알고리즘을 적용할 경우 각 트래픽의 유형별로 서로 다른 종단간 전송 지연 및 트래픽의 성능을 비교·평가 하였다.

Ⅱ. ATM 스위치에서 블록 스케줄링 알고리즘 설계

1. 망 모델 구성도

멀티캐스트 ATM 스위치에서의 블록 스케줄링 알고리즘을 적용하기 위한 망의 모델을 그림 1에 나타내었다. 이 ATM 망에는 여러 CBR, VBR 연결들과 함께 하나의 멀티캐스트 세션 연결이 송신측과 수신측으로 구분, 설정되어 있다.

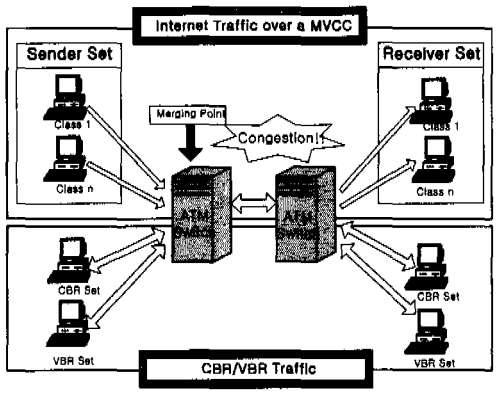


그림 1. 멀티캐스트 ATM연결을 포함한 망 모델 구성도

각각의 ATM 스위치는 서비스 클래스의 우선 순위에 따라 CBR과 VBR 연결들을 먼저 서비스해주고, 여분의 대역폭에 한해 멀티캐스트 세션을 통해 설정된 LAN 기반망 연결을 서비스해주게 된다. 이와 같은 망 모델로부터 망에 혼잡이 발생하는 경우에 주목하여, 멀티캐스트 세션 내 UBR 트래픽의 효과적인 전송을 위한 알고리즘을 설계하고자 한다.

2. MVCC 세션에서의 트래픽 관리

ATM 스위치 내에서는 여러 CBR 및 VBR 트래픽과 멀티캐스트 연결인 MVCC(Multicast Virtual Channel Connection) 세션이 함께 통과한다. 일반적인 망 상황에서는 모든 트래픽이 원활하게 서비스되지만, 혼잡이 발생할 경우에는 UBR 트래픽인 MVCC 세션에서 서비스에 제한을 받게 된다. 따라서, 제한된 전송 대역폭을 최대한 효율적으로 활용하여 트래픽의 성능을 향상하도록 하는 메커니즘이 필요하다. 하지만, MVCC 세션의 UBR 트래픽은 앞서 살펴본 바와 같이 서비스 품질(QoS)을 보장하기 위한 별도의 트래픽 관리 메커니즘을 제공하지 않는다. 다만, 각각의 ATM 스위치는 UBR 트래픽

에 대해 EPD 방식이라는 버퍼관리 메커니즘을 사용하며, 이를 위해 VC별로 버퍼링 한다.

이러한 상황에서 큐 관리 기법의 제공이 트래픽 성능 향상을 위한 좋은 메커니즘이 될 수 있다. MVCC 세션 내에는 여러 유형의 트래픽이 서비스 된다. MVCC 세션 내 여러 트래픽들을 각각의 속성에 따라 클래스로 분류하고 서비스 가중치를 부여한다. 이를 위해 ABT/IT의 RM 셀 내에 <Classification ID> 필드를 둔다. 각각의 ATM 스위치에서 MVCC 세션내의 각 트래픽을 서비스 가중치에 따라 스케줄링 및 서비스함으로써 실질적인 트래픽 성능의 효율을 높일 수 있다.

3. 블록 스케줄러 구성도

망에서 각 연결들은 서로 다른 서비스 요구 가중치를 갖는다. 공평성은 망 대역폭의 사용이 혼잡한 경우 각 연결에 할당되어 있는 대역폭이 최대한 보장되며, 대역폭의 사용이 혼잡하지 않은 경우 여분의 대역폭을 현재 전송중인 연결들에 공평하게 분배되는 것으로 정의한다[10]. 따라서 ATM 스위치는 실시간 및 비실시간 서비스 제공을 위해 데이터를 전송중인 모든 연결의 최소한의 전송률과 공평성을 보장해야 한다.

