논문 03-28-9B-8 한국통신학회논문지 '03-9 Vol.28 No.9B

실시간 인터넷 서비스를 위한 상태 의존 RED 및 동적 스케줄링 기법에 관한 연구

정회원 유 인 태*, 홍 인 기*, 서 덕 영*

A Study on State Dependent RED and Dynamic Scheduling Scheme for Real-time Internet Service

In-Tae Ryoo*, Een-Kee Hong*, Doug-Young Suh* ***Regular Members***

요 약

인터넷에서 실시간 서비스를 지원하기 위해서는 큐 관리 및 스케줄링 기법이 고도화되어야 한다. 기존의 큐 관리 기법은 TCP 플로우의 혼잡을 제어하는 기능은 제공하고 있으나 실시간 서비스 지원에는 문제점을 가지고 있다. 즉, 큐가 어느 정도 점유된 이후 계속적인 버스트 성의 트래픽이 발생할 경우 성능이 급격히 떨어지고, 특히 큰 지터 값으로 인해 실시간 전송에 부적절하다. 본 논문에서는 최대 임계값과 큐 가중치 등의 RED 파라미터를 큐 상태에 따라 적응적으로 조정하여 실시간 서비스에 대한 지연 및 지터 성능을 향상시킬 수 있는 상태 의존 RED (SDRED; State Dependent Random Early Detection) 큐 관리 및 동적 스케줄링 기법을 제안한다. 제안한 기법은 큐 상태에 따라 RED의 최대 임계값과 큐 가중치를 네 개의 서로 다른 레벨로 변경하여 트래픽 상황에 적응하도록 설계하였으며, 시뮬레이션을 통하여 SDRED가 평균 큐 크기를 가지는 RED나 평균 큐 크기를 가지지 않고 현재의 큐 크기를 사용하여 혼잡을 제어하는 BLUE, 그리고 RED 게이트웨이의 수율이나 지연 특성을 향상하기 위해 제안된 ARED 및 DSRED에 비해 낮은 지연 값과 안정적인 지터 값을 나타냄을 검증하였다.

Key Words : Real-time Internet Service, Queue Management, Scheduling, RED, State Dependent RED

ABSTRACT

To satisfy the requirements of the real-time Internet services, queue management and scheduling schemes should be enhanced to accommodate the delay and jitter characteristic of them. Although the existing queue management schemes can address the congestion problems of TCP flows, they have some problems in supporting real-time services. That is, they show performance degradation when burst traffics are continuously going into the system after the queue is occupied at a predefined threshold level. In addition, under the congestion state, they show large jitter, which is not a desirable phenomenon for real-time transmissions. To resolve these problems, we propose a SDRED (State Dependent Random Early Detection) and dynamic scheduling scheme that can improve delay and jitter performances by adjusting RED parameters such as max_{th} and w_q according to the queue status. The SDRED is designed to adapt to the current traffic situation by adjusting the max_{th} and w_q to four different levels. From the simulation results, we show that the SDRED decreases packet delays in a queue and has more stable jitter characteristics than the existing RED, BLUE, ARED and DSRED schemes.

* 경희대학교 전자정보대학 (itryoo@khu.ac.kr, ekhong@khu.ac.kr, suh@khu.ac.kr)
논문번호 : 020531-1209, 접수일자 : 2002년 12월 11일

※ 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음. (KRF-2002-003-D00293)

I. 서 론

인터넷 서비스에 대한 요구가 다양화됨에 따라 현재 인터넷은 매우 혼잡해지고, 이는 네트워크 전체의 성능 저하 및 사용자의 불만을 유발시킨다. 이러한 문제를 해결하기 위하여 물리적으로 대역폭을 증가시키고도 있으나 이와 함께 QoS 제공 기법에 의한 인터넷 자원의 효과적 사용을 유도할 필요가 있다. IETF에서는 혼잡 발생 시 매우 큰 패킷 손실이 발생하는 것을 해결하기 위하여 RED (Random Early Detection)와 같은 AQM (active queue management) 메커니즘을 전개하였으며[1], 적절한 자원 할당을 위해 지금까지 여러 가지 큐 관리 및 스케줄링 기법이 사용되어 왔다. 혼잡이 발생한 상태에서는 큐 길이가 거의 최대치에 도달하고 이후에 도착하는 모든 패킷은 폐기되므로 해당 패킷을 전송한 모든 송신 호스트들은 동시에 윈도우 크기를 줄이는 slow start 단계에 접어들어 일시적으로 링크 사용 율이 떨어지는 global synchronization 현상이 발생하게 된다[2][3].

이러한 문제를 해결하기 위한 RED를 변형한 여러 가지 방법들이 제시되어 있다. 그러나 RED에서 평균 큐 크기를 유지하는 것은 상대적으로 느린 반응을 나타내게 한다. 따라서 RED나 이의 변형된 방법들과 달리 평균 큐 크기를 유지하지 않고 현재의 큐 크기를 시간을 기반으로 하여 체증을 제어하는 BLUE나 현재의 라우터나 스위치까지의 누적된 혼잡도를 측정하여 이를 반영하는 REM (Random Exponential Marking)[4]과 같은 방법들이 나오게 되었다.

한편 RED를 다양한 트래픽 상황에서 적용할 경우 발생하는 문제점을 지적하고 이를 개선하기 위한 방법들이 많이 제시되었고 이러한 제안들은 서로 다른 특성을 가지는 트래픽에 대해 같은 패킷의 폐기확률을 적용할 경우 발생되는 링크 사용의 공평성이 유지되지 않는다는 것과 고정된 패킷의 폐기확률을 유지했을 때 많은 트래픽이 한꺼번에 유입이 되었을 때 혼잡의 제어가 제대로 되지 않는다는 것에 초점이 맞추어져 있다.

RED에 의한 혼잡 제어는 큐 오버플로우에 의한 패킷 폐기, RED 평균 큐 크기 값 avg_q가 최소 임계값 Min_{th}와 최대 임계값 Max_{th} 사이에 있을 경우 최대 패킷 폐기 율 Max_p의 확률 값 이내에서의 패킷 폐기, 그리고 avg_q 값의 Max_{th} 초과 시의 패킷 폐기 등의 결과가 송신 측에 전달되어 이에 따라 트래픽 발생 율을 조절함으로써 이루어진다. 그러나, 패킷 폐기 결과에 따라 혼잡 제어가 이루어지는 것은 네트워크 상황에 수동적으로 동작하는 것이며 AQM의 효율성을 저하시킨다[5]. 따라서 네트워크 상황에 보다 능동적으로 대처하기 위한 ECN 마킹에 의한 혼잡 제어도 검토되었다[6][7].

RED의 파라미터 값 중에서 큐 가중치 값 W_q가 너무 작으면 평균 큐 크기가 최대 임계값을 초과했을 경우 도착하는 모든 패킷들은 모두 폐기가 되는데 이때 가능한 빨리 최대 임계값 이하로 평균 큐 크기가 내려가야 하지만 실제 큐 길이는 감소했다 하더라도 평균 큐 크기가 이를 따르지 못해 실제 큐 크기는 많이 줄어들었음에도 패킷의 폐기가 계속적으로 이루어지며 또한 큐 가중치가 너무 클 경우 현재 큐 크기에 너무 많은 영향을 받아 RED 사용으로 인해 얻는 이점이 줄어들게 된다.

