Integrated Service 모델을 응용한 웹 기반 QoS 보장형 트래픽 제어시스템

정회원 이 명 섭*, 박 창 현*

A Web-based and QoS-guaranteed Traffic Control System using Integrated Service Model

Myung-Sub Lee*, Chang-Hyeon Park* ***Regular Members***

요 약

최근 인터넷 기술이 급속도로 발전하고 인터넷상에서의 멀티미디어 통신이 일반화되면서 네트워크 트래픽이 급속히 증가되고 있으며, 이러한 트래픽 급증으로 인하여 네트워크 회선 부족 및 통신 서비스의 품질 저하 등의 문제점들이 발생하고 있다. 본 논문에서는 이러한 문제점을 해결하기 위해서 멀티미디어 통신 등과 같은 실시간 패킷의 전송에서 품질 보장을 지원하는 웹 기반 트래픽 제어 시스템을 제시한다. 본 논문에서 제시하는 트래픽 제어 시스템은 IntServ 모델을 응용하여, 망 운영자의 정책 및 망 종단간의 트래픽 부하를 고려한 최적의 자원 할당에 따라 패킷의 전송률을 결정하여 전송 품질을 보장하고, 또한 실시간 패킷의 경우, 인증 제어 과정과 비동기 큐잉 및 클래스 기반 큐잉을 지원하도록 수정된 WF^2Q^+ 스케줄러를 통하여 전송품질을 보장한다.

ABSTRACT

As the recent rapid development of internet technology and the wide spread of multimedia communications, massive increase of network traffic causes some problems such as the lack of network paths and the bad quality of service. To resolve these problems, this paper presents a web-based traffic control system which supports QoS of realtime packet transmission for the multimedia communication. The traffic control system presented in this paper applies the integrated service model and provides QoS of packet transmission by means of determining the packet transmission rate according to the policies of network manager and the optimal resource allocation considering the end-to-end traffic load. It also provides QoS for the realtime packet transmission through the admission controller and the packet scheduler by the modified WF^2Q^+ algorithm to support asynchronous and class-based queuing.

Ⅰ. 서 론

인터넷은 수많은 컴퓨터 통신망을 상호 연결하는 통신망 중의 통신망으로서, 1969년 미국 국방성의 컴퓨터 통신망인 ARPANET[1]으로 시작된 이래 꾸준히 발전해 오고 있으며 최근에는 그 발전 속도가 기하급수적으로 빨라지고 있다. 최근 들어 인터넷 사용자가 늘어남에 따라 인터넷에서 구할 수 있는 정보도 빠른 속도로 늘어나고 있으며, WWW (World-Wide-Web)의 대중화와 함께 다양한 인터넷 서비스가 제공되고 있다. 인터넷 서비스를 제공하는 다양한 응용 프로그램들이 네트워크 기술과 관련되어 개발되면서 네트워크 트래픽은 계속적으로 증가 하고 있다. 이러한 네트워크 트래픽의 급속한 증가는 네트워크의 회선 부족, 원하는 서비스의 응답 시간 지연 등의 네트워크 전송 품질과 관련된 여러 가지 문제점들을 야기하고 있다.

최근, 이와 같은 네트워크 상의 여러 가지 문제

* 영남대학교 컴퓨터공학과 인공지능 및 지능정보시스템 연구실(skydream@cse.yu.ac.kr, park@cse.yu.ac.kr)
논문번호 : 020275-0617, 접수일자 : 2002년 6월 17일

제 해결에 있어서 가장 필요한 정보들을 제공해 주지 못한다는 단점이 있다.

WebTrafMon은 웹 기반의 사용자 인터페이스를 채택하여 사용자가 웹 브라우저로 쉽게 트래픽 정보에 접근할 수 있도록 하였다. 이 시스템은 네트워크 트래픽에 대한 자세한 정보를 사용자들에게 제공해주기 위해서, 네트워크의 각 계층별로 트래픽 정보와 송, 수신지 정보를 함께 제공하고 있다. 이를 위해서 네트워크의 가장 하위 계층부터 상위계층까지 단계적으로 패킷의 헤더 정보를 추출해 내고, 이를 송신지와 수신지를 기준으로 웹 인터페이스를 통해 보여준다. WebTrafMon의 구성요소는 프로브(probe)와 뷰어(viewer)이다. 프로브는 네트워크 패킷으로부터 정보를 얻어내어 로그 파일에 저장하고, 뷰어는 사용자와 상호 대화식으로 작동하여 웹 브라우저를 통해 사용자에게 정보를 제공한다. 이는 프로브와 뷰어가 하나의 컴퓨터에서 동작하므로 트래픽이 급속히 증가할 때는 시스템의 속도가 성능 저하를 초래하는 문제점을 가지고 있으며, 또한 트래픽을 조절할 수 없다는 단점을 가지고 있다.