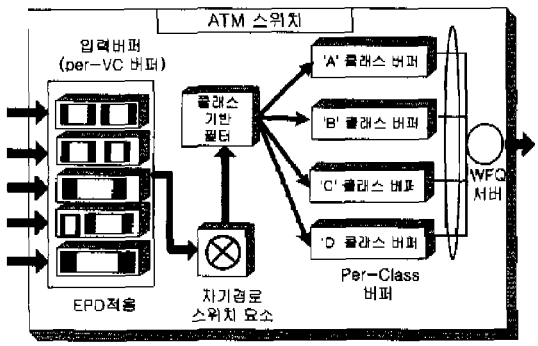


그림 2. ATM 스위치에서의 블록 스케줄러

본 논문은 ATM망을 통해 전송되는 CBR, VBR과 같은 실시간 연결들에 대한 스위치에서의 큐 관리 및 스케줄링에 관해서는 다루지 않는다. 실시간 트래픽들은 자기경로 스위칭 요소를 통과한 후에 별도의 스케줄링 없이 출력 포트를 통하여 전송될 것이다. 따라서, 본 논문에서는 MVCC 세션을 통해 전송되는 비실시간 트래픽들에 대한 큐 관리, 즉 블록 스케줄링에 대해서 만 다룬다. 다만, ATM 스위치는 블록 스케줄링을 통과한 MVCC 세션의 비실시간 트래픽과 그 외의 실시간 트래픽들을 출력 포트를 통해 전송하는 과정에서, 실시간 트래픽에 의해 비실시간 트래픽의 전송률 보장이 무시되지 않도록 해야 한다[11]. 멀티캐스트 ATM 망에서 블록 스케줄링 알고리즘을 위한 시스템을 ATM 스위치에 구현하였다(그림 2).

MVCC 세션 내 UBR 트래픽의 블록 단위 전송 시, ATM 스위치 내에서 블록 스케줄링 알고리즘을 적용하기 위한 시스템을 '블록 스케줄러'라고 정의한다. 블록 스케줄러의 구성 요소들을 살펴보면, 기본적인 ATM 스위치에 '클래스 기반 필터'와 '클래스 버퍼', 그리고 'WFG 서버'가 새롭게 추가 되었다. 또한, 입력 포트를 통해 임의대로 들어오는 셀 블록 들에 대해 각 VC별로 버퍼를 두고 EPD 버퍼 관리 방식을 적용할 수 있도록 입력단에 'per-VC 버퍼'를 두었다. 입력버퍼를 통해 임의대로 들어온 셀블록들은 EPD 메커니즘에 의해 전송 내지 폐기 여부를 결정한다. 그런 후에는 자기경로 스위치 요소에 의해 출력링크를 결정한 다음, 클래스 기반 필터를 통하여 트래픽 유형에 맞게 해당 클래스 버퍼에 입력된다. 클래스 버퍼에 입력된 셀 블록들은 WFQ 서버에 의하여 스케줄링되어 미리 정해진 출력 링크를 통해 전송되어 나가게 된다. 클래스 기반 필터는 MVCC를 통해 전송되는 ATM 셀 블록들에 대해서만 적용되는 구성 요소이다. MVCC와 같은 출력 링크로 전송되는 그 밖의 셀 트래픽들은 클래스 기반 필터를 거치지 않는다. WFQ 서버는, ① CBR, VBR, ABR 트래픽에 대해 우선적으로 보장된 서비스를 제공해 주고, ② 여분의 대역폭에 대해서 WFQ 알고리즘을 적용, 스케줄링하여 각 클래스 기반 버퍼의 ATM 셀 블록들에게 향상된 서비스를 제공한다.

4. 큐 알고리즘 설계

멀티캐스트 ATM 스위치에 대해 MVCC 세션의 UBR 트래픽을 위한 블록 스케줄링 알고리즘을 설계하는 데 있어서 고려해야 할 사건은, 셀블록의 도착과 큐 입력을 나타내는 <Arrival (enqueuing)> 사건, 그리고 트래픽의 유형과 가중치에 따라 서비스 순서를 정해주는 'dequeuing'으로 이루어지는 <Departure> 사건이다. 본 논문에서는 블록 스케줄러는 큐 입력을 수행하면서, 동시에 스케줄링 및 큐 출력을 수행한다.

4.1 알고리즘에 사용되는 개체 및 변수

블록 스케줄링 알고리즘은 각 큐(Queue)와 각 셀

블록(cell_block)의 두 개체, 그리고 그외 프로그램 전역 변수로 구성된다. 각 개체가 포함하는 변수를 살펴보자.

