2002년도 대한전자공학회/대한전기학회 시스템 및 제어분야 합동 추계학술대회 논문집(2002.11.30)

Different Service QoS 구현 및 정책 수립 연구

Piao Lin, 최 민 석, 이 종 민, 김 용 득
아주대학교 전자공학부
전화 : 031-219-2372

A Study On The Implementation of Different Service QoS And Establishing The Policy

Piao Lin, Min-Seok Choi, Jong-Min Lee, Yong-Deak Kim
Dept. of Electronics Engineering, Ajou University
E-mail : tuscani@comnet.ajou.ac.kr

Abstract

IP can be operated on virtually any network transmission media and used to communicate each other whatever system platforms are. But, with the enormous growth of the Internet in the past a few years, the amount of network traffic has increased as the number of users and applications has increased. New applications have new service demands for fulfilling user's satisfaction, and as a result the Internet needs to change as well. This paper shows that why the Internet must change to satisfy the need for QoS to accommodate these and performance of network using QoS establishing 5 Scheme has a dominant improvement through a simulator.

Ⅰ. 서론

지난 몇 년간 인터넷의 괄목할만한 성장과 더불어 여러 종류의 인터넷 응용프로그램이 개발됨에 따라 트래픽이 증가추세에 있다. 기존의 인터넷은 최선 노력 서비스 프로토콜인 IP에 기반하여 구축되어 왔지만 증가하고 있는 실시간 멀티미디어 서비스를 위해서는 네트워크의 고지능형으로의 변화가 불가피하다. QoS는 네트워크를 이루는 구성 요소로 하여금 어느 정도의 보장성을 부여하여 트래픽이나 서비스 요구사항이 충족되도록 하는 것이다. 본 논문에서는 Diffserv QoS 방식 요소를 시뮬레이터를 통해 구현하여 정량적으로 결과를 분석, 정책을 제시함으로써 양단간 QoS보장을 보이고자 한다.

Ⅱ. QoS의 유형

큰 범주로 볼때 QoS는 두가지로 나눌 수 있는데, Intserv방식과 Diffserv 방식이다. Intserv 방식에서 네트워크 리소스는 각 응용프로그램의 QoS 요구사항에 맞추어 할당되고 대역폭 관리 정책의 영향을 받는다. Diffserv방식에서는 네트워크 트래픽을 몇가지 범주로 나누어 정책에 따라 범주별로 리소스를 할당한다. 즉, 요구사항이 많은 응용프로그램을 더 우선적으로 처리한다. 이들 QoS 프로토콜과 알고리즘들은 상호보완적으로 작용한다. QoS를 제공하기 위해서는 일단 지연을 최소화 해야 하고 지연이 있다고 하더라도 지연변화를 최소화해야 하며, 또한 지속적이고 꾸준한 투과성을 제공해야만 한다. IP 성능 매트릭은 크게 대역폭, 지연, 손실, 지연변이 그리고 연결성으로 구분할 수 있으며, 이는 현재 네트워크망의 특성상 단방향을 고려하지만 향후 상호 동작형 실시간 반응을 고려하여 양방향 특성을 고려해야 한다.

QoS 관련 핵심 메커니즘에는 Admission Control(요구받은 연결을 허락할지의 유무 결정), 패킷 분류 및 표시(QoS 가능한 형태로 조작), 우선 순위와 스케줄링 메커니즘(트래픽에 대한 서로 다른 지연 제공), Traffic Shaping/Conditioning(트래픽 프로파일 부합 유무), 큐잉(혼잡상황하에서 패킷폐기여부 결정), 혼잡제어(RED, ECN)등이 있다. 본 논문에서는 RED를 코어 라우터에서

사용하여 혼잡을 회피하고 Round Robin방식으로 스케줄링하고 있다.