RTFM은 IETF(Internet Engineering Task Force)에서 연구 중에 있으며, 1991년의 RFC 1272[9]가 기본 개념을 제공하고, 1997년 실시간 트래픽 흐름 측정의 기본 구조가 잡혔으며, 현재 확장 작업을 하고 있다. RTFM은 실시간으로 트래픽 흐름(traffic flow)을 측정 하는 것으로 네트워크 상의 트래픽을 실시간으로 수집하여 흐름 단위로 정보를 분석하는 기법으로 메타(meter), 메타 리더(meter reader), 매니저, 분석 애플리케이션(analysis application)으로 구성되어 있다. 메타는 네트워크 내에 주어진 측정 지점(metering point)에서 트래픽 흐름에 대한 데이터를 수집하여 측정 데이터를 만들어 내고, 메타리더는 메타로부터 측정 데이터를 읽어 들인다. 매니저는 네트워크를 통해 메타와 메타리더가 통신하는 호스트에서 실행되는 프로그램으로, 메타를 설정하고 메타리더를 제어하는 관리 모듈이며, 하나 또는 그 이상의 메타를 제어할 수 있다. 분석 애플리케이션은 측정 데이터를 분석하여 사용자가 원하는 형태로 출력한다. 메타는 실시간 트래픽 흐름을 측정하고 메타 MIB를 실행하는 SNMP 에이전트이고, 매니저는 흐름 데이터를 저장하고 접근하기 위해 메타 MIB를 사용하는 SNMP 매니저이다. RTFM은 하나의 매니저가 여러 시스템들의 트래픽의 흐름을 관리 할 수 있는 장점은 있으나 하나의 시스템에서 트래픽의 흐름을 보여주는 룰셋이 하나만 동작하므로, 실시간으로 처리되는 시스템에서는 장시간에 대한 히스토리의 저장이 어려우며, 또한 트래픽 전송에 있어 QoS를 제공하지 못한다는 단점이 있다.

Ⅲ. Integrated Service 모델

IntServ 모델은 QoS를 위해 단대 단 패킷 전송 지연 시간을 극복할 수 있도록 자원예약을 위한 RSVP(resource ReSerVation Protocol)라는 시그널링(signalling) 프로토콜을 사용하여 각 흐름(flow)별로 자원을 예약 한 후 패킷을 전송하도록 하는 방식이다. 흐름 별로 각기 다른 QoS가 요구되어지므로, 각 서비스 유형에서 요구되는 QoS를 나타내는 흐름 명세(flowspec) 파라미터를 결정하고, 이를 사용하여 응용 서비스에서 요구되는 QoS를 보장하기 위해서는 망에 필요한 대역폭을 요청한다. 이를 위해 종단 호스트와 망 노드 사이에서 파라미터 정보를 전달하는데 이를 담당하는 것이 RSVP 프로토콜이다. 망 노드는 호스트가 요청한 파라미터와 현재 네트워크의 상태에 따라 서비스를 제공할 것인지 거부할 것인지 결정하게 된다. 그리고 최종적으로 응용 서비스가 요구하는 QoS를 보장하기 위해서 링크 계층에서는 패킷 스케줄링과 패킷 분류(classification) 기능을 수행한다.

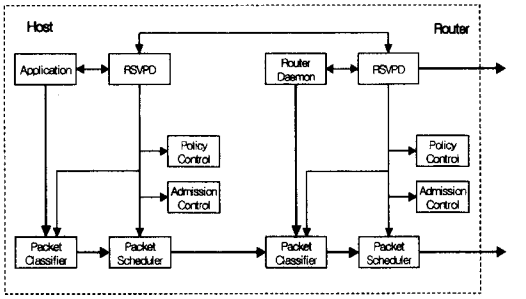

그림 1. RSVP가 동작하고 있는 호스트 및 라우터에서의 데이터 및 제어정보

그림 1은 RSVP가 수행되고 있는 라우터와 호스트간의 데이터 및 제어정보 전송에 필요한 주요한 모듈들을 나타낸 것이다. 라우터 내의 RSVP 데몬은 자원 예약을 하기 전에 요청된 패킷에 요구되는 자원이 충분한지를 판단하는 인증제어(admission control)와 자원 예약의 허용여부를 결정하는 정책제어(policy control)모듈과 통신을 수행하게 된다. 만약 이 두 모듈에서 해당 패킷에 대한 거부(reject) 판정을 하게 되면 RSVP 데몬은 응용 프로세스에게

에러정보를 전송하며, 승인(accept)으로 판정되면 RSVP 데몬은 서비스 요청을 위해 패킷 분류기(packet classifier)와 패킷 스케줄러(packet scheduler)에 적절한 파라미터를 설정하게 된다. 패킷 분류기는 각 패킷들의 QoS 클래스를 결정하고, 패킷 스케줄러는 설정된 QoS 클래스를 바탕으로 패킷 스케줄링 순서를 결정하게 된다. IntServ 모델을 지원하기 위하여 각 중간 라우터에서 흐름에 관련된 정보를 저장하여야 하기 때문에 라우터에서 이런 정보를 저장하는 공간을 요구할 뿐만 아니라 라우터의 처리 오버헤드가 커질 수 있다. 인터넷 백본 라우터인 경우 전송 속도가 상당히 높고 연결된 흐름의 개수가 많으므로 IntServ모델을 지원하기가 힘들다. 또한, IntServ 모델을 지원하기 위하여 라우터마다 RSVP, 연결 관리, 패킷의 스케줄링 등의 기능을 지원하여야 하며 라우터에 대한 성능 요구가 높다[10][11][12].

본 논문에서는 이러한 문제점을 해결하기 위하여 종단간 게이트웨이 상에 IntServ 모델을 응용한 QoS보장형 트래픽 제어 시스템을 구현함으로써 멀티미디어 통신에서 전송품질을 제공한다. 또한 다양한 흐름들을 서비스 클래스 별로 분류하여 처리하므로 시스템의 처리 오버헤드를 줄인다.

Ⅳ. 트래픽 제어 시스템

본 논문에서 제시하는 웹 기반 QoS 보장형 네트워크 트래픽 제어시스템은 그림 2에 보이는바와 같이 트래픽 제어 에이전트(traffic control agent)와 관리 서버(management server), 클라이언트 인터페이스(client interface)로 구성되어 있다.