먼저, '큐'의 개체를 보면 다음과 같다. 큐에서는 셀블록이 도착하면 셀블록을 받아들이는 부분에서 EPD 방식에 의해 버퍼 관리가 이루어지고,

표 1. 큐 알고리즘 설계를 위한 구조체 및 변수

개체	개체가 포함하는 변수	
큐 구조체(Queue)	버퍼관리	Buffer size, current size, threshold
	스케줄링	Weight, head of line, prev finish time
	성능평가 요소	Total_cb_delay, total_cb_ arrived, total cb discarded, total cb served
셀블록 구조체 (cell_block)	스케줄링	Class type, finish time
	성능평가 요소	Arrival_time, departure_ time
그외 프로그램 전역변수	System_finish_time, link_capacity Total buffer size, arrival time interval	

폐기되지 않은 셀블록에 대해서 큐에 입력한다. 이 때, 큐의 상태가 갱신되며 입력된 셀블록에 대해 GPS 서비스 종료시간을 구한다. 또한, 셀블록이 출력 링크를 통해 전송되기 위해 WFQ 알고리즘을 이용한 큐 스케줄링이 이루어진다. ATM 스위치는 큐 스케줄링에 의해 선택된 셀블록을 해당 큐로부터 출력하고 그에 따라 큐의 상태를 갱신한다.

'EPD 방식에 의한 버퍼 관리'를 위해 큐의 전체 크기, 즉 수용 가능한 셀블록의 수를 알아야 하고(buffer_size), 네개 큐의 현재 용량(current_size)의 총합이 전체 큐의 용량(total_buffer_size)을 넘어서는지 체크한다. 만약 그 값을 넘어설 경우에는, 현재 용량이 미리 정해진 특정 한계값(threshold)을 초과하는 큐에 대해 들어오는 셀블록 전체를 폐기시킨다(total_cb_discarded). 또한, 큐가 입력될 때 'GPS 서비스 종료시간 계산' 및 '큐 갱신'이 이루어진다. 이를 위해 각 큐는 트래픽 유형별로 미리 정해진 서비스 가중치(Weight)를 갖고, 그 가중치와 이전 셀블록의 GPS 서비스 종료시간(prev_finish_ time), 그리고 출력 대역폭(link_capacity)을 이용해 현재 입력되는 셀블록의 GPS 서비스 종료시간을 구한다. 그런 후에 큐의 갱신이 이루어진다 (current_size, total_cb_arrived).

이제 출력링크로 전송되는 셀블록에 대해 '큐 스케줄링'이 이루어지는데, 비어있지 않은 큐의 맨 앞 셀블록(head_of_line)들에 대해 GPS 서비스 종료시간을 비교해 가장 작은 값을 갖는 셀블록을 해당 큐로부터 삭제, 출력한다. 그런 후에 큐의 상태가 갱신되고(current_size 감소), 성능평가를 위한 큐의 전송지연(total_cb_delay)과 전송된 셀블록의 수(total_cb_ served)도 누적, 갱신된다.

다음으로, '셀블록(Cell_block)'의 개체를 살펴보면, MVCC 세션을 통해 스위치로 진입해 들어오는 셀블록의 트래픽 유형(class_type)을 구별하여, 셀블록이 해당 큐에 입력될 수 있도록 한다. 해당 큐에 입력될 때 위에서 말한 여러 변수에 의해 GPS 서비스 종료시간을 계산하여 그 값을 저장한다 (finish_time). 해당 셀블록이 큐의 맨 앞에 위치하면, 이 값의 크기를 다른 큐의 맨 앞 셀블록들과 비교하여 서비스 순서를 정하게 될 것이다.

위에서 말한 두 개의 개체와는 별도로, 블록 스케줄링 알고리즘 전체에 적용되는 전역 변수가 필요하다. 앞서 언급한 전체의 큐의 용량(total_buffer_ size)외에도, WFQ 서버에서 각 큐의 맨 앞에 위치한 셀블록의 GPS 서비스 종료시간을 비교하여 가장 작은 값을 계속 갱신하는 변수(system_finish_ time)가 필요하다. 또한, UBR 트래픽이기 때문에 그 속성상 CBR, VBR을 서비스하고 남은 여분의 대역폭을 사용하므로, 시간에 따라 변하는 출력 링크 용량(link_capacity)을 알고 있어야 한다. 위의 두 변수는 각 큐에 진입하는 셀 블록들의 GPS 서비스 종료시간을 구하는 데 이용되는데, 자세한 것은 알고리즘을 통해 살펴보겠다.

4.2 블록 스케줄링 알고리즘

```
큐('Queue', q[0]~q[3]) 초기화
FOR i=0; i<=3; ++i
  q[i].current_size = 0
  q[i].threshold = 0.8 * q[i].buffer_size
  total_buffer_size += q[i].buffer_size
  q[i].preve_finish_time = 0
  q[i].weight = 0.4, 0.3, 0.2, 0.1 (또는 그외)
  (단위 : cell blocks/sec)
  q[i].total_cb_delay = 0
  q[i].arrived/discarded/served = 0
```

그림 3. 블록 스케줄링 알고리즘(큐 초기화)

먼저, 블록 스케줄러를 초기화 한다. 우선 각 셀블록을 트래픽 유형에 따라 'A'~'D'로 분류한다.

