멀티미디어 서비스를 위한 DiffServ 망에서의 빠른 혼잡 제어 알고리즘

준회원 박 종 훈*, 정회원 유 명 식*

Early Rate Adaptation Protocol in DiffServ for Multimedia Applications

Jonghun Park* *Associate Member*, Myungsik Yoo* *Regular Members*

요 약

인터넷에서 멀티미디어 데이터가 트래픽의 상당 부분을 차지하면서 네트워크 혼잡 상황을 효과적으로 제어하는 방법이 필요하게 되었다 또한 현재의 Best-effort 망을 통한 멀티미디어 데이터의 전송은 한정된 네트워크 자원과 다양한 환경 등으로 인해 QoS 보장이 어려워, 안정된 QoS를 제공하기 위한 IntServ, DiffServ 등의 새로운 QoS 망이 대두되었다 하지만, RAP와 같은 송신측 기반 혼잡 제어 알고리즘은 혼잡 상황 제어에 ACK 응답에 따른 RTT 값의 변화를 이용하기 때문에, RTT 값이 큰 플로우는 상대적으로 RTT 값이 작은 플로우에 비해 ACK 응답이 늦어져 혼잡 상황에 대한 대처 시간이 늦어진다. 또한 DiffServ와 같은 QoS 망에는 호스트 단의 혼잡 제어 알고리즘과는 별도의 혼잡 제어 알고리즘이 존재하여 호스트 단의 알고리즘에 악영향을 줄 수 있다.

본 논문에서는 네트워크 중심부의 혼잡 상황에 대하여 End-to-end ACK 응답에 의한 반응 이전에 네트워크의 혼잡 정보를 유기적으로 이용하여 혼잡 상황에 빨리 대처하는 기법을 제안한다 제안된 기법을 통해 기존의 RTT 기반의 혼잡 제어 기법의 문제점을 해결하고 그 성능을 향상 시켰다

Key Words : Congestion Control, Rate Control, Multimedia QoS, TCP-friendly, Differentiated Service

ABSTRACT

As the multimedia application traffic takes more portion in the internet traffic, it is necessary to control the network congestion through the congestion control protocol In addition, the QoS-enabled networks such as DiffServ become an indispensable technology when running the multimedia applications. However, the previously proposed end-to-end congestion control algorithms take the round trip time to react the network congestion Thus, as the RTT becomes larger, the reaction against the congestion gets delayed further, while the network congestion gets worse. In addition the performance of end-to-end congestion control algorithm is degraded if the QoS-enabled network runs the congestion control mechanism in the network level without any coordination between them. In this paper, we propose the early rate adaptation protocol for the DiffServ network, which effectively links the congestion control algorithm at the host and the congestion mechanism in the network together By taking advantage of early congestion notification from the network, it is possible to react the network congestion more quickly and effectively

*숭실대학교 정보통신전자공학부 (bluearts@hanmail net, myoo@e ssu ac kr)
논문번호 #KICS2004-06-052, 접수일자 2004년 6월 15일
※ 본 연구는 숭실대학교 지원으로 수행되었음

Ⅰ. 서 론

확장성과 융통성이 뛰어난 IP의 등장으로 인해 인터넷은 괄목할만한 성장을 해왔다 그러나 멀티미디어 데이터의 비중이 점점 커지면서 이를 효과적으로 조절하는 방법에 대한 연구가 더욱 중요하게 되었다 한편 현재의 Best-effort 서비스 인터넷은 예측할 수 없는 지연 및 손실을 초래하기 때문에 멀티미디어 서비스와 같은 다양한 응용 서비스 지원을 위하여 서비스 특성에 따른 QoS(Quality of Service) 보장을 요구받고 있다[1][2]

인터넷 전화, 스트리밍 비디오처럼 실시간성을 요구하는 데이터는 일반적인 데이터와는 달리 전송 지연에는 민감한 반면 적은 패킷 손실은 어느 정도 허용하는 QoS 특성을 가지고 있다 이러한 멀티미디어 데이터의 전송에는 일반적으로 TCP와 UDP가 사용될 수 있지만, TCP는 신뢰성 있는 전송으로 인하여 실시간 멀티미디어 데이터 전송에는 한계가 있고, UDP는 네트워크의 혼잡 상황을 조절하는 능력이 없어서 혼잡 상황을 가중시킬 수 있기 때문에 멀티미디어 스트리밍 데이터 전송에는 최적의 성능을 보여주지 못한다 이에 대해 혼잡 상황에 따른 조절 능력을 갖고 있는 TCP의 개념을 UDP 기반의 응용계층에서 동작하도록 하는 TCP-friendly 혼잡 제어 알고리즘이 멀티미디어 데이터 전송의 QoS 보장을 위해 연구되어 왔다.