트래픽 제어 에이전트는 수집기(collector), 패킷 분류기(packet classifier), 제어기(controller), 패킷 스케줄러(packet scheduler)로 구성되어 내부 네트워크와 외부 네트워크와의 접속 노드 상에서 동작하며, 이 노드를 통과하는 패킷에 대한 트래픽 제어 정책 반영, 지속적인 패킷 정보 수집, 수집한 패킷 정보의 관리 서버로의 전송 등을 담당한다. 관리 서버는 구성기(configurator), 정책 관리기(policy manager), 트래픽 분석기(traffic analyzer)로 구성되며, 네트워크의 트래픽에 대한 정보를 트래픽 제어 에이전트로부터 전송 받아 저장한 후, 요구에 적합한 형태로 가공, 분석한 다음 클라이언트 인터페이스에게 실시간으로 정보를 전송한다. 클라이언트 인터페이스(client interface)는 네트워크 관리자가 원격으로 실시간 트래픽에 대한 정보를 보면서 트래픽 제어 정책을 반영할 수 있도록 하는 그래픽 사용자 인터페이스를 제공한다. 이러한 구성요소들에 대한 자세한 설명은 다음의 각 절에서 주어진다.

1. 트래픽 제어 에이전트

트래픽 제어 에이전트는 노드를 통과하는 패킷 정보를 수집하는 수집기(collector), 네트워크 관리자

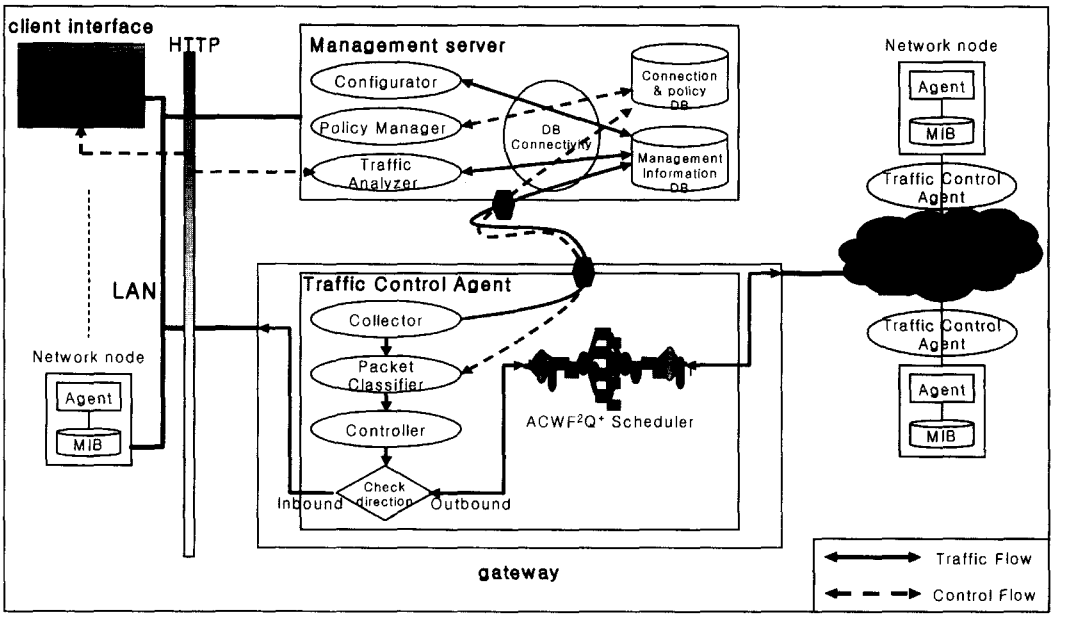

그림 2. 트래픽 제어 시스템 구성도

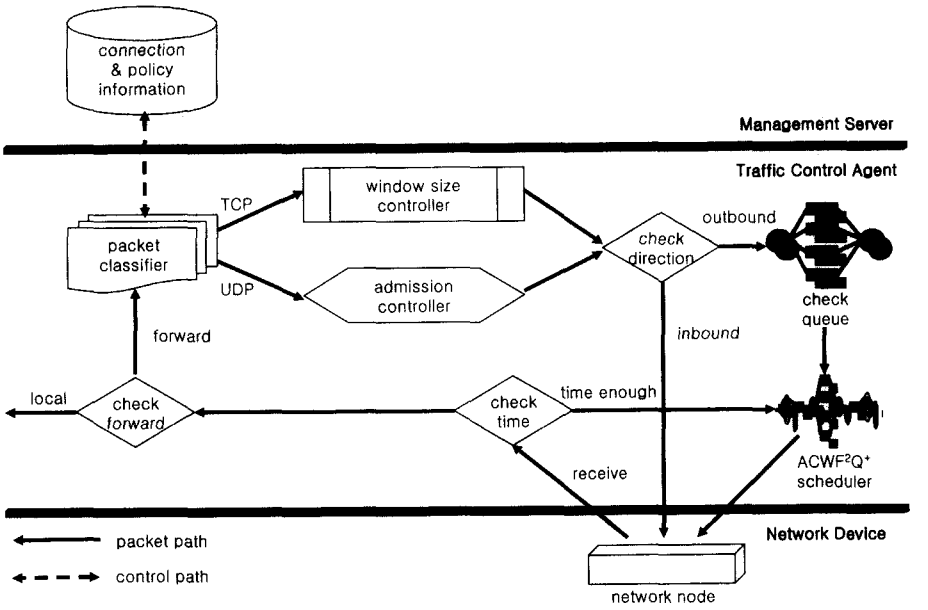

그림 3. 트래픽 제어 에이전트 내부 시스템 구성도

의 정책에 따라 패킷들을 해당 클래스로 분류하는 패킷 분류기(packet classifier), TCP 패킷의 윈도우 크기 제어와 UDP 패킷의 인증 제어를 담당하는 제어기(controller), ACWF^{2}Q$^+$ 알고리즘에 의해 출력 패킷의 순서를 제어하는 패킷 스케줄러(packet scheduler)로 구성되며, 각 구성요소들은 커널 내부의 네트워크 스택의 IP 계층에서 동작된다. 그림 3은 트래픽 제어 에이전트의 전반적인 동작과정을 각 구성요소의 기능별로 보여주고 있으며, 이를 간략하게 기술하면 다음과 같다. 먼저 트래픽 제어 에이전트는 관리 서버로부터 클래스 정책을 받는다. 네트워크 노드로부터 패킷이 수신되면 수신된 패킷이 포워드 패킷인지를 판단하여 포워드 패킷인 경우 클래스 정책 및 연결 구분에 따라 분류한 다음 해당 큐에 저장한다. 이 때 패킷의 종류가 실시간 패킷인 경우는 인증 제어를 거쳐 큐에 저장하고, TCP 패킷인 경우는 윈도우 크기 제어 과정을 거쳐 큐에 저장한다. 패킷을 큐에 저장하기 전에 내부(inbound) 패킷과 외부(outbound) 패킷을 구분하여 내부 패킷은 그대로 전송하고, 외부 패킷의 경우에만 큐에 저장하게 된다.

1) 패킷 분류기