분류된 셀블록들을 저장하는 각 큐는 '버퍼 관리'와 '스케줄링', 그리고 '성능평가 요소'에 해당하는 변수들을 모두 초기화 한다. 각 큐는 저장하는 셀블록들의 트래픽 유형에 따라 서로 다른 서비스 대역폭 할당 가중치를 부여한다. 블록 스케줄러의 초기화는 그림 3과 같이 이루어진다.

이제, 하나의 셀블록이 ATM 스위치로 들어오면 트래픽 유형에 따라 셀블록을 분류하고 해당 큐로 입력시킨다. 이 때, EPD 버퍼 관리 방식을 적용하여 버퍼 용량을 초과해 들어오는 셀블록을 폐기하며, 큐 내에 폐기된 셀블록의 수를 증가시킨다. 이와 함께, 큐의 현재 버퍼 용량 등 큐의 상태를 갱신한다. 셀블록이 도착하여 큐에 입력될 때, GPS 서비스 종료 시간을 계산하여 해당 변수(finish_time)에 저장한다. GPS 서비스 종료 시간을 구하는 알고리즘은 그림 4와 같다.

```
Fin_time 함수
IF there is PREVIOUS_HOL in a q[i]
  Cblock.finish_time = q[i].prev_finish_time +
  1/(t)*q[i].weight
ELSE there is NO PREVIOUS_HOL in the q[i]
  Cblock.finish_time = system_finish_time +
  1/(t)*q[i].weight
```

그림 4. 블록 스케줄링 알고리즘(GPS 서비스 종료 시간 계산)

만약, 큐 내에 여러 셀 블록들이 잇달아 들어오는 경우에는, 이전 셀블록의 종료시간을 이용해 GPS 서비스 종료시간을 구한다.

```
WFQ 스케줄링
target_queue = -1 //Comparison of GPS finish time
FOR i=0; i<=3; ++i
    IF q[i].current_size NOT 0
    IF first comparison
    taret_queue = i
    F_temp = q[i].head_of_line.finish_time
    ELSE
    IF q[i].head_of_line.finish_time LT F_temp
    target_queue = i
    F_temp = q[i].head_of_line.finish_time
IF target_queue GE 0 // Dequeue
     system_finish_time_time = F_temp
     Dequeue 절차
```

그림 5. 블록 스케줄링 알고리즘(WFQ 스케줄링)

하지만, 하나의 셀블록이 큐에 들어올 때 큐가 비어있는 경우에는 이전 셀블록의 종료시간이 갱신되지 않았기 때문에, 최근 서버에 의해 전송이 이루어진 셀블록의 서비스 종료시간을 이용하여 GPS 서비스 종료시간을 구하게 된다.

WFQ 서버는 각 큐의 맨 앞 셀블록들에 대해 서비스하기 전에, 각 셀블록의 GPS 서비스 종료시간을 비교한다. WFQ 서버는 그 중에서 가장 작은 값을 갖는 셀블록이 들어 있는 큐를 선택한 후에, 그 셀블록을 큐에서 삭제, 출력한다. 이 때, 전송된 셀블록의 GPS 서비스 종료시간을 해당 전역 변수(System_finish_time)에 저장, 갱신한다. GPS 서비스 종료시간을 비교하여 서비스할 큐를 선택하는 알고리즘을 그림 5에 나타내었다.

4.3 블록 스케줄링 알고리즘 설계모델

본 논문에서는 UBR 트래픽에 할당된 출력 링크의 대역폭이 시간에 따라 변하기 때문에 WF^2Q 알고리즘 대신에 WFQ 알고리즘을 사용한다. 그 이유는 GPS 서비스 시간(Virtual time)은 출력링크 대역폭과 각 트래픽 유형에 따른 서비스 가중치의 함수이므로, 출력 링크의 대역폭이 변할 경우 GPS 서비스 종료시간 뿐만 아니라 GPS 서비스 시작시간도 계속 불규칙하게 변하게 된다. 따라서, WF^2Q 알고리즘을 사용하게 될 경우, GPS 서비스 시작시간의 변동으로 인해 처리속도에 큰 부하가 발생하는 문제가 있다. 하지만, WFQ 알고리즘을 사용하게 되면, GPS 서비스 종료시간만을 고려하므로 시간에 따른 대역폭의 변화가 스케줄링에 영향을 미치지 않는다. 그것은 GPS 서비스 종료시간은 모든 클래스 큐들이 다 같은 비율로 증가 또는 감소하므로, 출력 링크의 대역폭의 변화가 스케줄링에 영향을 미치지 않기 때문이다.