현재 멀티미디어 스트리밍 데이터에 대한 TCP-friendly 알고리즘은 혼잡 상황을 예측하고 반응하는 주체에 따라 송신측 기반 혼잡 제어 알고리즘, 수신측 기반 혼잡 제어 알고리즘 그리고 혼합 방식 등으로 나뉜다 송신측 기반 혼잡 제어 알고리즘은 대표적으로 RAP(Rate Adaptation Protocol)가 있고, 수신측 기반 혼잡 제어 알고리즘은 대표적으로 DSG(Destination Set Grouping)가 제안되었다[3][4].

RAP는 수신자의 ACK 응답과 전송한 패킷에 대한 손실 및 지연 등의 혼잡 상황 예측을 통하여 AIMD(Additive Increase, Multiplicative Decrease) 방식으로 전송 속도를 조절한다 DSG는 수신측에서 네트워크의 상태에 따라 채널을 더하거나 뺌으로서 수신 속도를 조절하는 방식이다

그러나 호스트 단의 응용계층 혼잡 제어 알고리즘만으로 멀티미디어 서비스 QoS를 만족시키기에는 어려움이 있다.

IETF(Internet Engineering Task Force)는 인터넷 QoS를 제공할 목적으로 자원 예약 프로토콜(RSVP), 수락제어(Admission Control) 등으로 구성되어 있는 IntServ(Integrated Service)를 제안하였다[5][6][7] 하지만 IntServ는 백본 라우터의 과도한 오버헤드와 네트워크의 확장성의 문제로 인해 지금은 논의가 주춤한 상태다.

IETF는 IntServ의 확장성 문제를 해결하기 위하여 QoS 보장을 위한 새로운 개념으로 DiffServ(Differentiated Service)를 제안하였다[8][9][10] DiffServ는 개별 플로우가 아닌 많은 수의 플로우를 몇 개의 클래스로 나누어 중간 라우터에서 클래스별로 처리한다 DiffServ의 차등화 서비스를 위한 정보는 IPv4 패킷 헤더의 ToS(Type of Service) 필드 또는 IPv6 패킷 헤더의 TC(Traffic Class) 필드 중에서 6bit를 DSCP(Differentiated Service Code Point)로 사용하고, 버퍼 관리와 스케줄링을 위해 사용된다[11].

본 논문에서는 DiffServ 망에서 세 가지 혼잡 제어 알고리즘의 성능을 비교 및 분석한다 첫 번째는 수정이나 보완이 없는 기본적인 RAP 알고리즘에 대한 성능 평가로서 나머지 알고리즘에 대한 비교 잣대가 된다. 두 번째는 Edge 라우터에서 빠른 혼잡 제어를 수행하는 Router-based Rate Control 알고리즘에 대한 성능 평가이고 마지막 세 번째는 호스트 단에서 빠른 혼잡 제어를 수행하는 Host-based Rate Control 알고리즘에 대한 성능 평가이다 세 알고리즘의 비교를 통해 송신측 기반 응용계층 혼잡 제어 알고리즘과 DiffServ 망의 상호 연계를 통하여 네트워크의 혼잡 상황을 빠르고 효율적으로 제어할 수 있는 알고리즘의 필요조건을 파악하고, 그 성능을 극대화할 수 있는 방법에 대하여 논의하고자 한다.

본 논문의 구성은 다음과 같다. Ⅱ는 본 논문의 배경이 되는 송신측 기반 응용계층 혼잡 제어 알고리즘과 DiffServ 망의 차등화 서비스에 대하여 설명하고 문제점에 대하여 서술한다 Ⅲ에서는 빠른 혼잡 제어에 대한 호스트 단과 네트워크의 상호 연계 방법에 대하여 제안하고, Ⅳ에서 시뮬레이션 환경 및 세 알고리즘에 대한 실험 결과에 대한 비교와 분석을 한다 마지막으로 Ⅴ에서 결론을 맺고자 한다

Ⅱ. 관련 연구

1 RAP Rate Adaptation Protocol

현재 멀티미디어 응용의 혼잡 제어 방식 중 가장 활발히 연구가 진행되는 방법은 전송 속도 제어 방식

이다[12][13] 전송 속도 제어는 네트워크의 가용한 대역폭을 예측하여 멀티미디어 데이터의 전송 속도를 결정하는 기술이다.