Ⅲ. Differentiated Service 방식의 QoS

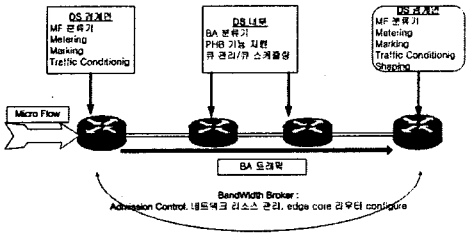

그림 1 Diffserv 방식에서의 각 노드 주요 기능

Diffserv 아키텍쳐는 IETF에서 명시하여 제공한 프레임 워크이며, 이 프레임 워크 안에서 서비스 공급자는 각 사용자에게 성능에 기반을 둔 차별화된 서비스를 제공한다. 이 모델에서 네트워크로 유입되는 모든 트래픽은 네트워크 경계부분에서 분류되고 심사를 받는다. 즉, 정교한 분류절차와 표시, Policing, Shaping 동작이 네트워크 경계나 호스트에서 구현된다. Diffserv 프레임 워크의 중요한 특징은 확장성으로 매우 큰 네트워크에도 적용가능하다. 이것은 패킷단위의 흐름이 아닌 클래스별로 묶여진 트래픽을 제공함으로써 가능하다.

3.1 Diffserv 방식의 구성요소

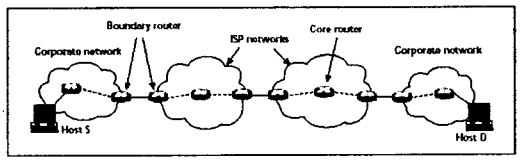

그림 2 End to End 간 Diffserv QoS 구성장치

Diffserv 아키텍처는 다음 네 가지 구성요소를 정의함으로써 가능하다.

(1) Service

네트워크 내에 하나 또는 그 이상의 경로를 통과하는 단방향 패킷 전송의 중요한 특성을 정의한다.

(2) 조건검사 함수와 PHB

호스트나 사용자 접근 계층 라우터, 경계 라우터에서 생기는 트래픽 제어 기능들을 보통 트래픽 조건부합검사라고 하며, PHB는 Diffserv 호환 노드에서 Diffserv Behavior 클래스별 묶음에서 적용되는 포워딩 Behavior를 참고한다.

(3) DS Code Point

IP 헤더내의 DS 필드중 현재 사용 중인 6비트를 DSCP로 사용하며 이는 각 노드에서 패킷이 경험할 PHB를 선택하는 기능을 한다.

(4) PHB를 하기 위한 메커니즘

서비스 정책 제공에 중요한 처리특성으로 정의되며 약간의 버퍼 관리와 패킷 스케줄링 메커니즘으로 코어노드에서 구현된다.

3.2 Diffserv 의 동작 메커니즘

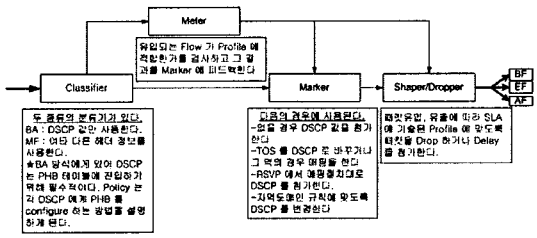

그림 3 Edge 라우터에서의 Diffserv제공 오브젝트 기능

(1) 분류기

입력되는 패킷을 클래스로 분리해 낸다. 분류기는 동일 클래스 묶음행위를 하는 데 있어 IP 패킷 헤더에 있는 DS 영역을 사용하여 패킷의 우선순위를 결정한다.

(2) 패킷 조작 구성요소

표시기, 절대적 폐기장치, 멀티플렉서, 계수기등 4개의 서로 다른 구성요소들이 트래픽 흐름에 적용될 수 있다.

(3) 큐

들어오는 패킷을 버퍼링하여 메모리에 저장한다.

(4) 메터

분류기에 의해 선택된 트래픽 흐름을 트래픽 제어 명시사항 안에 정의되어 있는 프로파일내용과 비교한다.

(5) 알고리즘적 패킷폐기장치

알고리즘을 사용하여 혼잡상황을 예측하고 패킷 폐기 여부를 결정한다.