패킷 분류기에서는 관리 서버의 정책 데이터베이스의 정책 정보를 참조하여 패킷들을 클래스별로 분류하게 된다. 해당 패킷의 클래스는 호스트 IP와 포트 번호의 쌍으로 구별할 수 있도록 구성하고, 각 클래스 별로 대역폭의 일정 비율을 할당하며, TCP나 UDP가 아닌 프로토콜은 디폴트(default) 클래스로 구성한다. 각 클래스에서 실시간 패킷은 인증

표 1. 실시간 패킷의 클래스의 구성요소

클래스 항목	내 용
Source IP	패킷의 출발지 주소
Destination IP	패킷의 도착지 주소
Source Port	패킷의 출발지 포트
Destination Port	패킷의 도착지 포트
Class Rate	하나의 클래스에 할당된 전체 대역폭 중 비율
Min Bandwidth	클래스 한 개의 연결 당 최소 대역폭

표 2. TCP 패킷의 클래스의 구성요소

클래스 항목	내 용
Source IP	패킷의 출발지 주소
Destination IP	패킷의 도착지 주소
Source Port	패킷의 출발지 포트
Destination Port	패킷의 도착지 포트
Class Rate	하나의 클래스에 할당된 전체 대역폭 중 비율
Window Rate	TCP 윈도우 사이즈 조절 비율 (기본값 : 100%)

제어를 받게 되고, TCP 패킷은 정책에 따라 주어진 윈도우 사이즈 비율만큼 윈도우 사이즈를 변경하게 된다. 표 1은 실시간 패킷의 클래스 구성요소, 표 2는 TCP 패킷의 클래스 구성요소를 보인다.

2) 윈도우 크기 제어

TCP 패킷의 헤더에는 수신 측의 윈도우 버퍼의 크기가 지정되어 있어서, 이 값이 매우 클 경우에는 송신 측에서 한꺼번에 그 크기에 해당하는 많은 양의 데이터를 전송하게 된다. 이로 인해 내부 네트워크에는 많은 패킷이 발생하게 되어 패킷 과잉(congestion) 상태를 발생시키게 된다. 본 논문에서 제시하는 트래픽 제어 에이전트의 윈도우 크기 제어기능은 송, 수신 측의 중간에서 수신 측의 TCP 패킷의 윈도우 사이즈를 변조함으로써 송신 측의 데이터 전송을 제어할 수 있도록 한다. 그림 4에서 본 논문의 트래픽 제어 에이전트가 중간 노드 역할을 하면서 수신 측의 윈도우 크기를 축소하는 과정을 보이고 있다.

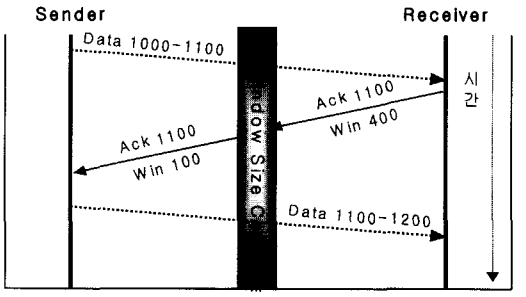

그림 4. 중간노드에서 TCP 패킷 변조

그림 4에서 보면, 송신 측에서 데이터를 전송한 후 이에 대한 TCP 패킷의 윈도우 사이즈가 400일 경우, 트래픽 제어 에이전트의 윈도우 크기 제어에서 수신 측의 윈도우 버퍼 크기를 100(25%)으로 변조하여 송신 측의 데이터 전송 량을 줄임으로써 내부 네트워크의 패킷 과잉을 조절할 수 있다. 본 논문의 윈도우 크기 제어에서는 이러한 기능을 수행하기 위해서 해당 클래스의 구성요소 중에서 윈도우 비율(window rate)을 참조하여 전송하는 TCP 패킷 헤더의 수신 측 윈도우 버퍼 크기를 윈도우 비율만큼 변경하고, 변경된 TCP 헤더를 위한 TCP 체크썸(checksum)을 수행한다.