RAP는 송신측 기반 혼잡 제어 알고리즘으로 기본적으로 AIMD에 기초를 둔다. 송신한 패킷에 대하여 수신측은 응답 패킷을 보내고, 송신측은 해당 응답 패킷에 기초하여 패킷간의 간격을 조정하여 전송 속도를 조절한다 전송 속도의 증가는 TCP의 Congestion Avoidance처럼 일정 주기로 조정되어 증가한다[14]

RAP는 혼잡 상황을 감지하면 즉시 전송 속도를 반으로 줄인다. RAP는 수신측으로부터 오는 응답 패킷의 정보를 판독하거나 전송한 패킷에 대한 타임아웃 여부 검사로 혼잡 상황 여부를 판단한다

RAP와 같은 송신측 기반 혼잡 제어 알고리즘은 전송한 패킷에 대한 응답으로부터 네트워크의 혼잡 정도를 예측한다. 하지만 RAP와 같이 RTT 값을 네트워크의 혼잡 상황 예측에 이용하는 알고리즘은 송신측과 수신측의 전송 시간이 길면 길수록 혼잡 상황 감지가 늦어지게 된다 따라서 이러한 알고리즘은 네트워크의 혼잡 상황을 늦게 판단하는 단점이 갖고 있어 보완이 필요하다

2. DiffServ. Congestion Control

DiffServ는 IntServ의 확장성 문제를 해결하기 위해 많은 수의 플로우를 몇 개의 클래스로 분류하여 서비스한다. DiffServ는 패킷 전달 방식에 따라 DE(Default) PHB(Per Hop Behavior), EF(Expedited Forwarding) PHB, AF(Assured Forwarding) PHB의 세 가지 형태로 서비스를 구분한다.

*DE PHB*는 현재 인터넷에서 사용 중인 *Best-effort* 방식을 정의하여 기존의 트래픽을 포용하는 서비스 클래스이다. EF PHB는 우선순위가 가장 높은 전달 방식으로, 버퍼에서의 전송 지연을 최소화하고 패킷 손실도 최소화한 서비스를 제공한다 AF PHB는 패킷 전달 순서를 결정하는 4개의 클래스와 혼잡 상황 발생시 패킷을 폐기하는 3단계의 폐기우선순위에 의해 정의된다.

DiffServ에서 사용되는 서비스 차등화 알고리즘은 대부분 RED(Random Early Detection)나 변형된 RED를 사용하여 혼잡 상황을 파악하여 서비스한다[15] RED는 라우터의 큐들에 대하여 미리 정의해둔 최소 임계값(min_th)과 최대 임계값(max_th)을 평균 큐의 길이(avg)와 비교하여 도착하는 패킷의 폐기 여부를 결정한다. RED는 도착하는 패킷들에 의하여 버퍼의 큐 크기가 최소 임계값을 넘는 경우에 확률적으로 패킷을 폐기하고, 최대 임계값을 넘어가면 도착하는 모든 패킷을 폐기하도록 한다.

혼잡 상황에 대한 DiffServ 망에서의 전송 속도 제어 방법은 크게 Edge 라우터에서 트래픽 전체의 전송 속도를 조절하는 방법과 망이나 수신측으로부터 얻은 정보를 이용하여 호스트 단에서 전송 속도를 조절하는 방법이 있다 대표적으로 라우터에서 전송 속도를 조절하는 방법은 E2E-CCS(Edge-to-edge Congestion Control Scheme)가 있고, 호스트 단에서 전송 속도를 제어하는 방법은 TPDCM(Two Phase Distributed Congestion Management)이 있다[16][17].

E2E-CCS에서 Ingress 라우터는 Egress 라우터로 일정시간마다 QoS 컨트롤 패킷을 전송한다 Ingress 라우터는 QoS 컨트롤 패킷에 대한 피드백 정보에 의해서 트래픽의 전송 속도를 결정한다. E2E-CCS는 Ingress 라우터에서 네트워크로 흘려보내는 전체 트래픽의 전송 속도를 조절함으로서 네트워크의 혼잡 상황과 무관한 플로우의 전송 속도도 줄어든다 따라서 전체 트래픽을 조절하는 E2E-CCS는 네트워크의 혼잡 상황에 대한 개별 호스트 단의 혼잡 상황 대처에는 부족하다

TPDCM은 호스트 단의 전송 속도 조절과 Edge 라우터의 전송 속도 조절로 이루어져있다 Core 라우터는 혼잡 상황이 감지되면 Edge 라우터로 FCMP(Flow Control Message Protocol) 메시지를 보내고, FCMP 메시지를 받은 Edge 라우터는 자신의 전송 속도를 조절한다. Edge 라우터는 자신에게 유입되는 트래픽으로 인하여 혼잡 상황이 발생하면 호스트 단에 FCMP 메시지를 전송하고, 호스트 단은 자신의 전송 속도를 조절하여 유입량을 조절한다 그러나 TPDCM도 네트워크 중심부의 혼잡 상황에 대해서는 Edge 라우터에서만 전송 속도를 조절하여 단대단 지터나 패킷 손실 여부가 중요한 멀티미디어 스트리밍 데이터 전송에는 부족한 점이 있다.