Ⅳ. Diffserv QoS 구현을 위한 테스트벤치

4.1 Network Simulator

IBNL(Lawrence Berkeley Simulator)의 네트워크 연구그룹에 의해 개발된 Ns-1이라는 시뮬레이션 도구는 tcl을 기술 언어로 사용한다. MIT에 의해 개발된 Otcl을 사용하여 새로운 구조를 갖도록 Ns-1을 개선한 것이 Ns-2 이다. Ns는 Tcp, 라우팅 프로토콜, 멀티캐스트 프로토콜 Rtp(Real Time Protocol), SRM(Scalable Reliable Multicast) 등 다양한 인터넷 프로토콜에 대한 시뮬레이션을 수행하기에 적절한 여러 환경을 제공하는 현재 널리 사용되고 있는 네트워크 시뮬레이션 도구이

다. Ns는 객체지향형 tcl 스크립트 해석기로서, 아래 그림은 Ns의 구성을 보여준다.

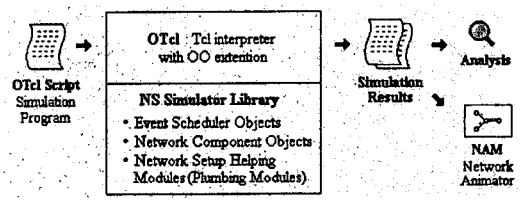

그림 4 NS 동작 메카니즘

시뮬레이션 이벤트 스케쥴러는 시뮬레이션 및 셋업 하기 위해 Otcl Script 언어로 초기화해야 하며, 네트워크 콤포넌트 오브젝트 라이브러리와 네트워크 셋업 모듈 라이브러리를 사용하여 네트워크 토폴로지를 형성한다.

4.2 파라미터 제어형 dsRED 알고리즘

본 논문에서 Diffserv모듈은 현재 4개의 클래스로 구현되어 있으며 각각의 클래스는 3단계의 폐기 우선순위를 갖는다. 이들 폐기 우선순위는 각 클래스 내에서 트래픽을 차별화하도록 해주며, 하나의 트래픽 클래스는 대응되는 물리적 RED 큐에 큐잉되고 여기에 다시 3개의 가상적인 큐가 있게 되는 셈이다. Diffserv RED 파라미터들은 가상 큐에 사용되며 이는 하나의 가상 큐로부터 패킷을 다른 가상 큐에 있는 패킷보다 더 자주 drop시키는데 사용된다. Ns 시뮬레이션 환경은 Policy, Edge 라우터, Core 라우터의 3가지 구성요소를 갖는다. Diffserv를 이루는 각 오브젝트는 다음과 같이 구현된다.

(1) Diffserv를 위한 RED 큐

Ns에서 Diffserv의 기능은 큐 오브젝트가 대부분을 차지한다. 이 오브젝트는 droptail이나 CBQ, RED 같은 큐 형태의 대체품으로 간주된다. Diffserv의 큐는 dsREDQueue로서 이는 Queue 베이스 클래스에서 파생된다.

(2) 코어 라우터와 경계 라우터 들

Diffserv Edge 라우터와 Core 라우터들은 EdgeQueue와 CoreQueue 클래스에 정의되어 있다. 이들 클래스는 둘다 dsREDQueue에서 파생된다.

아래의 그림 5는 RED Queue 메커니즘을 사용한 경우와 사용하지 않은 경우 노드에서 생기는 현상을 직관적으로 보여준다. 큐 관리 방식을 사용하지 않을 경우 패킷이 한꺼번에 몰리거나 폐기되는 문제점이 생기게 된다. 이를 방지하기 위해서는 혼잡상황 예견능력이 필요하며 예측시 패킷을 폐기 혹은 반향 시킴으로써 혼잡을 피하는 방법이 강구되어야 한다. 본 논문에서는 Core 라우터에서의 PHB로 혼잡 회피방식인 RED 큐 관리 방식을 사용하여 어느 정도 큐 버퍼가 차게 되면 임의 패킷을 폐기시켜 소스측이 ACK를 받지 못하게 하여 윈도우 크기를 줄이게 함으로써 혼잡상황을 피해가고 있다. 하지만 본 논문은 TCP가 아닌 UDP를 사용하여 큐 관리방식에서는 효과를 기대할 수 없고 대신 큐 스케쥴링 메커니즘으로 WRR을 사용함으로써 Policy 에 해당하는 Weight에 따라 큐를 Out Port로 보내게 된다.