본 논문에서는 RED의 변형된 방법으로 평균 큐 크기 avg_q를 유지하면서 RED에서 사용되는 Min_{th}, Max_{th}, Max_p, W_q 등의 4개 파라미터 중 최대 임계값 Max_{th}와 큐 가중치 값 W_q를 현재의 큐 상태에 따라 변경시켜 가용 큐 크기를 늘려 큐 사용 율을 높이고, 또한 평균 큐 크기가 최대 임계값에 도달한 이후에 급격한 패킷의 폐기를 줄여 지터를 줄이도록 하였다. 즉, 최대 임계값 Max_{th}와 큐 가중치 W_q를 avg_q의 변화에 따라 총 네 가지 단계로 변화하도록 제안하였으며, 큐 크기의 90%를 초과할 경우 버퍼 오버플로우로 가정하여 모든 패킷에 대한 ECN 마킹 또는 패킷 폐기를 수행하도록 하였다. 이는 실시간성 트래픽에 대한 높은 수율 (throughput)과 낮은 지연을 함께 성취하기 위함이다. 평균 큐 크기 avg_q 값의 증가에 따라 Max_{th}를 초과하는 상황이 발생하지만 기존의 방법에서와 같이 Max_{th} 초과 시 모든 패킷을 폐기하는 것을 지양하고, 초기 최대 임계값인 $init_max_{th}$를 초과함에 따라 전체 큐 크기의 10% 크기를 Max_{th} 값에 단계 별로 추가하고 이에 따라 W_q 값을 지수함수적으로 증가시켜 단계 별 혼잡 상태 대처 기준을 강화시켰다. 이렇게 함으로써 혼잡 상태 진입 초기부터 수율은 전체 큐 크기의 90%까지 증가할 수 있으며, 이와 동시에 혼잡 상태를 유발하는 트래픽에 대한 혼잡 제어는 단계 별로 더 강화되어 패킷 지연을 일정 수준 (본 논문에서는 안정적인 큐 크기를 전체 큐 크기의 최대 70% 이하로 정의하였으며, 이에 대한 상세한 내용은 III-1절 SDRED에서 기술한다) 이하로 유지할 수 있다. 더욱이 스케줄러에서 큐 관리 기법의 파라미터를 사용하게 함으로써 심한 혼잡이 발생한 플로우의 부하를 그렇지 않은 플로우에서 나누어 처

리함으로써 전체적인 큐 사용 율을 높이고 혼잡이 심한 플로우에 대한 서비스를 보완하도록 하였다.

본 장에 이어 II 장에서는 본 연구와 관련된 큐 관리 및 스케줄링 기법을 살펴보고 III 장에서는 SDRED 및 동적 스케줄링 기법을 제안한다. IV 장에서는 제안 기법과 기존 기법을 비교 분석하여 제안 기법의 성능을 검증하며 V 장에서 논문의 결론을 맺는다.

II. 큐 관리와 스케줄링 기법

본 장에서는 III 장에서 제안하는 SDRED와 관련된 연구로서 RED와 이로부터 변형 또는 보완된 큐 관리 기법들을 우선 살펴보고, GPS 및 DRR을 포함한 스케줄링 기법에 대하여 살펴본다.

1. 큐 관리 기법

RED는 큐의 길이가 임계값을 넘으면 평균 큐 길이를 기준으로 계산된 확률로 패킷을 폐기하는 방식이다. RED의 목표는 평균 큐 크기를 제어하여 폭주를 회피하고 TCP의 global synchronization을 해결하는 것이다. 즉, RED 알고리즘은 패킷이 도착할 때마다 평균 큐의 길이를 구해 정해진 최소 임계값과 최대 임계값을 비교하여 최소 임계값 이하이면 정상적인 상태로 인식하고, 두 임계값 사이이면 랜덤하게 패킷을 폐기하고 최대 임계값 이상이면 입력 패킷을 모두 폐기한다[3].

큐 운용을 위한 중요한 값의 하나인 큐 가중치 W_q는 현재 큐의 크기가 평균 큐 크기에 얼마나 영향을 주는가를 결정하는 값이다. 이 값이 너무 작으면 실제 큐 길이가 감소하더라도 평균 큐 크기가 최대 임계값 이하로 떨어지는 데 시간이 너무 많이 걸려 불필요한 패킷의 폐기가 일어나고, W_q 값이 너무 크면 현재 큐 크기의 영향을 너무 많이 받게 되어 RED를 사용하는 의미가 감소하게 된다. 한편, RED는 특별한 타입의 플로우를 차별하지 않음으로써 모든 플로우에 공정하게 대역폭을 보장하지 않는다. 사실상 RED가 저속의 TCP 플로우에 대해서 불공정성을 나타내는데, 이는 RED가 최소 임계값을 지날 때 패킷을 랜덤하게 폐기시키기 때문이다. RED의 응용으로 FRED (Flow RED)가 있는데 라우터의 큐에서 각 TCP 연결이 차지하는 큐 길이를 추적함으로써 트래픽 유형에 관계없이 모두 공평하게 링크를 사용하게 한다[8]. 다음으로 RED를 기반으로 보완된 기능을 제공하는 RIO, WRED, BLUE, ARED에 대하여 기술한다.

RIO (RED In and Out)는 사용자 별 차별 서비스를 제공한다. SLA (Service Level Agreement)를 준수하는 패킷의 경우는 In-profile 패킷으로, 그렇지 않은 경우는 Out-profile 패킷으로 구분하여 각각 다른 RED를 적용한다. 평균 큐 크기가 두 임계값 사이에 있을 경우 In-profile의 패킷은 폐기시키지 않고, Out-profile의 큐는 랜덤하게 폐기하게 된다. 평균 큐 크기가 최대 임계값을 넘었을 경우에는 Out-profile의 패킷을 In-profile의 폐기보다 더 높은 확률로 폐기하게 된다[9]. RIO 알고리즘은 대역을 지나치게 남용하는 사용자로부터 규약을 준수하는 사용자를 보호하여 공정한 대역의 사용을 보장하는 방법이다.

WRED (Weighted RED)는 가중치가 다른 트래픽이 혼합되었을 경우 이를 차별적으로 서비스함으로써 공정성을 제공하기 위한 스케줄링 방법과 RED를 적용한 것이다. WRED는 구분된 트래픽 클래스에 대해 별도의 RED큐를 운영하며 각 큐에 대해서 다른 RED 파라미터를 지원한다. WRED에서는 큐가 포화상태에 이르렀을 때 패킷이 우선 순위에 따라 폐기되지만, 서로 다른 우선 순위의 패킷이 동시에 혼재할 때 보다 높은 우선 순위의 패킷이 나중에 서비스를 받을 수도 있다. 즉, 각 트래픽에 서로 다른 파라미터를 적용한다 하더라도 스케줄러에는 전혀 영향을 미치지 못하므로 낮은 우선 순위의 패킷이 높은 우선 순위의 패킷보다 먼저 서비스를 받을 수 있다[10].