Ⅲ. ERAP (Early Rate Adaptation Protocol)

제안하는 알고리즘은 호스트와 라우터의 혼잡 제어 알고리즘의 두 축으로 구성되어 동작한다. 이 알고리즘의 특징은 다음과 같다. Core 라우터는 큐의 크기 변화를 통하여 혼잡 상황을 감지한다. 혼잡 상황을 감지한 Core 라우터는 즉시 Ingress 라우터로 혼잡 상황을 통보하는 패킷(Congestion Notification

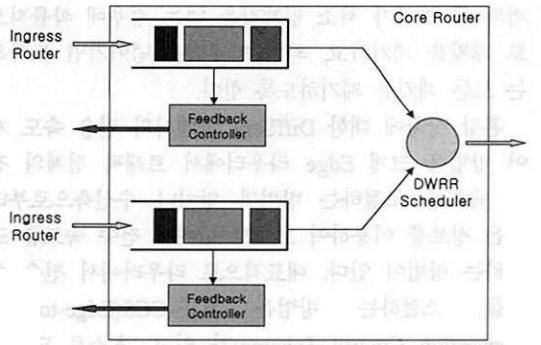

그림 1. Core 라우터의 구조

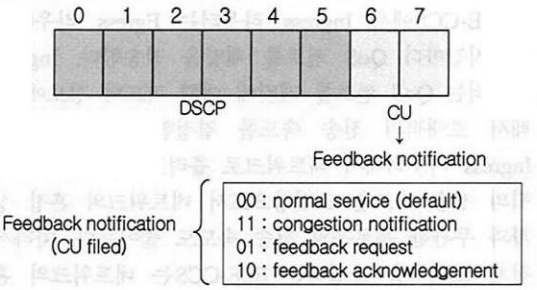

그림 2. 피드백 통보 메시지 설정 필드

Packet)을 전송하게 된다. 혼잡 상황 통보 패킷을 받은 Ingress 라우터는 혼잡 상황을 야기한 해당 호스트 단으로 이 사실을 알려주고, 해당 호스트 단은 해당 플로우(Flow)의 전송 속도를 조절하게 된다. 라우터와 호스트 단의 혼잡 제어 알고리즘은 각각 다음과 같다.

1. 라우터의 혼잡 제어 알고리즘

Core 라우터의 구조는 그림 1과 같다. Core 라우터는 혼잡 상황이 발생하면 피드백 제어기(Feedback Controller)에 이 사실을 통보하고, 혼잡 상황을 통보받은 피드백 제어기는 Ingress 라우터로 혼잡 상황 통보 패킷을 전송한다.

혼잡 상황 통보 패킷은 그림 2와 같이 패킷 헤더의 CU(Currently Unused) 필드를 사용한다. 피드백 제어기는 혼잡 상황을 일으킨 패킷을 통하여 회신할 곳의 주소를 획득 할 수 있으며, CU 필드를 11로 설정한 혼잡 상황 통보 패킷을 생성한다. 생성된 패킷은 피드백 제어기를 통하여 Ingress 라우터로 전송된다. Core 라우터의 혼잡 상황 판단 알고리즘은 기본적으로 RED의 알고리즘을 이용하며 추가된 부분은 그림 3과 같다.

Ingress 라우터는 전송 속도 조절과 호스트 단에 혼잡 상황을 통보하는 두 가지의 알고리즘으로 구성

```
// Check the congestion condition
REPEAT every packet arrival
   Calculate the average queue size avg
      using detailed algorithm for RED gateways of [15]

   Calculate the condition of queue
      using general algorithm for RED gateways of [15]

   In case of Congestion Condition,
      Notify feedback controller
END REPEAT every packet arrival
```

그림 3. Core 라우터의 큐 혼잡 상황 판단 알고리즘

되어있다. Ingress 라우터의 전송 속도 조절 알고리즘은 그림 4와 같다. Core 라우터로부터 혼잡 상황 통보 패킷을 받은 Ingress 라우터는 네트워크 중심부의 혼잡 상황을 파악할 수 있고, 네트워크 중심부로 유입시키는 트래픽의 속도를 IPG(Inter-packet Gap)를 조정하여 혼잡 상황에 대처한다.