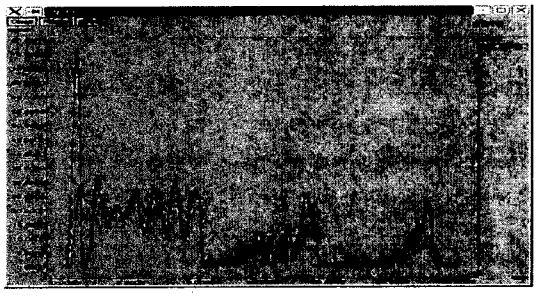

그림 5 큐 관리 방식 없는 노드에서의 큐 크기변화와 평균 큐 크기의 변화

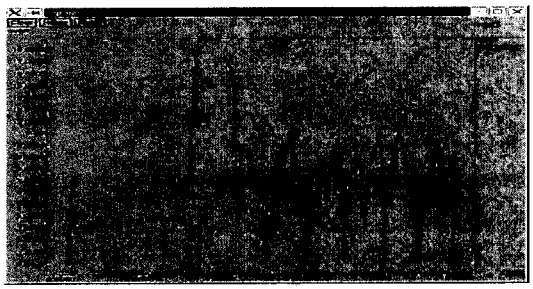

그림 6 RED 큐 관리방식을 사용하는 노드에서의 큐 크기변화와 평균 큐 크기의 변화

그림 6은 RED 방식의 큐 관리 방식을 사용한 경우의 큐 그래프이다. 일단 큐 크기가 시간이 지남에 따라 평균적으로 항상성을 유지하게 되었으며 동기 문제를 해결하여 톱니파의 그래프가 사라지고 평균 큐의 크기가 유지됨으로 인해 시스템의 자원이용도가 향상되는 걸 볼 수 있다.

V. 정책기반형 Diffserv 방식모델 제안

5.1 정책 모델 제시

현재 다섯 개의 정책 모델이 구현되어 있다.

(1) TSW2CM : CIR과 두 개의 폐기 우선순위를 사용

(2) TSW3CM : CIR과 PIR, 세 개의 폐기 우선순위 사용

(3) Token Bucket : CIR과 CBS, 그리고 두 개의 폐기 우선순위 사용

(4) Single Rate Three Color Marker : CIR과 CBS, 그리고 EBS를 사용하여 세 개의 폐기 우선순위 사용

(5) Two Rate Three Color Marker : CIR과 CBS, PIR, 그리고 PBS를 사용하여 세 개의 폐기 우선순위 사용

5.2 정책 모델 구현

Policy 클래스는 EdgeQueue 클래스에서 사용되며 모든 정책 기능을 다루게 된다. 정책은 소스측과 목적지사이에서 설정되며 그 사이의 모든 흐름들은 단일한 형태를 이루는 종류별로 묶여진 형태가 된다. 정책은 Policer 형태, 목적지까지의 전송률, 그 외 Policer 특정 파라미터를 정의하고 있다. 각 트래픽 종류별 묶음은 연관있는 Policer 형태, 메터 형태, 초기 Code Point를 갖는다.

VI. 정책모델 성능 평가

다음은 시뮬레이션에 사용된 네트워크 토폴로지이다.

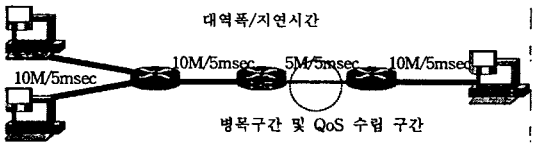

그림 7 시뮬레이션에 사용된 토폴로지

소스 2개와 목적지 1개가 있으며 각각 경계 라우터 한 개와 연결되어 있고 이 경계 라우터 들은 코어 라우터에 연결되어 있다. 정책 메커니즘을 사용하지 않을 경우의 시뮬레이션 결과는 다음과 같다.