3) 인증 제어

모든 클래스는 새로운 연결이 추가될 때마다 각 클래스 별로 할당된 대역폭을 연결별로 공평하게 나눈다. 새로운 연결이 추가되면 연결별 대역폭이 줄어들게 되며, 특히 실시간 클래스의 경우에는 최대 지연 값을 만족하지 못할 수도 있게 되어 심각한 전송 품질의 저하를 초래할 수 있게 된다. 본 논문의 트래픽 제어 에이전트의 인증 제어기에서는 실시간 클래스의 연결별 대역폭이 최소 대역폭보다 작아지는 경우 추가의 연결을 거절함으로써 실시간 트래픽에 대한 전송품질을 보장하게 된다. 그림 5에서는 클래스와 클래스별 연결의 구성 및 인증 제어의 수행이 필요한 예를 보이고 있다.

그림 5에서 실시간 클래스 ①은 전체 대역폭의 30%를 사용하며 현재 3개의 연결이 존재한다. 현재 각 연결은 10%씩 대역폭을 할당받고 있으며, 새로운 연결이 요구될 때 마다 인증 제어를 거쳐 연결에 대한 가부가 결정되고, 나머지 클래스들은 새로운 연결의 요구에 대해서 아무런 인증 절차가 없이 연결이 허용된다.

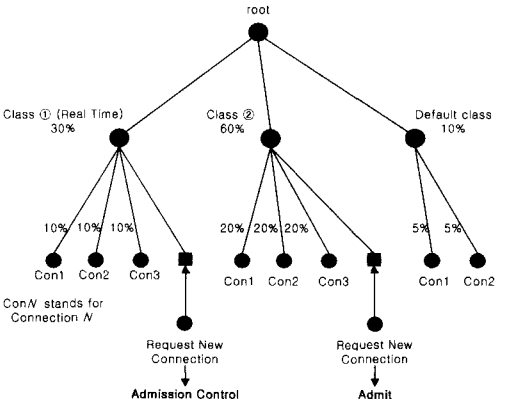

그림 5. 클래스와 연결의 관계도

4) 패킷 스케줄러

본 논문에서의 패킷 스케줄러는 패킷 큐잉 알고리즘으로 잘 알려진 WFQ(Weighted Fair Queueing)[13]의 제약점을 보완한 WF^2Q(Worst- case Fair Weighted Fair Queueing)[14]와 이의 구현 복잡도(implementation complexity)를 낮춘 WF^2Q^+ 알고리즘[15]을 기반으로 비동기(asynchronous) 큐잉과 클래스 기반(class- based) 큐잉을 지원하도록 변형한 $ACWF^2Q^+$ (Asynchronous and Class-based WF^2Q^+) 알고리즘으로 동작한다. 본 논문의 $ACWF^2Q^+$ 스케줄링 알고리즘은 WF^2Q^+의 가상 서비스 시간 함수를 그대로 이용하고 있으며, 클래스마다 busy field를 두어 비동기 큐잉을 지원하며, WF^2Q^+ 알고

리즘에서 흐름의 수가 많아 질 경우 정렬 작업의 복잡도가 증가하는 문제를 해결하기 위해 클래스별로 정렬 작업을 수행한다.

비동기 큐잉은 기존 방식에서 송수신시 하나의 흐름에 대해 하나의 큐잉 작업이 존재하는 것과는 달리, 큐 입력(enqueue)할 때 수신 인터럽트에 의한 하나의 작업과 큐 출력(dequeue)할 때 송신 인터럽트에 의한 작업이 독립적으로 존재한다. 즉, 비동기 큐잉에서는 큐 입력이 수행된 후 연결의 부모 클래스와 루트 노드의 상태에 따라 바로 큐 출력이 실행될 수도 있고, 수신 여부와 상관없이 큐 출력만 독립적으로 발생할 수도 있다. 이 경우 전송중임을 나타내는 busy field를 두어 큐 입력 작업에서 스케줄러를 호출하지 못하도록 하여, 특정 클래스에만 빈번한 스케줄링이 일어나는 것을 방지함으로써 공평한 전송을 제공한다. 클래스 기반 큐잉은 일반적인 스케줄링 알고리즘의 구현에 있어서 정렬의 대상 수가 많아질 때 구현 복잡도 증가 문제를 보완하기 위해서, 흐름의 서비스율이 비슷한 종류의 흐름들을 하나의 클래스로 구성하여 클래스 단위로 정렬하고 전송할 패킷을 결정한다. 이는 패킷의 크기가 고정된 ATM 네트워크에서 서비스 요구율에 따라 클래스를 구성하는 것과는 달리, 패킷 크기가 가변인 패킷 네트워크의 경우, 흐름을 서비스 요구율에 따라 클래스를 구성하는 것은 의미가 없기 때문이다. 따라서, 본 논문의 클래스 기반 큐잉은 패킷의 서비스 간격에 따라서 흐름을 특정 클래스로 분류하고, 이를 기반으로 정렬하여 전송 패킷을 결정한다.

그림 6에서 본 논문의 ACWF^{2}Q$^+$ 스케줄러에서의 큐 입력과 큐 출력의 동작 과정을 보이고 있으며, 이를 각 단계별로 설명하면 다음과 같다.

(1) 패킷을 수신하면 패킷 분류기를 거쳐 큐 입력 단계로 간다.

(2) 이 때, 해당 연결의 큐가 비어 있으면 연결의 가상 시간을 설정한 후 입력하고, 그렇지 않으면 바로 큐 입력한다.

(3) 패킷의 부모 클래스의 busy field가 0이면 스케줄링을 수행한다.

(3-1) 스케줄링이 끝난 후, 패킷의 부모 클래스의 연결 중 HOL(head of line)에 있는 패킷들에 대해서 스케줄링을 수행한다.