Ingress 라우터는 호스트 단에 혼잡 상황을 통보하기 위하여 Core 라우터로부터 전송받은 혼잡 상황 통보 패킷의 헤더 정보를 확인하여 해당 호스트 단을 확인할 수 있으며, 해당 호스트로 직접 혼잡 상황을 통보할 수 있다. 이를 이용하여 호스트 단의 빠른 혼잡 상황 대처를 이끌어낸다.

Ingress 라우터에서 감소한 전송 속도는 일정 기간 w마다 복구된다. w는 Ingress 라우터와 Egress 라우터 사이의 전달 시간에 비례한다. Ingress 라우터는 Edge 라우터와의 전달 시간을 측정하기 위한 일정 기간마다 Egress 라우터로 피드백 요청(Feedback Request) 패킷을 보낸다.

피드백 요청 패킷은 Ingress 라우터에서 CU 필드를 01로 설정하여 전송하게 되고 피드백 요청 패킷의

```
1. Increase transmission rate
REPEAT for period w
   If( IPG > IPG_threshold of Edge Router )
      // default IPG_threshold = tx_time of small packet
      // IPG_threshold : proportion to (1/sum of input port)
      decrease IPG using (1) of [3], (here, C = w)
   Else
      IPG set to 0                          // Maximum Tx_rate
END REPEAT for period w

2. Decrease transmission rate
IF congestion notification packet arrival
   increase IPG by IPG *= β
   // β = 1+ (1/sum of input port)
   // for example, if sum of input port is 1, β = 2
```

그림 4. Ingress 라우터의 전송 속도 조절 알고리즘

1. Increase transmission rate
REPEAT for period *SRTT*
  decrease *IPG* using (1) of [3], (here, *C* = *SRTT*)
END REPEAT for period *w*

2. Decrease transmission rate
IF congestion state
  increase *IPG* using (3) of [3]

그림 5. 호스트 단의 혼잡 상황 제어 알고리즘

전송 시각을 유지한다. 해당 패킷을 받은 Egress 라우터는 CU 필드를 10으로 설정하여 Ingress 라우터로 피드백 응답(Feedback Acknowledgement) 패킷을 보낸다. 피드백 응답 패킷을 받은 Ingress 라우터는 Ingress 라우터와 Egress 라우터 사이의 전달 시간을 측정하여 w의 값을 측정한다. 본 논문에서는 w의 값으로 Edge 라우터 사이의 패킷 왕복시간을 사용한다.

2. 호스트 단의 혼잡 제어 알고리즘

호스트 단은 멀티미디어 스트림 데이터를 보내기 위해서 UDP 기반에서 응용계층의 혼잡 제어 알고리즘을 사용한다. 이 알고리즘은 RAP의 변형으로서 DiffServ 망으로부터 혼잡 상황에 대한 피드백 정보를 직접 받아서 전송한 패킷에 대한 응답에 의한 혼잡 상황 대처 시간보다 혼잡 상황을 빠르게 대처하도록 한다.

호스트 단의 전송 속도 조절 알고리즘은 그림 5와 같다. 호스트 단에서 예측하는 네트워크의 혼잡 상황은 타임아웃, 응답에 의한 패킷 손실 확인과 네트워크로부터의 혼잡 상황 피드백의 세 가지를 통하여 이루어진다. 호스트 단은 Ingress 라우터로부터 혼잡 상황 피드백을 수신한 경우 전송 속도를 감소한다. 타임아웃과 응답 패킷에 의한 혼잡 상황은 기존의 RAP의 알고리즘과 동일하게 수행된다.

제안하는 혼잡 제어 알고리즘은 중심부 라우터와의 유기적인 조화를 통하여 기존의 단대단 혼잡 제어 알고리즘의 단점인 긴 RTT로 인한 상대적으로 느린 혼잡 제어를 극복할 수 있다.

Ⅳ. 성능평가 및 비교분석

실험 토폴로지는 UDP 기반의 송신측 기반의 혼잡 제어 알고리즘의 DiffServ 망과의 유기적인 연계를 보기 위하여 그림 6과 같이 간단히 구성된다. 시뮬레이션 모델은 C++를 이용하여 구현하였으며 이 토폴로지에 기초하여 Default RAP, Router-based Rate Control, Host-based Rate Control의 세 알고리즘을 비교 및 분석한다. 각 알고리즘의 동작은 다음과 같다.