BLUE는 RED의 문제점을 보완하기 위해 제안된 방법으로, 패킷이 큐에 들어올 경우 폐기 또는 마킹하는데 사용하는 단일 확률 (single probability)를 유지한다[11]. BLUE의 주 아이디어는 순간적인 큐나 평균 큐 길이 보다는 패킷 손실과 링크 사용 율을 기초로 큐 관리를 수행한다는 것이다. 큐가 버퍼 오버플로우에 의해 계속적으로 패킷을 폐기하면 BLUE는 마킹 확률을 증가시키고 따라서 되돌려 지는 혼잡율 (congestion rate)이 증가한다. 반대로 큐가 비거나 링크가 idle 할 경우에는 마킹 확률을 감소한다. BLUE는 트래픽 부하가 급속히 증가할 경우에 신속히 반응한다는 장점이 있으나 혼잡 상황 이후에 계속적인 버스트 트래픽이 도착할 때는 혼잡 제어가 원활하게 수행되지 않으며 RED의 최대 임계값을 전체 큐 크기에 가깝게 설정을 했을 때와 유사한 성능을 보인다.

ARED (Adaptive RED)[6]는 RED가 Drop Tail 큐보다 향상된 성능을 보이지만, 다양한 혼잡 시나리오에서 RED 큐 파라미터를 조정하기 어려운 점에 착안하여 평균 큐 크기와 활성 상태의 연결 수의 변화를 주시하여 RED의 패킷 제어 정도 (aggressiveness)를

조절하는 방안을 제시하였다. 즉 ARED는 RED의 혼잡 통지가 해당 링크에서 멀티플렉싱 되는 connection 수에 직접적으로 의존하지 않고, 패킷 손실 결과를 혼잡 통지의 수단으로 사용할 경우 RED의 이점이 상당히 감소된다는 사실을 지적하고 있다. 따라서 ARED는 RED의 패킷 제어 정도에 직접적으로 영향을 미치는 max_p 값을 connection 수의 변화에 따라 적응적으로 변경하고 이를 ECN 기능과 함께 사용함으로써 수율과 패킷 손실 측면에서 일정한 max_p 값을 갖는 경우보다 좋은 결과를 얻을 수 있음을 검증하였다. ARED는 활성 상태의 연결 수를 기초로 패킷 드롭 확률을 조절할 수는 있었지만, RED의 최대 및 최소 임계값과 큐 길이 가중치 등의 다른 RED 파라미터에 대한 고려가 충분치 않고, 실시간 특성을 갖는 트래픽에 대한 논의는 포함되어 있지 않다.

RED는 TCP 연결이 많을 경우 낮은 성능을 보이고 또한 트래픽 소스가 많거나 버스트 트래픽 환경에서는 공정성과 성능이 낮아진다. 이러한 RED의 공정성 문제를 해결하기 위한 방법으로 FB-RED (Fair Buffering RED)와 FRED (Fair RED), SRED (Stabilized RED), DSRED (Double Slope RED)와 같은 RED의 응용 기법들이 제안되었다.

FB-RED[12]는 ATM 상에서 TCP를 운용하기 위한 것이다. FB-RED는 링크 사이에 공정성을 제공하지만 모든 링크에 대한 정보를 추적해야 하며 큰 규모의 네트워크에서는 안정성을 보장하지 못하는 문제점을 가지고 있다. 즉, FB-RED 는 RED에서 제어하는 각 플로우에 대한 공정성에 초점을 맞춘 혼잡 제어 방법으로 혼잡에 직접적인 원인이 되는 플로우를 찾아 제어를 수행한다. 따라서 적은 규모의 네트워크에서는 비교적 수용할 만한 공정성과 성능을 보이지만, 네트워크의 규모가 커지고 플로우의 수가 증가함에 따라 이에 따른 오버헤드 및 하드웨어적 부담, 처리 시간 등이 문제가 된다. FRED[13] 역시 ATM 상에서 TCP를 운용하기 위한 것으로 각 플로우에 의해 점유되는 큐 공간을 기반으로 플로우의 폐기 율을 결정한다. 따라서 각 플로우간에 공정성을 제공할 수는 있지만 FB-RED와 마찬가지로 각 플로우의 상태를 추적해야 하므로 역시 안정성 문제를 내포한다.

이러한 안정성 문제를 해결하기 위하여 SRED가 제안되었는데[14], SRED는 폭넓은 네트워크 부하 범위에서 동작하며 활성화되어 있는 연결의 수에 영향을 받지 않는다. 그러나 작은 수의 트래픽 플로우에서도 매우 낮은 성능을 나타낸다는 단점이 있다. 한편, RED의 낮은 성능과 지연 특성을 보완하기 위해 DSRED가 제안되었는데[15], RED가 두 개의 임계값을 가지는데 비해 DSRED는 세 개의 임계값을 가지도록 함으로써 선형적인 패킷 폐기 확률을 두 개로 나누었다. 따라서 서로 다른 기울기의 확률 함수로 패킷 폐기 확률이 계산되어, RED에 비해 낮은 혼잡 상태에서 낮은 패킷 폐기를 보이며 높은 혼잡 상태에서도 RED 보다 더 낮은 큐 지연과 큐 크기를 유지한다. 그러나 이 기법 역시 혼잡 상태에서 패킷이 계속적으로 유입될 경우에는 성능 저하를 막을 수는 없다.

2. 스케줄링 기법

스케줄러 구현 시 요구되는 사항 중 가장 중요한 것 중의 하나가 공정성 제공이다. GPS (Generalized Processor Sharing)는 가장 이상적인 스케줄링 방법으로 각 플로우에 아주 적은 양의 데이터들을 라운드 로빈 형태로 서비스한다. 플로우에 가중치가 설정되어 있을 경우 그 가중치에 비례하여 서비스 시간이 늘어나며 이때 처리되는 데이터의 양은 변함이 없다. 그러나 아주 적은 양의 데이터나 비트 단위로 서비스를 할 경우에는 너무 많은 부하가 생기기 때문에 구현 측면에서 볼 때 사실상 어렵다.

현실적으로 구현이 힘든 GPS 방식을 그대로 따르지 않고 비트 단위가 아닌 패킷 단위로 서비스하는 방법으로 PGPS (Packetized GPS)로도 알려져 있는 WFQ (Weighted Fair Queueing)[16]가 있다. WFQ에서 서비스는 가상 종료 시간에 따라 이루어진다. 여기서 가상 종료 시간은 일단 GPS를 비트별 라운드 로빈 방법으로 모델링하고, 임의의 시간에서 비트별 라운드 로빈 GPS의 라운드 수 (시스템 가상 시간)를 라운드 넘버 또는 시스템 가상 시간이라 한다. 따라서 시스템 가상 시간은 실제로 비트별 라운드 로빈 GPS로 스케줄 했을 때 특정 시간을 기준으로 한 GPS 서비스 양을 의미한다. PGPS 정책은 유동 흐름 도착 프로세스 (fluid flow arrival process)와 참조 시스템 (reference system)에서 패킷의 출발 시간을 계산해서 참조 GPS시스템을 시뮬레이션 하는데, PGPS 스케줄러는 참조 GPS 시스템에서 가장 일찍 출발한 패킷을 선택하여 전송한다. 여기서 참조 GPS 시스템이란, Fair Queueing 스케줄러가 실제로는 유동 흐름을 가지지 않으므로, GPS를 하나의 참조 모델 (reference model)로서 주로 사용하고 있고, 이러한 참조 모델을 시뮬레이션 함으로써 GPS 정책에 근접한 서비스 정책 (service disciplines)을 정의할 수가 있으며, 참조 모델로 시뮬레이션 된 시스템을 참조 시스템이라 정의한다[17]. PGPS 정책이 많이 쓰이는

이유 중 하나는 초과 트랙픽에 대한 공정한 처리 때문이다. 스케줄러는 가장 작은 가상 종료 시간을 갖는 패킷을 전송하므로 패킷의 종료 시간을 기반으로 하는 우선순위 큐를 이용할 경우 효과적으로 수행될 수 있다. 그러나 GPS와 마찬가지로 흐름의 수가 많아질 경우에는 효과적이지 못하다[8].