(3-2) 가상 시간이 가장 작은 연결이 선택되고, 이 연결의 패킷이 다시 부모 클래스의 최종 패킷으

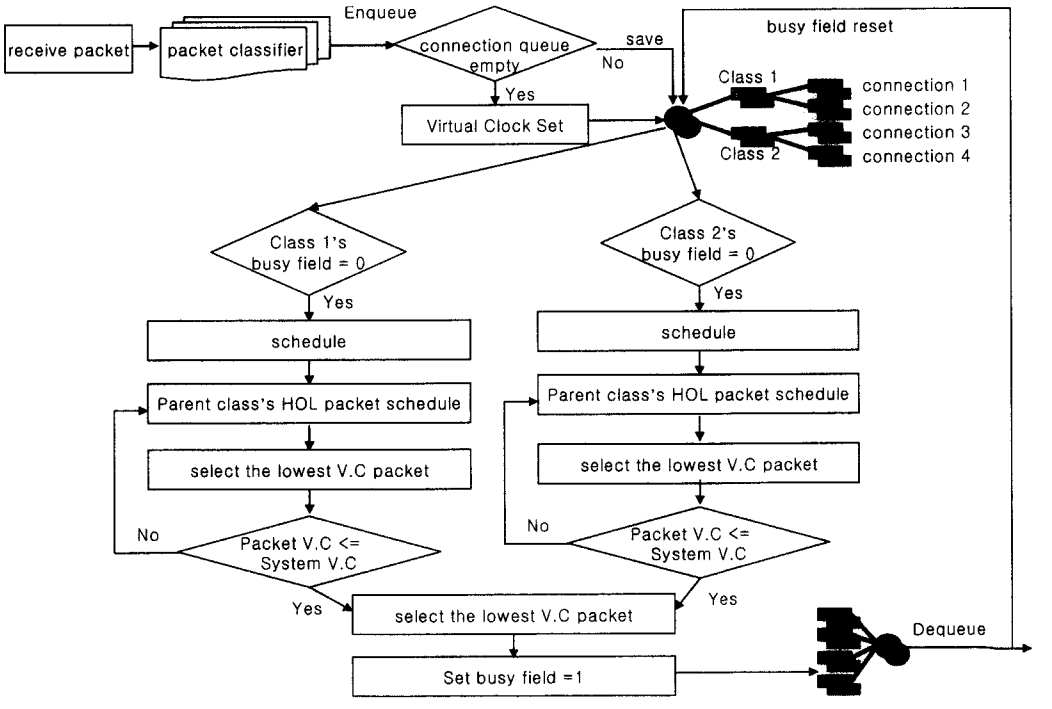

그림 6. ACWF^{2}Q$^+$ 스케줄러의 큐 입력, 출력 동작과정

로 선택한다.

(3-3) 루트 노드 아래의 모든 클래스 중 하나가 같은 방식으로 다시 선출한다.

(3-4) 선출된 클래스의 해당 패킷은 다른 클래스와의 경쟁을 통하여 가장 작은 가상시간을 갖는 패킷이 서비스할 패킷으로 선택한다.

(4) 서비스할 패킷이 결정되면 busy field를 1로 설정하고 큐 출력을 실행한다.

(5) 큐 출력 후 다음 서비스를 위해 최근 전송된 클래스와 연결을 찾아 busy field를 리셋한다. 이러한 과정을 반복한다.

2. 관리 서버

관리 서버는 그림 2에서 보인 바와 같이 구성기, 정책 관리기, 트래픽 분석기로 구성되며, 네트워크의 트래픽에 대한 정보를 트래픽 제어 에이전트로부터 전송 받아 저장한 후, 요구에 적합한 형태로 가공, 분석한 다음 클라이언트 인터페이스에게 실시간으로 정보를 전송한다.

구성기는 트래픽 모니터링에 필요한 네트워크 장비 및 네트워크 구성에 대한 정보를 가져와 데이터베이스에 저장한다. 네트워크 구성 정보는 모니터링에 필요한 장비에 대한 정보뿐만 아니라 장비들의 네트워크 인터페이스 정보를 포함한다. 정책 관리기는 네트워크 관리자가 여러 클래스에 대한 세밀한 정책 결정을 반영할 수 있도록 해주며, 이를 위해 최대 10개의 클래스에 대한 정책을 관리할 수 있다. 정책 반영을 위한 각 클래스는 IP, 포트 번호 등에 대한 값을 입력받아 저장하고 있으며, 나머지는 디폴트 클래스로 적용된다. 트래픽 분석기는 트래픽 제어 에이전트로부터 수집한 트래픽 정보를 분석하여 동적 서브넷 리포팅, CPU 부하 모니터링 등의 작업을 수행한다.

본 논문의 관리 서버는 Java로 작성되어 플랫폼에 독립적이며, JDK의 표준 API로 작성되어 별도의 다른 패키지가 필요하지 않다. 관리 서버의 기본 기능은 트래픽 제어 에이전트와 클라이언트 인터페이스가 원활하게 동작될 수 있도록 이들에서 발생하는 정보들을 저장, 처리, 분석, 전송 등을 수행하는 것이다. 특히, 관리 서버는 트래픽 제어 에이전트와 클라이언트 인터페이스 사이의 정보 전송을 실시간으로 처리해야 하며, 이들과의 정보 전송이 서로 독립적으로 이루어져야 하기 때문에 서로 간의 작업이 비동기적으로 동작되어야 한다. 이러한 동작은 다음과 같은 Java 메소드 코드로 설명될 수 있다.

```
private Thread thread;
private void start() throws Exception() {
    thread = new Thread(this);
    thread.setPriority(Thread.MIN_PRIORITY);
    thread.setName("ManagerServer");
    thread.start();// 클라이언트 인터페이스로 정보를
                   // 전달하기 위한 쓰레드 시작
    receiveDate();// 트래픽 제어 에이전트로부터의
                   // 실시간 정보를 받는 데이터그램
                   // 서버 소켓
}
```