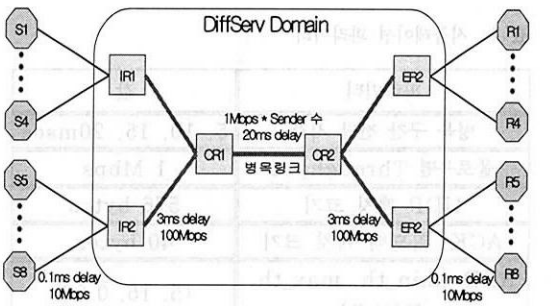

그림 6. 실험 토폴로지

Default RAP 알고리즘은 DiffServ 망에서는 별도의 혼잡 제어 알고리즘이 수행되지 않고 호스트 단에서만 혼잡 제어 알고리즘을 수행한다. 이 알고리즘은 RAP의 혼잡 제어 알고리즘을 따른다.

Router-based Rate Control 알고리즘은 네트워크 중심부의 혼잡 상황에 대해 Edge 라우터에서 네트워크 중심부로 유입되는 전체 트래픽을 제어한다. 호스트 단에서는 이와 별개로 Default RAP 알고리즘이 수행된다.

제안하는 Host-based Rate Control 알고리즘(Early Rate Adaptation Protocol: ERAP)은 Core 라우터로부터 혼잡 상황 통보 패킷을 받은 Ingress 라우터가 해당 호스트 단에 혼잡 상황을 알려주고, 혼잡 상황을 전달 받은 호스트 단은 빠른 혼잡 제어를 통하여 혼잡 에 대응한다.

1. 실험 토폴로지 및 파라미터

실험에서 S1~S8은 UDP 기반의 이 논문에서 제안한 송신측 기반 혼잡 제어 알고리즘을 사용한다. S1과 S5, S2와 S6, S3와 S7, S4와 S8의 서비스 클래스는 각각 AF1, AF2, AF3, AF4로 설정하였다.

표 1에는 시뮬레이션에 사용된 여러 파라미터들의 값을 보여주고 있다. 실험에서 사용된 DiffServ 망은 기본적으로 RED 알고리즘과 DWRR(Deficit Weighted Round Robin) 큐 스케줄러를 사용한다. 큐 스케줄링에 사용된 가중치는 모든 플로우가 약 1Mbps의 Throughput을 갖도록 AF1:AF2:AF3:AF4=1:1:1:1로 설정하였다.

실험 토폴로지에서는 CR1과 CR2 사이의 구간이 작은 대역폭을 갖도록 설정되어 병목 구간이 되고, CR1은 병목 지점이 되어 혼잡 상황을 야기하게 된다.

표 1. 시뮬레이션 파라미터

파라미터	값
병목 구간 전달 시간	5, 10, 15, 20msec
플로우별 Throughput	1 Mbps
UDP 패킷 크기	576 bytes
ACK, 피드백 패킷 크기	40 bytes
RED (min_th, max_th, max_p)	(5, 15, 0.02)
wq	0.002

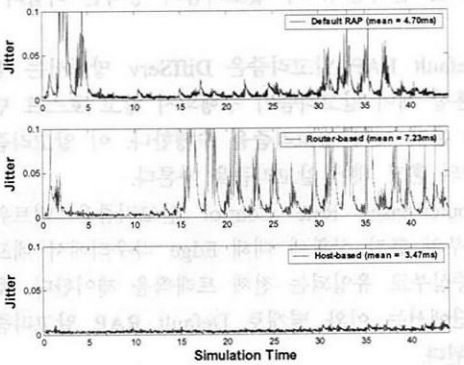

그림 7. R1에서의 알고리즘별 지터 비교

2. 시뮬레이션 결과

앞서 설명한 시뮬레이션 환경에서의 혼잡 상황 제어 결과를 CR1과 CR2의 전달 시간의 증가 추이에 따라서 비교 및 분석하였다. 시뮬레이션에서 CR1과 CR2의 전달 시간은 기본적으로 20msec 값을 갖으며, 모든 플로우가 동시에 전송을 시작한다.

그림 7에서는 CR1과 CR2 사이의 전달 시간이 20msec인 경우에 R1에서의 알고리즘별 지터 변화를 볼 수 있다. 그림에서 보듯이 제안된 알고리즘으로 호스트 단에서 혼잡 제어를 수행하는 경우에 네트워크 중심부의 혼잡 상황을 빨리 파악할 수 있어서 멀티미디어 QoS의 중요한 요소인 지터 값의 변화가 적다는 것을 볼 수 있다.