또 다른 스케줄링 기법인 WRR (Weighted Round Robin)은 간단한 방법으로 GPS를 적용한 것이다. 즉, GPS와 같이 라운드 로빈 방식으로 패킷 단위의 서비스를 하는데 각 플로우의 가중치에 비례하여 패킷을 서비스한다. 이 경우 플로우가 얻게 되는 대역폭이 패킷 길이에 의존하므로 패킷의 길이가 일정한 ATM과 같은 환경에서는 비교적 정확한 보장을 할 수 있으나 가변적일 경우에는 공정성을 제공하지 못할 수도 있다. 한편, DRR (Deficit Round Robin)은 각 플로우의 평균 큐 크기를 참조하지 않고도 가변길이의 패킷을 고려해 공정성을 제공할 수 있도록 고안된 방법이다. 각각의 흐름은 부족 계수기 (deficit counter)를 가지며 스케줄러는 각 흐름을 번갈아 선택하여 특정량 (quantum) 만큼만 서비스한다. 각 흐름의 앞에 있는 패킷의 크기가 퀀텀 값보다 작을 경우 서비스하고 남은 차감 값이 부족 계수기에 더해진다. 즉, 부족 계수기의 값은 서비스를 받아야 하는데 패킷의 크기가 작아 서비스를 받지 못하고 남은 양을 의미한다. 결국, DRR은 구현이 쉬우면서 WRR의 프레임 시간 내에서는 공정성이 약화된다는 단점을 극복한 것으로 프레임 시간이 작거나 공정성 요구가 크지 않은 상황에서 잘 동작한다[18].

III. SDRED와 동적 스케줄링 기법

본 장에서는 실시간 인터넷 서비스를 II장에서 기술한 큐 관리와 스케줄링 기법보다 좀더 효율적으로 지원하기 위한 SDRED 기법과 SDRED의 다중 큐에서의 동적 스케줄링 방법에 대하여 기술한다.

1. SDRED

RED는 평균 큐 크기를 제어하고, 혼잡 상태를 회피하며, TCP의 global synchronization 문제를 해결한다. 그러나 혼잡 상태가 어느 정도 지속적으로 이어질 경우 패킷 폐기 율이 증가하고, 큐의 사용 율이 저하되는 문제가 발생한다. 이러한 문제는 RED가 고정된 파라미터 값을 사용하고, 평균 큐 값을 유지하려고 하는 속성으로부터 발생한다. 즉, 평균 큐 크기를 유지하는 것이 RED에서 혼잡을 제어하는 핵심 부분이지만, 이로 인해 반응 (reaction)이 늦어지는 결과를 초래하게 된다. 이를 보완하기 위한 SDRED 기법은 RED의 다음과 같은 동작 시나리오를 통해 상세사항을 기술할 수 있다.

단계 1: TCP 부하 집합의 충분한 변화로 병목 링크의 용량을 초과하는 TCP 전송 측의 전송 율 발생
단계 2: 부하와 용량의 불일치로 큐에 병목 형성
단계 3: 평균 큐 길이의 최소 임계값 (Min_{th}) 초과로 congestion-mechanism 유발
단계 4: TCP 수신 측은 패킷 손실을 검출 또는 ECN 비트가 설정된 패킷을 확인
단계 5: Duplicate ACK와 ECN 신호는 송신 측에 혼잡 상황을 통지
단계 6: 송신 측은 결국 혼잡을 검출하고 자신의 전송 율에 이를 적용
단계 7: 마지막으로 공급된 부하가 병목 링크에서 감소

위로부터 링크 용량보다 공급된 부하가 작게 되기 전까지 단계 1~7이 수행된다. 이는 실제로 병목 링크의 낮은 사용 율을 유발한다[19]. 늦은 반응 문제를 극복하는 방법으로 평균 큐 크기를 이용하지 않고 현재의 큐 크기를 이용하여 혼잡을 제어하는 BLUE와 같은 방법이 있다. 그러나 본 논문에서는 RED의 가장 큰 특징인 평균 큐 크기를 유지하면서 혼잡이 발생하기 시작했을 때의 RED의 동작 방식을 변형하여 좀더 빠른 반응과 향상된 큐의 사용 율을 유도하고, 최대 임계값 이후의 Drop Trail과 같은 동작을 수정함으로써 실시간 전송에 중요한 영향을 미치는 지터를 감소할 수 있도록 SDRED를 제안한다.

SDRED는 RED의 네 가지 파라미터 중 최대 임계값과 큐 가중치 값을 평균 큐 크기에 가변적인 값으로 가진다. 일반적으로 큐가 최대 임계값에 도달한 후에도 큐 크기가 증가할 경우 평균 큐 크기도 계속 증가하여 최대 임계값을 초과하게 되며, 혼잡 상태가 지속되어 평균 큐 크기가 최대 임계값을 계속 유지하게 될 경우 인입되는 패킷은 모두 폐기된다. 그러나 SDRED에서는 최대 임계값을 큐 크기의 90%까지 증가시키도록 설계하였다. 그리고 증가된 최대 임계값으로 인하여 작아지는 drop 확률은 큐 가중치를 증가시킴으로써 현재의 큐 크기를 평균 큐 크기에 더 많이 반영한다. 결국, 초기 최대 임계값 이후로는 평균 큐 크기가 증가할수록 현재 큐 크기를 더욱 많이 반영하도록 되어 있다.

SDRED는 패킷이 도착하면 RED와 마찬가지로 평균 큐 크기를 계산한다. 그러나 평균 큐 크기가 전체 큐 크기에 도달한 값을 기준으로 임계값을 증가 또는 감소시킨다. 이 때 최대 임계값의 증감에 따라 큐 가중치 값 또한 증감되고, 다음 패킷이 도착하면 갱신된 파라미터 값으로 평균 큐 크기가 계산되는 것이 SDRED의 특징이다. SDRED의 동작 흐름도를 그림 1에 도시하였다. 그림에서 k는 큐 점유율 증가에 따른 W_q 값 변경의 가중치로 사용한 변수이며, 이 값이 1일 경우가 original RED[3]에 해당한다. 본 논문에서는 큐 점유율을 4 단계로 구분하여 점유율 증가에 따라 W_q 값을 k^0, k^1, k^2, k^3의 상수 값을 곱하여 증가시킴으로써 현재 큐 점유율의 avg_q에의 반영도를 높인다.