관리 서버가 클라이언트 인터페이스에게 실시간 정보를 보내는데 필요한 통신은 데이터그램 소켓을 통해 전송하지만, 정책에 대해서는 패킷의 손실이 있을 수 있기 때문에 정보 유실방지와 신뢰도를 유지하기 위해 Java RMI(remote method invocation)를 이용하여 관리한다. 다음은 원격 메소드(remote method)를 이용한 정책의 송수신작업을 수행하는 Java 코드를 보여주고 있다.

```
public synchronized Void
setRealtimePKinfo(boolean flag) {
  pkdata[pkcnt].source = ntohs(tcph->source);
                    ⋮
  //1초 간격으로 수집된 패킷을 매니저 서버로 전달한다.
  if((current_time=time(NULL) >
      last_time + TIME_INTERVAL) {
                    ⋮
      sendto(sockfd, buf, n, 0,
                (struct sockaddr *)& ser_addr,
                    sizeof(ser_addr));
                    ⋮
  }
}
```

3. 클라이언트 인터페이스

클라이언트 인터페이스는 네트워크 관리자가 트래픽 제어 에이전트를 원격에서 관리하기 위한 도구로서, 관리자가 실시간 트래픽에 대한 정보를 보면서 트래픽 제어 정책을 반영할 수 있도록 하는 그래픽 사용자 인터페이스를 제공한다. 본 논문의 클라이언트 인터페이스는 트래픽 모니터링 인터페이스와 정책 작성 인터페이스로 구성되어 있으며, 이들은 모두 표준 Java API의 Java Applet으로 구현되어 웹 페이지 상에 쉽게 포함시킬 수 있어서 웹

브라우저가 있는 어떤 곳에서도 관리 대상 네트워크를 관리할 수 있다.

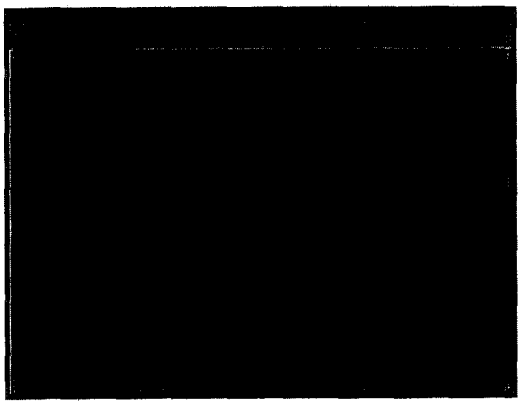

그림 7. 트래픽 모니터링 실행화면

트래픽 모니터링 인터페이스는 Java 2D로 구현되어 대상 네트워크 상의 전체 트래픽 및 호스트별, 클래스별 트래픽을 그래픽 화면으로 제공한다. 트래픽 모니터링 인터페이스는 데이터그램 소켓을 열어 실시간으로 트래픽 정보를 받음과 동시에 비동기적 메소드를 이용하여 1초 간격으로 패킷 정보를 화면에 나타내어 주고, 관리 서버의 원격 메소드를 호출하여 정보를 주고받는다. 그림 7에서 네트워크의 트래픽을 모니터링하고 있는 화면을 보이고 있다.

정책 작성 인터페이스는 관리자가 다양하게 정책을 적용할 수 있도록 10개의 클래스에 대한 항목값을 입력할 수 있도록 작성되었으며, 각 클래스는 IP 어드레스, 포트번호 등을 입력할 수 있고, 나머지는 기본 클래스로 적용되게 된다. 그림 8에서 정책 작성 인터페이스의 실행화면을 보인다.

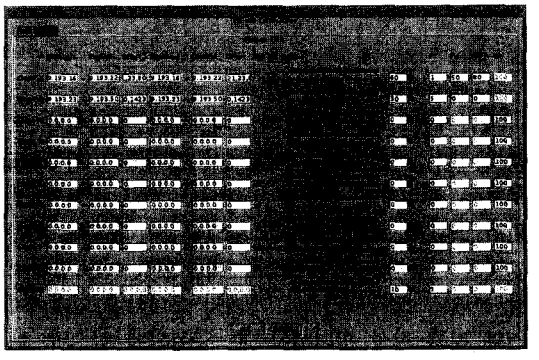

그림 8. 정책 작성 사용자 인터페이스

V. 실험 및 고찰

본 논문의 QoS 보장형 트래픽 제어 시스템을 통한 네트워크 트래픽 관리를 실험하기 위해서, Windows Media Encoder 7.0을 사용하여 높은 대역폭을 가지는 동영상 스트림 데이터 타입의 advanced stream format(asf) 파일을 생성하여 전송 데이터로 이용하였으며, 각 종단 시스템에서는 패킷 생성기를 사용하여 임의로 트래픽을 발생시켰다. 그림 9는 본 논문의 실험에서 이용한 패킷 생성 화면을 보이고 있다.

그림 9. 패킷 생성 화면

그림 10과 그림 11은 이러한 실험 데이터를 이용하여 실제 트래픽 제어를 적용한 결과를 그래프로 나타낸 것이다. 정책 적용전의 트래픽의 흐름은 기존의 FIFO방식으로 전송되는 그래프의 형태이며, 정책 적용후의 트래픽 흐름은 본 논문에서 제안한 QoS 파라미터를 적용했을 때의 그래프이다. 발생된 트래픽은 모두 외부로 흘러나가는(outbound) 패킷이며, 그 대역폭은 1.2Mbyte/sec 보다 크도록 하였다.