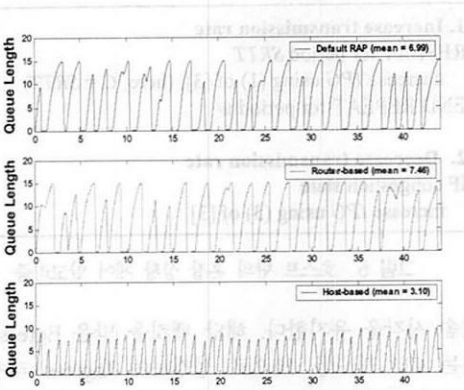

그림 8. 알고리즘 별 RED 평균 큐의 길이

또한 표시된 지터의 평균치를 확인해보면 호스트 단의 혼잡 제어는 기존의 RAP 알고리즘과 라우터에서의 혼잡 제어보다 각각 평균적으로 약 74%, 48% 정도 줄어드는 것을 확인할 수 있다. 라우터에서 전체 트래픽의 흐름을 조절하는 경우에는 오히려 혼잡 상황에 직접적인 영향을 주지 않은 트래픽까지 전송이 늦어지는 경향이 발생할 수 있어서 지터 측면에서는 좋지 않음을 알 수 있다.

그림 8에서는 CR1과 CR2 사이의 전달 시간이 20msec인 경우에 AF1 클래스의 RED 평균 큐의 길이 변화를 볼 수 있다. 그림에서 보듯이 호스트 단에서 빠른 혼잡 제어를 하는 경우가 다른 경우보다 각

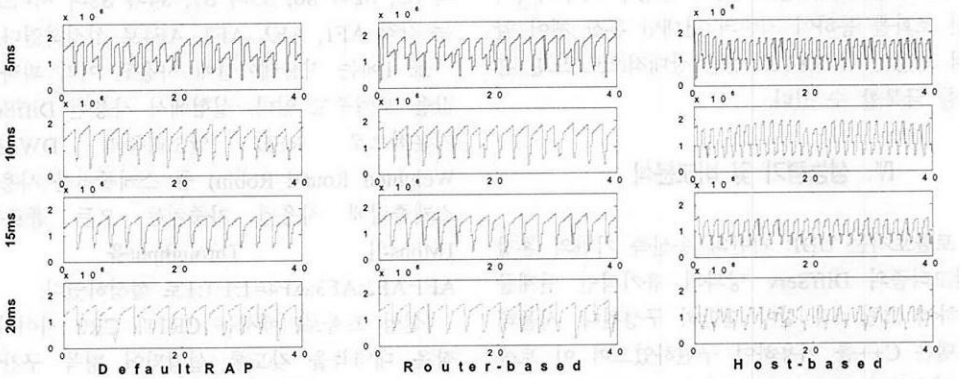

그림 9. 병목 지점에서의 전달 시간 증가에 따른 알고리즘 별 전송 속도 조절 변화 추이

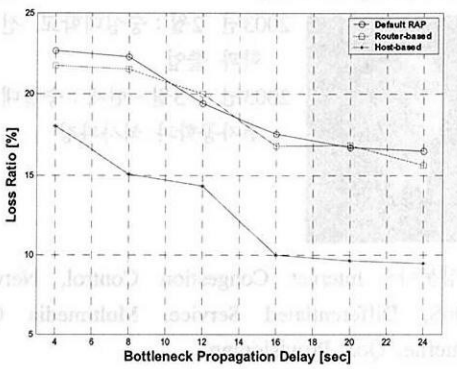

그림 10. 병목 구간 전달 시간 증가로 인한 패킷 손실률 변화

각 44%, 41% 더 적은 평균 길이를 갖는 것을 볼 수 있다. 이를 통해 호스트 단의 빠른 혼잡 제어가 큐를 더 적게 더 빈번히 이용하여 그 효율이 높음을 알 수 있다.

그림 9에서는 CR1과 CR2 사이의 전달 시간이 각각 5msec, 10msec, 15msec, 20msec인 경우에 S1에서의 알고리즘별 패킷 전송 속도의 변화를 볼 수 있다. 호스트 단에서 빠른 혼잡 제어를 하는 경우가 다른 알고리즘보다 전송 속도의 최대치와 최소치의 차이가 상대적으로 적으며 평균치에 근접해서 움직임을 볼 수 있다.

그림 10은 병목 구간의 전달 시간(RTT)이 증가함에 따라 각 알고리즘의 패킷 손실률의 변화를 나타낸 것이다. 호스트 단에서의 빠른 혼잡 제어 알고리즘은 네트워크의 혼잡을 야기하면서 Drop될 패킷 전송을 줄여주어 혼잡 상황 조절 성능이 뛰어남을 알 수 있다. 호스트 단에서의 빠른 혼잡 제어는 패킷 손실률은 줄이면서 기존의 알고리즘과 같은 Throughput을 보임으로서 한정된 네트워크 자원을 효율적으로 이용함을 알 수 있다.