또한 수율 및 지연을 모두 안정적으로 제공할 수 있는 큐 점유율 목표치를 전체 큐 크기의 70%까지로 정의하여 초기 최대 임계값 $init_max_{th}$를 전체 큐 크기의 60%로 설정하고 참고문헌[3]에서 검토한 W_q의 상한 값과 하한 값 요구 사항에 맞는 k=5 값을 선택

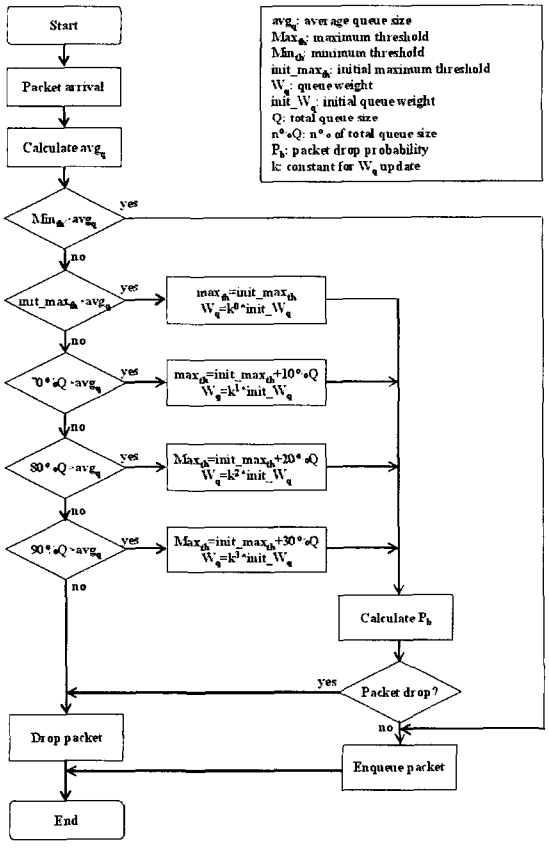


그림 1. SDRED의 동작 흐름도

하여 이를 만족하도록 하였다. 3단계와 4단계는 현재의 큐 크기를 avg_q 계산에 더 많이 반영하여 혼잡 제어를 강화하여야 하므로 큐 가중치를 한층 더 높이도록 설계하였다.

결과적으로, 최대 임계값을 늘림으로써 실제로 사용될 수 있는 큐의 가용 율을 늘리고, 큐 가중치는 재설정되는 최대 임계값에 비례하여 증가시킴으로써 급격한 패킷 폐기 상태를 줄일 수 있다. 따라서 큐가 혼잡을 경험하기 시작하는 시점에서 큐에 대한 보다 안정적인 혼잡 제어 메커니즘을 제공하고, 낮은 큐잉 지연과 작은 지터를 보장하게 됨으로써, 실시간 서비스 트래픽에 대하여 기존의 큐 관리 기법보다 더 향상된 성능을 제공할 수 있다.

2. SDRED의 다중 큐에서의 동적 스케줄링

기존의 RED나 WRED는 트래픽의 폭주를 제어하고 TCP의 global synchronization을 해결할 수는 있지만 트래픽을 플로우별로 서비스할 수는 없다. 본 논문에서는 실시간 서비스를 지원하기 위하여 큐를 서비스 종류 별로 분리하고 각각의 큐에 SDRED 큐 관리 기법을 적용하여 특정한 큐 상황에서의 스케줄링에 RED 파라미터를 이용하는 동적 스케줄링 방법을 고안한다.

분리된 각각의 큐는 idle 상태에서는 서로 독립적으로 SDRED에 의해 큐가 관리되고 스케줄링 된다. 그러나 어느 하나의 플로우가 혼잡 상태에 접어들게 되면 그렇지 않은 idle 한 큐의 대역폭을 스케줄러에 의해 나누어주게 된다. 혼잡 상태에 따른 이러한 차별된 스케줄링은 RED 파라미터에 의해 설정된 값을 기준으로 결정하게 된다.

실시간 트래픽에 대한 보다 향상된 서비스 제공을 위하여 동적 스케줄링 기법과 SDRED를 함께 적용함으로써 서로 독립적으로 영향을 미치지 않던 기존의 큐 관리 및 스케줄링 기법과는 달리, 스케줄러가 큐 관리 기법의 파라미터를 이용할 수 있고 큐의 상태를 반영하는 스케줄링 동작을 유도할 수 있다. 각 큐의 상태에 따라 동적 스케줄러는 계속적으로 변하는 가중치를 갖게 됨으로써, 특정 플로우의 혼잡에 의해 해당 플로우의 서비스 율이 저하되는 것을 방지하고 결과적으로 전체적인 큐 사용 율을 높일 수 있다.

가중치가 설정되지 않은 다중 큐에서의 SDRED는 큐가 일정 크기의 임계값을 넘으면 혼잡 상태로 판단하여, 그렇지 않은 큐보다 선형적으로 더 많은 횟수로 스케줄링 되도록 한다. 그림 2에 다중 큐에서의 큐의 상태 변화에 따른 SDRED와 동적 스케줄링 기

법의 동작 상황을 도시하였다. 그림에서 보는 바와 같이 idle 한 상태에서는 모든 큐가 라운드 로빈 방식으로 동일하게 스케줄링 되지만, 임의의 한 큐가 혼잡 상태로 들어서게 되면 SDRED 큐 관리 방식에 의하여 해당 큐의 최대 임계값이 증가하고, 스케줄러는 큐의 상태를 반영하여 혼잡 상태의 큐에 대한 가중치를 증가시켜 다른 큐보다 혼잡 상태의 큐를 더 빈번히 서비스한다.

Ⅳ. 성능 비교 및 분석

제안한 SDRED의 성능 평가를 위하여 OPNET을 사용하여 단일 큐와 다중 큐의 두 가지 큐 모델에 대하여 시뮬레이션을 수행하였다. 단일 큐 환경의 경우 SDRED, RED, ARED, BLUE 각각의 성능을 측정, 비교하였다. 다중 큐 환경에 대해서는 각 큐에 SDRED를 적용하고 각 큐의 상태에 따라 스케줄러가 SDRED의 파라미터를 참조하여 큐가 혼잡 상태인지를 확인하여 혼잡 상태인 경우 두 배의 가중치를 주어 스케줄링이 유동적으로 동작하는 경우와 스케줄러와 큐 관리 기법이 서로 영향을 미치지 않는 경우에 대하여 성능 비교를 수행하였다. OPNET을 이용한 몇 가지 AQM 알고리즘에 대한 모델링 방법 및 프로세스 모델은 참고문헌[20]을 참조한다.

1. SDRED와 RED, BLUE, DSRED 비교

각 기법의 성능 비교를 위하여 그림 3과 같이 단일 큐를 갖는 하나의 큐 모듈과 8 개의 패킷 발생 모듈을 가정한다.