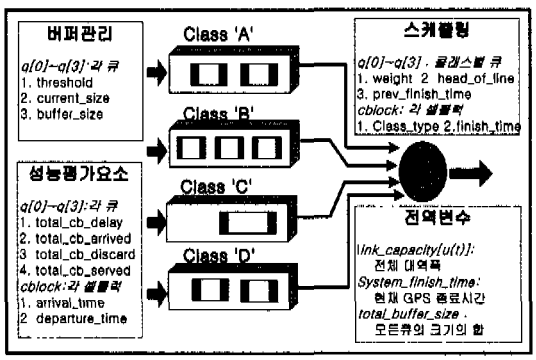


그림 6. 블록 스케줄링 알고리즘 설계모델

따라서, 출력 링크의 대역폭의 변화와 관계없이 가상시간, 곧 GPS 서비스 종료시간을 구하고 이를

비교하여 스케줄링을 하는게 더 나을 것이다. 요약하면, 출력링크의 대역폭의 변화로 인해 WF^2Q 알고리즘은 처리 속도에 부하가 발생하므로, WFQ 알고리즘을 사용하는 것이 바람직하다.

지금까지 살펴본 블록 스케줄링 알고리즘 설계 모델을 그림 6에 나타내었다.

Ⅲ. 성능 평가

ATM 네트워크에서 MVCC 세션 내에 여러 유형의 UBR 트래픽을 통해 응용 서비스를 제공할 때, 블록 스케줄링 알고리즘을 적용하게 되면 종단간 트래픽의 성능 향상을 기대할 수 있다. 즉, 블록 스케줄링 알고리즘을 사용함으로써 네트워크에 혼잡이 발생할 경우에 UBR 트래픽이 무차별적으로 폐기되는 것을 막는다. 또한, 트래픽의 특성에 맞게 차별적으로 폐기하거나 우선적으로 서비스해줌으로써 종단간 전송 지연이나 손실률 등의 측면에서 성능향상을 얻을 수 있다고 가정한다. C 언어를 이용한 모의실험(simulation)을 통해서 이를 검증하도록 한다.

1. 평가 방법

모의실험을 위한 단순화한 모델은 그림 7과 같다.

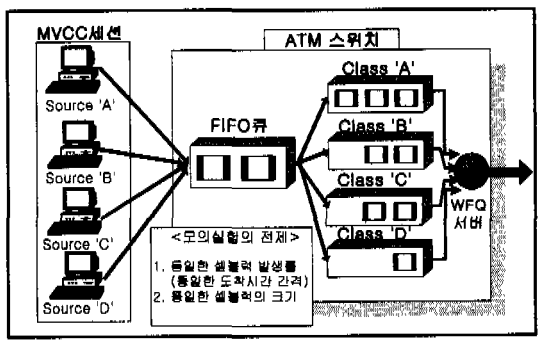


그림 7. 모의실험을 위한 단순화한 모델

위 모델에서 MVCC 세션은 네 개의 송신측으로 구성되고, 각 송신측은 각각 'A'~'D' 유형의 트래픽만을 전송한다고 가정한다. 이 모의실험에서는 성능을 효과적으로 평가하기 위하여 다음 몇 가지 사실을 전제로 하였다. 첫째, 각 큐의 모든 셀블록의 크기는 동일하다. 일반적으로 각 트래픽 유형마다 셀블록의 크기도 다르지만, 각 트래픽 유형에 따른 차별을 서비스 가중치(weight)에 의해서만 구현하였다. 둘째, 각 큐의 셀블록 도착비율, 즉 도착시간 간격의 분포는 동일하다. 이를 위해 <Arrival> 사건을 발생시킬 때, 셀블록의 트래픽 유형을 균일 분포(uniform distribution)에 의해 생성토록 하였다. 셋째, 종단간 트래픽의 성능 평가를 큐 내에서의 성능 평가로 제한하였다. 그것은 종단간 전송 지연이 큐 전송 지연에 의해 좌우되고, 셀블록의 손실도 거의 대부분이 네트워크에 혼잡 발생시 큐에서 이루어지기 때문이다.

표 2. 모의실험 시 트래픽의 유형 분류

우선순위	트래픽의 유형	코드에서의 표현
1	실시간, 신뢰성 있는 트래픽 (Real-time, Reliable)	'A'
2	실시간, 신뢰성 없는 트래픽 (Real-time, Unreliable)	'B'
3	비실시간, 신뢰성 있는 트래픽 (Non-realtime, Reliable)	'C'
4	비실시간, 신뢰성 없는 트래픽 (Non-realtime, Unreliable)	'D'

멀티캐스트 세션 내 각 송신측이 발생하는 트래픽 유형은 표 2와 같다. 그리고 각 트래픽 별로 다음의 변수에 대해 성능을 평가 한다.