V. 결 론

본 연구에서는 DiffServ 망을 이용한 멀티미디어 스트리밍 데이터의 전송을 위한 혼잡 제어 알고리즘에 대하여 호스트 단과 네트워크 내부와의 유기적인 연계를 통한 빠른 혼잡 제어 알고리즘을 제안하였다.

기존의 단대단 혼잡 제어 알고리즘은 긴 RTT를 갖는 플로우가 상대적으로 작은 RTT를 갖는 플로우에 비해 혼잡 상황을 늦게 판단하여 빠른 혼잡 제어를 하지 못하는 단점을 갖고 있었다.

이를 극복하기 위하여 네트워크의 병목 지점에서의 혼잡 상황 발생을 알려주는 피드백 정보를 이용하여 혼잡 상황을 야기한 개별 플로우에 대한 빠른 혼잡 제어를 통하여 패킷 손실, 네트워크 자원의 효율적인 이용, 멀티미디어 데이터에 대한 지터 값 등의 QoS 요인 성능이 좋아지는 것을 확인하였다.

본 논문에서 제안하는 알고리즘은 점점 증가하는 멀티미디어 데이터의 전송에 있어서 안정적인 전송을 보장하는데 그 이용 가치가 크다고 할 수 있다.

참 고 문 헌

[1] Xipeng Xiao, et al., "Internet QoS: A big picture", IEEE Network, March/April 1999.

[2] "The Need for QoS - A White Paper", URL:http://www.qosforum.com

[3] R.Rejaie, M.Handley, and D.Estrin "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet" Proc. IEEE Infocom'99,pp. 1337-1345, Mar. 1999.

[4] S. Y. Cheung, M Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution" in Proc. IEEE INFOCOM'96, pp. 553-560, Mar. 1996

[5] Barden, R., Zhang, L., Berson, S., Herzog, S. and Jamin S., "Resource Reservation Protocol (RSVP) Version 1 Functional Specification", RFC 2205, Proposed Standard September 1997

[6] Jamin, S., et al,. "Comparaison of measurement-based admission control algorithms for controlled-load service", Proc. IEEE INFOCOM 97, April 1997.

[7] Barden, R., Clark, D. and Shenker, S., "Integrated Services in the Internet Architecture: and Overview", Internet RFC 1633, June 1994.

[8] S.Blake, et al. "An Architecture for Differentiated Service", RFC 2475, December 1998.

[9] Bernet, Y, et al., "A Framework for Differentiated Service", draft-ietf-Diffserv-framework-02.txt, February

1999.
[10] Nicolas. K, et al., "A Two-bit Differentiated Services Architecture for the Internet", RFC2638, July 1999.
[11] Nichols, Kathleen, et al., "Definition of the Differentiated Services Field(DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
[12] Dapeng Wu, Yiwei Thomas Hou, Ya-Qin Zhang, "Transporting Real-Time Video over Internet: Challenges and Approaches", Proceedings of the IEEE, 88, pp. 1855-1875, Dec. 2000
[13] Dapeng Wu, Yiwei Thomas Hou, Wenwu ZAhu, Ya-Qin Zhang, Jon M. Peha, "Streaming Video over the Internet: Approaches and Directions", IEEE Transaction On Circuits And System For Video Technology, 11, pp. 282-300, Mar. 2001
[14] V. Jacobson, "Congestion Avoidance and Control," Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.
[15] S. Flyd, V. Facobson, "Random Early Detection gateways for congestion avoidance", IEEE/ACM Trans. on Networking, Aug. 1993.
[16] Bin Pang, Wen Gao, "An Edge-to-edge Congestion Control Scheme for Assured Forwarding", Proceedings of IEEE International Conference on Networks (ICON), Singapore, August 2002.
[17] Bin Pang, Wen Gao, "Two Phase Distributed Congestion Management for Differentiated Services Network", Proceedings of 5th IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC), Jeju, Korea, July 2002.

박 종 훈(Jonghun Park) 준회원

2003년 2월 : 숭실대학교 전자공학과 졸업
2003년 3월~현재 : 숭실대학교 전자공학과 석사과정

<관심분야> Internet Congestion Control, Network QoS, Differentiated Service, Multimedia QoS, Internet QoS Provisioning

유 명 식(Myungsik Yoo) 정회원

1989년 2월 : 고려대학교 전자전산공학과 졸업
1991년 2월 : 고려대학교 전자공학과 석사
2000년 1월~8월 : Dept. of Electrical Engineering, SUNY at Buffalo 박사
2000년 9월~현재 : 숭실대학교 정보통신전자공학부 조교수

<관심분야> 광네트워크, OBS, EPON, 네트워크 QoS, 혼잡 제어 알고리즘, Wireless TCP, Wireless Link Protocol