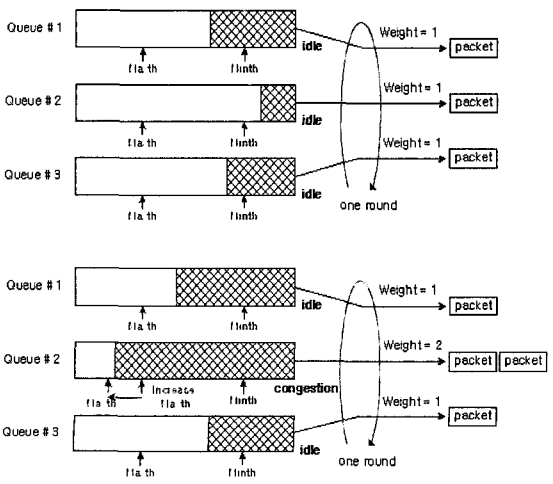


그림 2. 다중 큐에서의 SDRED와 동적 스케줄링

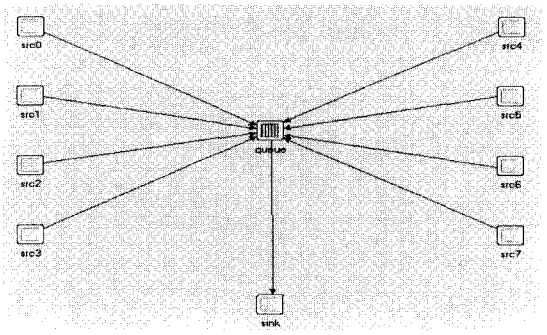


그림 3. 실험 환경

그림 3의 소스 중 4 개는 1024 바이트의 고정 크기의 패킷을, 그리고 나머지 4 개는 1024 바이트의 평균 크기를 가지는 지수 분포를 가지고 패킷을 발생하도록 하였다. 패킷 도착 시간은 평균 1초의 지수 분포를 가지도록 설정하였으며, 큐 모듈의 서브 큐 크기는 12,000 바이트로 설정하였다.

제안된 알고리즘은 큐 모듈 내의 프로세스 모델로 그림 4와 같이 구현된다. 총 5 개의 상태로 구성되고, 각 상태의 동작은 다음과 같다.

init: 시뮬레이션 변수 값을 초기화

arrival: 패킷의 큐 도착 시 수행되는 단계로, 알고리즘이 구현되는 부분이다. 알고리즘 내의 파라미터 설정과 패킷의 폐기 또는 삽입 등이 수행된다.

svc_start: 패킷에 대한 서비스 시작에서 종료까지의 단계로서 서비스 상의 지연, 서비스 시간 등이 계산된다.

svc_compl: 서비스 완료 패킷의 큐에서의 완전 삭제 및 다른 노드 모듈로의 패킷 전송이 수행된다.

idle: 서버의 휴지 상태를 나타낸다.

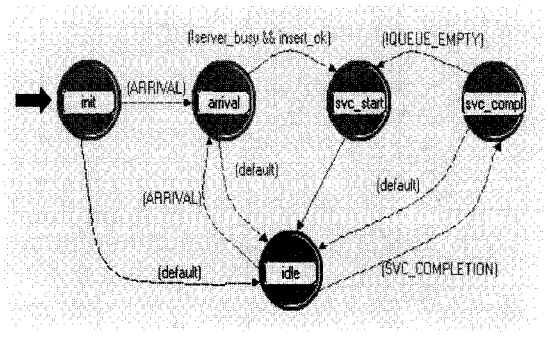


그림 4. 큐 모듈의 프로세스 모델

큐 활용 율에 대한 실험 결과는 그림 5와 같다. 그림에서 SDRED는 RED나 ARED보다 높은 큐 활용 율을 보이고 있으며, BLUE가 가장 높은 큐 활용 율을 나타냄을 알 수 있다. 이것은 BLUE가 RED, ARED, SDRED와는 달리 평균 큐 크기를 이용하지 않으므로 큐가 최대 임계값 근처에 도달한 후에야 패킷 폐기가 시작되기 때문이다. 그러나, 큐 활용 율은 그만큼 높은 지연을 가지게 하며 또한 같은 조건에서 유사한 성능을 보일 경우 요구되는 큐의 크기가 커짐을 의미한다. 한편, ARED는 max_p 값을 큐의 크기가 max_{th}를 초과할 경우 aggressive detection을 할 수 있도록 증가하므로 RED보다 큐 활용율 및 수율이 높게 나타난다. SDRED는 큐 상태의 각 단계 별로 W_q 값과 max_{th} 값을 조정함으로써 큐 상태에 더 적극적으로 적응하므로 ADRED 보다 높은 큐 활용율을 보인다. 아울러, SDRED는 그림 1의 동작 흐름도에 정의한 바와 같이 전체 큐의 70% 정도를 안정적으로 사용함을 알 수 있다.

그림 6은 지연 시간에 대한 결과이다. 위에서 지적한 바와 같이 BLUE가 평균적으로 가장 큰 지연을 겪는다. 이는 보다 큰 큐 크기를 유지함으로써 상대적으로 패킷 폐기 율이 감소하고, 큰 지연 시간으로 이어진 결과이다. RED와 ARED, 그리고 SDRED는 거의 비슷한 평균 지연 시간을 나타내고 있다. 그러나, 실시간 음성이나 영상 정보의 전송 시 큰 영향을 미치는 지터를 살펴보면, RED가 지터가 제일 크고, 다음으로는 큐 혼잡도에 따라 max_p 값을 조정하는 ARED, 그리고 SDRED의 순으로 지연변동 특성이 좋은 결과를 나타낸다. 이는 SDRED가 큐의 상태에 따라 단계별 max_{th} 값을 기준으로 현재 트래픽 상황에 최대한 적응적으로 반응하기 때문이며, 결국 실시간 서비스에 대해서는 SDRED가 가장 좋은 성능을 나타낼 수 있음을 확인할 수 있다.

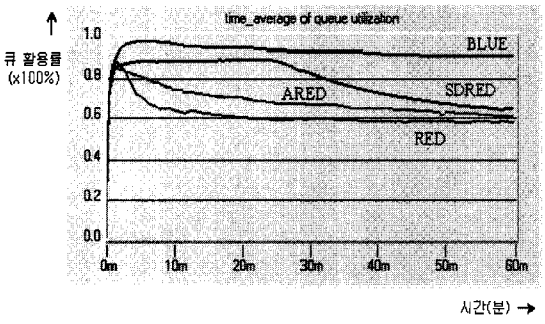


그림 5. 큐 활용율 비교

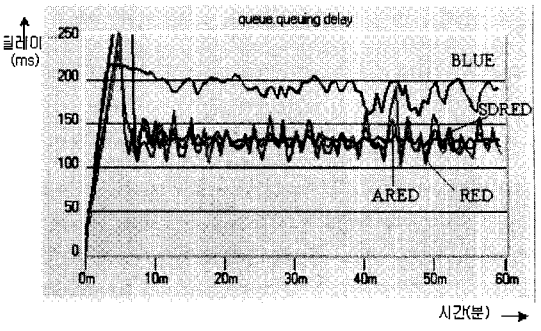


그림 6. 지연 시간 비교

그림 7과 그림 8에 RED의 낮은 성능과 지연 특성 보완을 위해 제안된 또다른 응용 기법인 DSRED와 SDRED의 큐 활용 율 및 지연 시간에 대한 성능 비교를 나타내었다. 그림 7에서 DSRED가 SDRED보다 전체적으로 다소 높은 큐 활용 율을 보이나 전체적으로는 유사한 결과를 나타낸다. 이는 DSRED가 최소 임계값과 최대 임계값 사이에 임계값을 하나 더 설정하여 패킷의 폐기 확률이 낮아짐에 따라 더 많은 패킷이 큐에 인입되기 때문이다. 한편, 최대 임계값 이후 DSRED는 RED나 ARED와 마찬가지로 모든 패