① 종단간 전송 지연 (end-to-end traffic delay)

② 종단간 손실률 (end-to-end cell block loss ratio)

본 논문에서는 C 언어의 사건유도 모의실험(event-driven simulation) 방식을 이용해 성능평가를 하고자 한다. 프로그래밍을 위한 블록 다이어그램은 그림 8과 같다. 이 모의실험에서는 두 가지의 사건이 동시에 일어나는데, 앞서 말한바와 같이 셀블록의 도착 및 입력을 나타내는 <Arrival> 사건, 그리고 GPS 서비스 종료시간을 비교하여 서비스할 큐를 선택하는 'scheduling'과 큐의 셀블록을 서비스하여 큐로부터 출력시키는 'dequeuing'으로 이루어진 <Departure>사건으로 구성된다.

모의실험 과정은 발생할 다음 사건을 찾고, 그 사건을 적용할 경우 버퍼에 생기는 변동을 반영하도록 큐를 갱신하고, 이로 인해 새로이 발생하는 다음 사건을 찾는 방식으로 이루어진다. 각 사건을 추적하기 위해 프로그램은 '사건 리스트'라고 부르는 올림차순 우선순위 큐를 사용한다. 이 리스트는 최대 2개의 사건 노드를 가지며, 각 노드는 2개 유형의 사건 중에서 다음에 발생할 사건을 표현한다. 사건 리스트는 사건 발생의 시간이 증가하는 순서로 정렬되어 있어, 모의실험의 각 순간마다 발생할 다음

사건을 알고 있어야 한다.

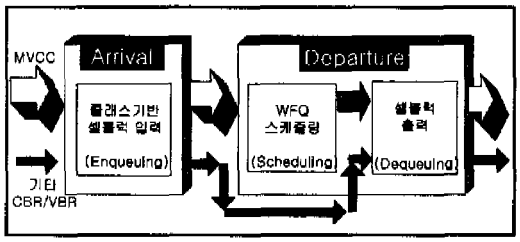


그림 8. 모의실험을 위한 블록 다이어그램

좀더 자세히 각 사건을 살펴보자. 맨 처음 셀블록의 도착으로 <Arrival>사건이 발생하면, <Arrival> 사건노드가 사건 리스트에 저장되는데, 이 사건 노드를 사건 리스트에서 삭제한 후에 성능 평가를 위한 요소들을 갱신하게 된다. 즉, 출발 시간과 도착 시간의 차를 구해 큐 전송 지연 변수에 누적시키고, 서비스된 셀블록의 수를 나타내는 변수를 누적, 갱신한다. 이제, <Departure> 사건 노드는 각 큐를 검색하여 비어있지 않은 큐가 하나라도 있는 경우에, 다시 <Departure> 사건을 발생시켜 해당 사건 노드를 사건 리스트에 저장하게 된다.

이 과정이 계속되어 마지막에는 사건 리스트가 비워지게 된다. 그럴 경우, 모의 실험이 종료되면서 평균 큐 전송 지연과 셀블록 손실률을 구한다. 이를 표로 나타내면 표 3과 같다.

표 3. 모의실험을 통한 성능 평가 요소

각 큐의 평균 전송지연	Σ(각 셀블록의 큐 전송지연)/(서비스된 셀블록 수) = Σ((출발시간-도착시간)+서비스시간)/(서비스된 셀블록 수)
각 큐의 셀블록 손실률	(폐기된 셀블록 수)/(전송된 셀 블록 수)

본 논문에서는 각 큐의 조건이 동일한 상황하에서, 위와 같이 구한 각 큐의 두 성능평가 요소가 서로 차별적임을 보이고자 한다. 또한, 각 큐의 버퍼 크기, 셀블록들의 도착 시간 간격 또는 서비스 시간 간격 등을 조작하면서, 각 큐별로 위 두 성능평가 항목의 추이를 파악할 수 있을 것이다.

2. 평가 결과

성능 평가를 위해 미리 정한 모의 실험 환경은 표 4와 같다. EPD 메커니즘에 의한 버퍼 관리를 위해 각 큐의 특정 한계값을 큐 버퍼용량의 80%로 정한다. 또한, 블록 스케줄링을 위해 각 큐의 서비스 가중치는 임의대로 0.4, 0.3, 0.2, 0.1로 정하여 트래픽 유형에 따른 차별화된 성능을 나타냄을 검증하고자 하였다.