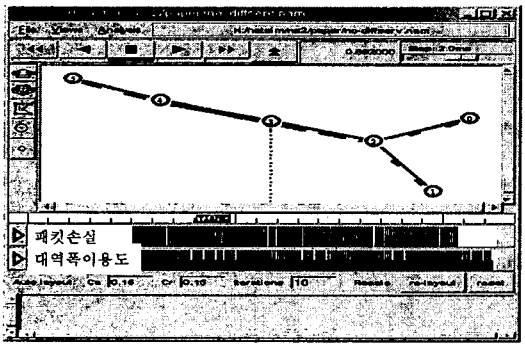

그림 8 정책 없을 경우의 트래픽 양상

혼잡이 발생한 시점부터 폐기되는 패킷은 소스에 관계없이 계속 발생하게 된다. 아래의 표는 각 정책 모델별로 시뮬레이터를 사용하여 얻은 통과 패킷 수와 폐기 패킷 수, 평균 대역 효율등을 비교한 것이다.

	통과패킷수	링크사용률	패킷통과 개선률	대역폭 개선률
No-Policy	645	0.4769	1.93배 개선	2.06배 개선
TB	1274	0.9942		
tsw2cm	1261	0.9957		
tsw3cm	1224	0.9727		
srtcm	1230	0.9764		
trtcm	1236	0.9838		

표 1 정책 적용 모델과 비적용 모델의 효율 비교

표 1에서 알 수 있듯이 정책모델을 적용한 경우는 적용하지 않았을 때보다 혼잡상황의 Core 라우터를 통과한 패킷수나 링크 대역폭 사용률이 두 배가량 증가했음을 알 수 있다. 이는 정책을 통한 각 노드별 패킷분류와 Metering에 의한 Marking으로 시스템의 자원을 최대한 이용할 수 있게 함으로써 결과적으로 통신 양단간 QoS를 증가시킴을 증명하는 것이다. 하지만, 여러 가지의 경우에 있어서 최대효과를 볼 수 있는 정책은 모두 상이하므로 해당상황 하에서 최적 정책을 부합시키는 관리 서버가 필요하다.

VII. 결론

현재 거의 모든 인터넷 서비스는 IP 네트워크를 통하여 이루어지고 있다. 하지만 그 설계이론으로는 지금의 일시적 과부하로 인한 QoS 문제점을 해결할 수 없다. QoS를 해결할 궁극적인 방법은 통신이 일어나는 모든 구간에서 일관된 정책으로 QoS를 감시 및 제어하는 것이다. 가까운 미래에는 향상된 물리망과 함께 상황별 정책을 융통성 있게 적용할 수 있는 정책 전담서버가 등장할 것이다. 본 논문에서는 Diffserv방식을 사용하는 5가지 모델을 사용하여 UDP 스트림의 경우 품질의 개선도를 각 정책이 사용하는 파라미터 별로 분석하여 증명하고자 하였다. 적용된 모델은 적용하지 않은 모델보다 약 200%의 성능 개선을 보여주고 있다. 각 정책의 시뮬레이션의 결과를 보면 특정 상황에서 QoS를 최대한 보장해 줄 수 있는 정책은 범용성을 가질수 없고 특수성과 유일성을 가짐을 알 수 있다. 이는 정책서버가 혼잡상황에 따라 정책테이블에서 최적 정책을 선택하고 도중의 망의 환경변화시 정책도 바뀌어야 함을 뜻한다. 따라서 향후의 연구는 지능적인 정책 전담 서버의 통신 프로토콜 개발과 Case별 정책 수립으로 모든 경우에 해당되는 최적의 정책을 작성하는 데 초점을 맞추어야 할 것이다.

참고문헌

[1] Srnivas Vegesna, " IP Quality Of Service", Cisco Press

[2] William Stalling, "High Speed Network", Prentice Hall, 2000

[3] K. Cho. "A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers" USENIX 1998 Annual Technical Conference, June 1998

[4] RFC 2474, 2475, 2597, 2598, 2983

[5] Ns class 정리
http://www-sop.inria.fr/redeo/personnel/Antoine.Clerget/