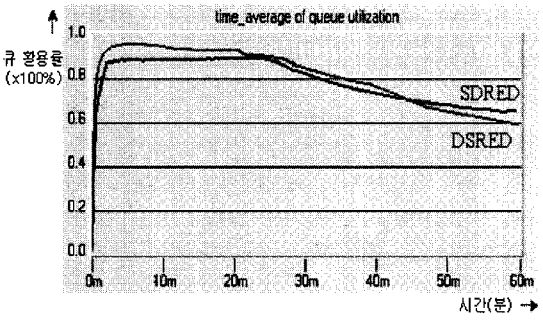


그림 7. SDRED와 DSRED의 큐 활용율 비교

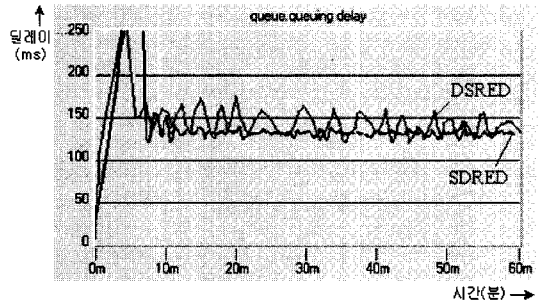


그림 8. SDRED와 DSRED의 지연 시간 비교

킷을 폐기하므로 큐 활용 율이 다소 감소하게 된다. 그림 8의 지연 특성에 대한 결과에 있어서는, DSRED가 SDRED에 비해 약간 더 높은 큐 활용율을 나타내기 때문에 평균 지연 값 또한 약간 높으며, 지터 특성 또한 RED나 ARED 등과 마찬가지로 SDRED보다 훨씬 더 큰 변동을 보임을 알 수 있다. 결국, 그림 5에서 그림 8의 결과를 종합해 보면 SDRED가 실시간 인터넷 서비스 제공 시 RED, BLUE, ARED, DSRED 등의 기법보다 더 좋은 성능을 나타낼 수 있음을 확인할 수 있다.

2. DRR과 동적 스케줄링 기법 비교

동적 스케줄링 기법의 실험 환경은 큐 모듈 내에 세 개의 서브 큐를 설정한 것과 svc_start 상태에서의 스케줄러가 round robin 형태가 아니라 서브 큐의 상태에 따라 동적으로 변하는 가중치를 가진다는 것 외에는 앞의 실험 환경과 동일하다.

시뮬레이션에서는 스케줄러가 혼잡 상태의 큐가 없는 idle한 상태에서 round robin으로 동작하다가 혼잡 상태의 큐가 발생하면 일시적으로 다른 가중치 값을 적용하여 스케줄링을 수행하는 동적 스케줄러와, 큐의 상태를 반영하지 않고 패킷의 크기를 고려한 DRR 기법을 비교하였다. 혼잡 상태 발생에 따른 두 기법의 성능 비교를 위하여 그림 9와 같은 패킷 입력 패턴을 사용하였다. 즉, 서브 큐 0번과 서브 큐 1번은 각각 평균 패킷 수를 8개와 40개로 하되 동적 스케줄러의 성능 검증을 위하여 평균 패킷 수가 5배 차이가 나고 시뮬레이션 시작 후 5분간은 평균 패킷 수 보다 더 많은 패킷을 입력하였으며, 서브 큐 2번은 시뮬레이션 시작 후 종료 시까지 점진적으로 평균 패킷 수를 증가시켰다. 결국 시뮬레이션 시작부터 3분까지는 서브 큐 1번에서 주로 혼잡을 겪게 되고 이후에는 서브 큐 2번이 혼잡을 겪게 된다. 이렇게 함으로써 각 서브 큐가 혼잡 상태에 도달하는 시점이 다르게 되고 결국 서로 다른 가중치 값으로 서비스를 받게 된다.

그림 10은 이러한 상황하에서의 전체 큐 활용 율에 대한 결과를 나타낸다. 실험 결과를 통하여 동적 스케줄러의 큐 활용 율이 DRR보다 더 높은 값을 유지하고 있음을 확인할 수 있다. DRR은 각 플로우에 패킷의 크기를 고려한 공정성을 제공하지만 특정의 혼잡한 플로우와 그렇지 않은 플로우를 구분하여 서비스하지는 않는다. 동적 스케줄러에서는 패킷의 개수에 의해 서비스를 하여 공정성을 제공하지 못하는 round robin과 유사하지만 큐의 현재 상태에 따라 가변적으로 각 큐에 가중치를 갖게 한다. 따라서 단순히 패킷 수를 기반으로 하지 않고 큐의 현재 상태를 기반으로 하는 방법으로 혼잡한 상태의 큐가 그렇지 않은 큐의 대역폭을 가져가게 하여 패킷의 폐기 시점을 늦추고 부하를 분산시킨다. 결국 혼잡하지 않은 큐를 혼잡한 큐가 사용하는 것과 같은 효과를 주게 되며, 특정 플로우의 현저한 서비스 저하를 막고 전체적인 큐 활용율을 높이게 된다.

그림 11에 DRR과 동적 스케줄링 기법의 시간 평

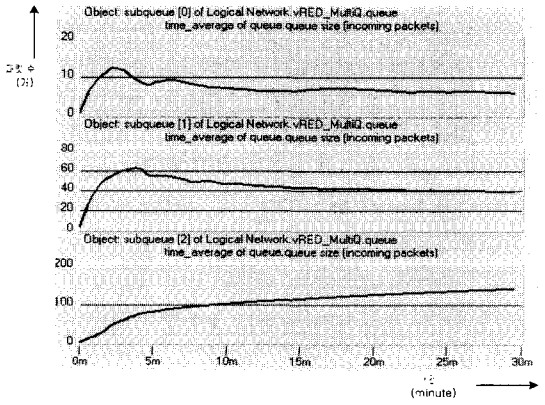


그림 9. 서브 큐에의 패킷 입력 패턴

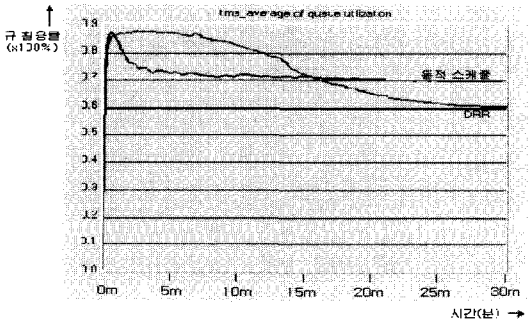


그림 10. 평균 큐 활용 율

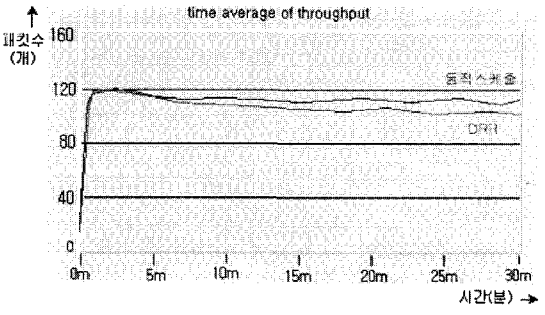


그림 11. DRR과 동적 스케줄링 기법의 수율 비교

균 수율 값에 대한 실험 결과를 나타내었다. 시간 당 패킷 처리 수에서 동적 스케줄러가 DRR보다 약간 높은 값을 나타낸다. 이는 각 서브 큐에 패킷 인입이 다르게 됨으로써 DRR의 경우 동적 스케줄러보다 낮은 큐 활용 율로 더 많은 패킷 폐기와, deficit 계수의 유지에 따라 전체적으로 더 낮은 패킷 처리율을 유도하기 때문이다.