표 4. 모의실험 환경

항목	변수명	값
셀블록의 도착시간 간격	arrival_time_interval	0.001 (secs/cell_block)
출력 링크 대역폭	link_capacity	100 (cell_blocks/sec)
각 큐의 특정 한계값	q[I].threshold	80% q[i].buffer_size
각 큐의 서비스 가중치	q[0].weight	0.4
	q[1].weight	0.3
	q[2].weight	0.2
	q[3].weight	0.1

주목할 것은 버퍼의 크기에 따라 종단간 전송지연, 셀블록 손실률의 성능이 전체적으로 어떤 양상을 띄는지를 보였다. 먼저, 트래픽 유형별로 종단간 전송지연을 구한 결과는 그림 9와 같았다.

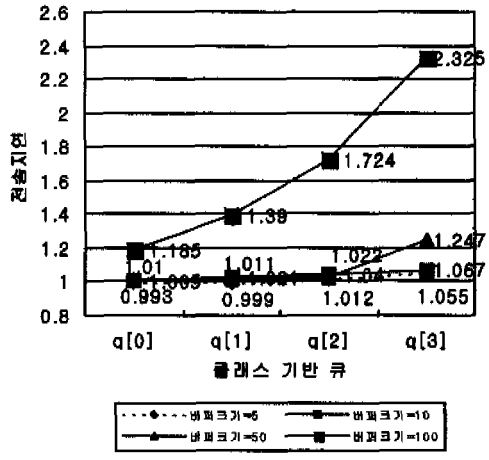


그림 9. 버퍼크기에 따른 트래픽 유형별 셀블록 전송 지연

그림에 따르면, 트래픽의 중요도 및 우선순위가 높아 서비스 가중치가 큰 'A' 유형의 트래픽에 비해, 서비스 가중치가 작은 'D'로 갈수록 전송 지연이 증가함을 알 수 있다.

특히, 'D' 유형으로 갈수록 전송 지연 증가율이 급증하게 된다. 이와 같이, 트래픽 유형 별로 전송 지연 성능을 차별적으로 제공해 줌으로써, 실시간

서비스와 같은 높은 우선순위의 트래픽의 성능을 크게 향상 시킬 수 있다. 유의할 점은 버퍼 크기가 증가할 경우(버퍼 크기=10), 전송 지연이 낮아져 트래픽의 성능이 향상되지만, 버퍼크기가 계속 증가할 경우에는 다시 전송지연이 높아짐을 알 수 있다. 이것은 망 혼잡 발생시, 버퍼의 용량을 무조건 증가시키는 것이 혼잡을 해결할 수 없다는 사실을 말해준다.

다음으로, 트래픽 유형별로 셀블록 손실률을 구하였다(그림 10 참조). 셀블록 손실률은 전송 지연과 같이 큰 차이를 보이지는 않지만, 전체적으로 'A' 유형의 트래픽이 낮은 셀블록 손실률을 보임을 알 수 있다. 유의할 점은 버퍼 크기가 증가할수록, 전체적인 셀블록 손실률이 매우 낮아진다는 사실이다. 실제로 버퍼 크기를 100으로 하였을 때, 모든 큐의 셀블록 손실률이 '0'을 나타내었다.

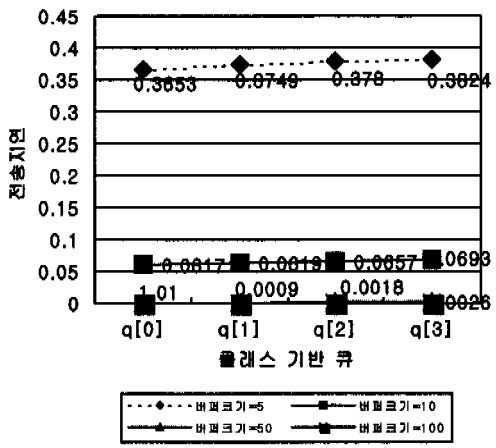


그림 10. 버퍼크기에 따른 트래픽 유형별 셀블록 손실률

Ⅳ. 결 론

ATM 멀티캐스트 세션(MVCC)을 통해 LAN 기반 망의 연동 서비스를 제공할 경우에, CIP 문제를 해결하기 위해 블록 단위 전송을 도입하게 되고, 그 특성상 UBR 서비스 클래스를 통해 전송해야 한다. 하지만, UBR 서비스는 그 속성상 best-effort 트래픽이므로 보장된 서비스 품질을 제공할 수 없기 때문에, MVCC 위의 각 소스 트래픽의 서비스 유형이 다름에도 불구하고, 무차별적으로 폐기됨으로 인해 전체적인 트래픽의 성능 및 공평도가 저하되는 문제점을 드러내게 된다.