그림 10에서 각 클래스에 정책을 적용하기 전에

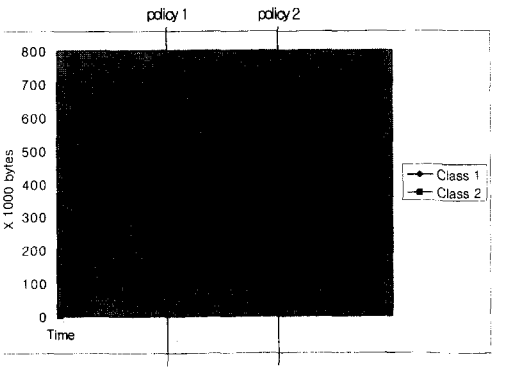

그림 10. 정책 적용전의 트래픽과 적용후의 트래픽

는 동일한 전송 량으로 전송이 이루어지다가 정책 1에서 클래스 1과 클래스 2에 각각 70%, 30%의 비율로 전송 량을 조절한 결과 7:3의 비율로 전송이 이루어지는 것을 보이고 있다. 정책 2에서 전송량을 각각 30%, 70% 비율로 재조정한 결과 전송량이 3:7의 비율로 전송이 이루어짐을 볼 수 있다.

그림 11에서는 클래스 1, 클래스 2, 클래스 3, 클래스 4에 각각 40%, 30%, 15% 10%의 비율로 전송 량을 주었으며, 기본 클래스에는 5%의 전송량을 배정한 경우의 그래프 화면이다. 그림 11에서와 같이 클래스 3, 클래스 4가 유휴상태에 있을 때 클래스 1과 클래스 2가 유휴 대역폭을 할당받아 사용하다가 클래스 3, 클래스 4가 유휴 상태에서 깨어나면 클래스 1과 클래스 2는 자신에게 주어진 대역폭을 사용함을 보여준다.

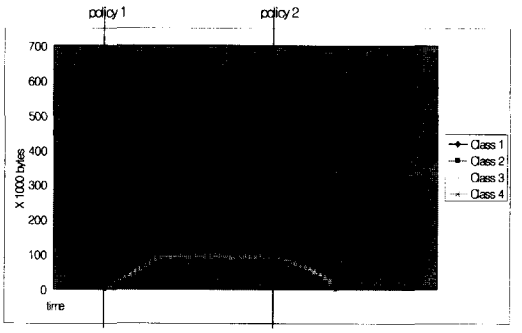

그림 11. 타 클래스 휴지 시 트래픽과 활동 중일 때의 트래픽

Ⅵ. 결 론

본 논문에서는 IntServ(Integrated Service) 모델의 구조를 응용한 웹 기반 QoS 보장형 트래픽 제어 시스템을 제시하였다. 본 논문에서 제시한 트래픽 제어 시스템은 내, 외부 네트워크의 접속 노드에 트래픽 제어 에이전트를 동작시켜 패킷의 재전송 빈도 감소 및 종단간 데이터흐름제어와 트래픽 폭주를 방지하고, 연결 회선별로 대역폭을 관리하여 같은 클래스 내에서 모든 접속 세션이 균등한 대역폭을 가질 수 있도록 구현하였다. 또한 실시간 패킷의 경우, 인증제어기와 $ACWF^2Q^+$ 패킷 스케줄러를 두어 전송 품질 보장을 지원하고 있다. 이 연구의 후속 연구는 본 논문에서 제시한 트래픽 제어 시스템에 지능형 에이전트가 개입되어 동작하는 지능형 네트워크 관리 시스템 구조를 설계, 구현하는 것이다.

참 고 문 헌

[1] Michael Hauben, "History of ARPANET", http://www.dei.isep.ipp.pt/docs/arpa-Content.html.

[2] Tobias Oetiker and Dave Rand, "MRTG: Multi Router Traffic Grapher", http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html.

[3] Craig Hunt, *TCP/IP Network Administration*, O'Reilly and Associates, Inc., 1992.

[4] V. Jacobson, C. Leres, and S. McCanne, "tcpdump", ftp://ftp.ee.lbl.gov.

[5] J. Won-Ki Hong, Soon-Sun Kwon and Jae-Young Kim, "WebTrafMon: Web-based Internet Network Traffic Monitoring and Analysis System", *Computer Communications*, Elsevier Science, Vol. 22, No. 14, pp. 1333-1342, September 1999.

[6] S. Handelman, S. Stibler, N. Brownlee and G. Ruth, "RTFM: New Attributes for Traffic Flow Measurement," *IETF RFC 2724*, October 1999.

[7] N. Brownlee, C. Mills, and G. Ruth, "Traffic Flow Measurement : Architecture," *IETF RFC 2722*, October 1999.

[8] Myung-Sub Lee and Chang-Hyeon Park, "Design and Implementation of Web-based Traffic Monitoring and Analysis System", *Proceedings of the 2001 International Conference on Internet Computing*, pp. 479-485, Las Vegas, June 2001.

[9] C. Mills, D. Hirsh, G. Ruth, "Internet Accounting : Background", *IETF RFC 1272*, November 1991.

[10] IETF, Integrated Services(IntServ) Working Group. http://www.ietf.org/html.charters/intserv-charter.html

[11] W. Almesberger, Linux Traffic Control Implementation Overview, Technical Report, EPFL, November 1998. http://tcng. sourceforge.net.

[12] S. Shenker, C. Partridge, R. Guerin, "Specification of Guaranteed Quality of Service", *IETF RFC 2212*, September 1997.

[13] A. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services