본 논문에서는 UBR 클래스를 통해 서비스를 제공하면서 최대한 전체 트래픽의 성능을 향상 시키기 위해 제안된 메커니즘이 블록 스케쥴링 알고리즘이다. 블록 스케쥴링 알고리즘은 ABT 셀블록의 RM 셀을 이용하여 RM 셀내에 'Classification ID' 필드를 두어 각 소스 트래픽의 유형을 분류하고, 그 우선 순위에 맞게 링크 대역폭을 할당하여 전송하는 방식으로 WFQ 알고리즘을 적용한다.

이를 통하여, 종단간 트래픽의 전송 지연이나 데이터 손실률이 충분히 개선됨을 알 수 있을 것이다. 또한, 각 트래픽들의 출력 링크 대역폭을 사용하는데 있어서 공평도가 높아짐을 알 수 있을 것이다.
큐 알고리즘을 통한 망 자원 관리는 혼잡제어 등과 함께 망 트래픽 관리 메커니즘 중의 하나이다. 특히, UBR과 같이 할당된 대역폭의 사용을 보장하는 트래픽 관리 메커니즘이 전혀 없는 상황에서는, 스위치 내에서의 스케쥴링 메커니즘이 중요한 트래픽 관리 요소로 사용될 수 있다. 현재, 많은 큐 알고리즘이 연구되고 있는데, 향후 그 알고리즘들이 MVCC 위에서의 LAN 기반 연동 서비스 제공에 적절히 응용된다면, 전송 지연이나 손실률, 공평도 등 전체 트래픽의 성능을 크게 향상 시킬 수 있을 것이다.

참 고 문 헌

[1] William Stallings, High-Speed Networks: TCP/IP and ATM Design Principles, Prentice-Hall, 1998

[2] E. Gauthier, J.-Y. Le Boudec, SMART: A Many-to-Many Multicast Protocol for ATM, *IEEE* Journal on Selected Areas in Communications, IQ 1997

[3] M. Grossglauser, K. Ramarkrishnan, SEAM: Scalable and Efficient ATM Multicast, *IEEE* INFOCOM '97, Kobe, Japan, April 1997

[4] Sam Manthorpe and J-Y Le Boudec, A comparison of ABR and UBR to support TCP traffic, DI Technical Report 97/224, April 1997

[5] A. Romanow and S. Floyd, Dynamics of TCP Traffic over ATM Networks, Proc. SIGCOMM '94, August 1994

[6] ATM Forum, Traffic Management Specification Version 4.0, May 1996

[7] T. M. Chen, S. Liu, V. Samalam, The ABR

Service for Data in ATM Networks, IEEE Comm. Mag., May 1996, pp. 56-71

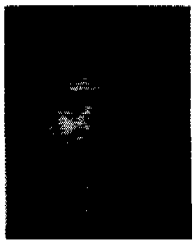
[8] Cisco Systems, Queue Management, White Paper, 1996

[9] C. Bennett and H. Zhang, Why WFQ Is Not Good Enough for Integrated Services Networks?, Proceedings of NOSSDAV'96, Apr, 1996

[10] Raj Jain, Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey, Comp. Networks and ISDN Sys., Oct. 1996

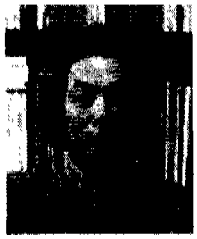
[11] Uwe Briem et al, Traffic Management for an ATM Switch with Per-VC Queuing: Concept and Implementation, *IEEE* Comm. Mag., June 1988

임 동 규(Dong-kyu Lim) 정회원



1988년 : 한양대학교 대학원 졸업(공학석사)
1990년~현재 : 한국전자통신연구원 상호운용성시험팀

박 용 진(Yong-jin Park)



1978년 3월 : 일본 와세다 대학교 졸업(공학박사)
1999년 9월~현재 : APAN 사무국장
1999년 1월~현재 : IEEE Seoul 지부 회장
1998년 11월~현재 : APAN-Korea 컨소시엄 의장
1997년 6월~1997년 9월 : 일본 와세다 연구센터 객원 연구원
1997년 1월~현재 : IEICE 자문위원
1996년 1월~1996년 12월 : 한국정보과학회 부회장
1994년~1995년 : 개방형컴퓨터통신연구회 회장
1991년~1992년 : 영국 켄트 대학교 객원교수
1983년~1984년 : 미국 일리노이 대학교 (Urbana-Champaign) 객원 교수
1978년~현재 : 한양대학교 전자공학과 교수