이상의 실험에서 검증한 바와 같이 SDRED는 기존의 RED의 혼잡 제어 기능과 global synchronization 문제 해결, 공평성 등의 이점을 그대로 유지하면서도, ADRED의 특징인 트래픽 상황 및 큐 상태에 대한 적응성과 패킷 지연 특성을 한층 향상시킬 수 있고, 지연 특성을 보완하기 위해 제안된 DSRED보다도 좋은 지연 특성을 나타내므로 향후 인터넷상에서 더욱 증가될 실시간 특성을 갖는 멀티미디어 서비스를 지원하기 위한 또 하나의 좋은 RED 응용 기법이 될 수 있을 것이다. 구현 측면에 있어서는 기존의 RED나 ARED가 큐 점유 상태를 min_{th} 이하와 min_{th}와 max_{th} 사이, 그리고 max_{th} 이상의 3 단계로 구분한 것에 비해 SDRED는 max_{th} 값이 4단계로 나뉘어져 있어서 각 단계별 처리를 위한 변수 값만이 바뀔 뿐 복잡한 큐 점유율 계산을 필요로 하거나 추가의 하드웨어가 필요하지 않다. 따라서 기존의 RED 기법이 적용될 수 있는 네트워크 환경이라면 SDRED 또한 쉽게 적용될 수 있는 특징을 가지고 있다.

V. 결 론

인터넷상에서의 멀티미디어 및 실시간 서비스가 급증함에 따라 이를 지원하기 위한 고도화 된 큐 관리 및 스케줄링 기법이 요구되고 있다. 큐 관리의 가장 중요한 사항은 혼잡 상태에 대한 제어 능력이다. 기존의 여러 기법들이 이러한 능력을 가지고 있으나, 실시간 인터넷 서비스가 급증하는 상황에서는 문제점을 가지게 되었다.

본 논문에서는 큐 활용율은 물론, 실시간 서비스 제공 시 성능 상 중요한 문제가 될 수 있는 패킷 지연 및 지터 특성에 대한 문제를 해결하기 위하여, 최대 임계값과 큐 가중치 값을 평균 큐 크기에 가변적인 값으로 가지는 상태 의존 RED를 제안하고, 큐 활용 율과 지연 시간 등의 측면에서 성능을 개선할 수 있음을 시뮬레이션을 통하여 검증하였다. SDRED는 RED와 ARED, BLUE 등에 비하여 큐 활용 율 측면에서는 허용할 만한 성능을 나타내고 지연 시간 측면에서는 실시간 서비스를 지원하기에 알맞은 특성을 나타냄을 알 수 있었다. 특히, 기존의 기법 중 가장 많이 사용되었던 RED와 RED의 문제점을 개선하고자 한 ARED, DSRED, BLUE 및 DSRED 등의 기법과의 지연 특성 비교에 있어서 지터 값을 상당히 감소할 수 있었다. 더욱이, SDRED와 동적 스케줄링 기법을 함께 사용함으로써 전체적인 큐 사용율과 수율을 향상시킬 수 있음을 확인하였다.

향후 큐의 현재 크기만을 기준으로 혼잡을 제어하여 빠르게 반응하는 방법과, 평균 큐 크기를 유지했을 때의 가중치를 트래픽 상황에 맞게 예측하는 방법, 그리고 가중치 증감에 대한 수학적인 분석 방법에 대한 연구가 지속되어야 할 것이다. 또한, 본 논문에서 사용한 실험 환경과 트래픽 설정은 실제의 실시간 인터넷 트래픽을 모델링하여 수행한 것이 아니며, 실시간 서비스를 지원할 경우 만족되어야 할 수율 성능과 지연 성능 결과에 초점을 맞추었기 때문에 실제로 스트리밍 서비스나 MPEG-4와 같은 실시간 인터넷 서비스를 지원할 경우 제시 방안의 가용성과 효율성에 대한 연구가 지속되어야 할 것이다.

참 고 문 헌

[1] A. Mankin and K. Ramakrishnan, "Gateway Congestion Control Survery," IETF RFC 1254, August 1991.

[2] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," Internet RFC 2001, January 1997.

[3] S. Floyd and V. Jacobson, "Random Early Detection Gateway for Congestion Avoidance," IEEE/ACM Trans. Networking, Vol. 1, No. 4, pp. 397-413, August 1993.

[4] D. E. Lapsley and S. H. Low, "Random Early Marking for Internet Congestion Control," in Proc. IEEE GLOBECOM, December 1999.

[5] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," Computer Communication Review, Vol. 27, No. 1, July 1997.

[6] W. Feng, D. D. Kandlur, D. Saha and K. Shin, "A Self-Configuring RED Gateway," in

Proc. IEEE INFOCOM, March 1999.

[7] S. Floyd, "TCP and Explicit Congestion Notification," Computer Communication Review, Vol. 24, No. 5, October 1994.

[8] http://www.cisco.com/nwtworkers/nw00/pres/

[9] W. H. Park, S. Bahk and H. Kim, " A Modified RIO algorithm that alleviates the bandwidth skew problem in Internet differentiated service," ICC 2000, No. 1, pp. 1588-1603, June 2000.

[10] 박 현, 인터넷 QoS 기술, Lucent Tech., 1999. (http://www.krnet.or.kr)

[11] W. Feng, D. D. Kandlur, D. Saha and K. G. Shin, "BLUE: A New Class of Active Queue Management Algorithms," Technical Report, UM CSE_TR_387_99, University of Michigan, 1999.

[12] W. J. Kim and B. G. Lee, "The FB-RED Algorithm for TCP over ATM," http://tsp7.snu.ac.kr/int_conf/ic_42.pdf

[13] W. J. Kim and B. G. Lee, "FRED - fair random early detection algorithm for TCP over ATM networks," Electronics Letters, Vol. 34, No. 2, pp. 152-154, January 1998.

[14] T. J. Ott, T. V. Lakshman and L. Wong, "SRED: Stabilized RED," in Proc. IEEE INFO COM, pp. 1346-1355, March 1999.

[15] B. Zheng and M. Atiquzzaman, "DSRED: Improving performance of active queue management over heterogeneous networks," ICC 2001, No. 1, pp. 2375-2379, June 2001.

[16] A. Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," Journal of Internetworking: Research and Experience, Vol. 1, pp. 3-26, January 1990.

[17] R. Guerin and V. Peris, "Quality-of-service in Packet Networks: Basic Mechanisms and Directions," Computer Networks, No. 31, pp. 169-189, 1999.

[18] Cisco Systems, QoS Features for Voice over IP: Cisco Quality of Service for Voice over IP Solution Guide, Ch. 2, Cisco Press.

[19] http://www.thefengs.com/wuchang/blue

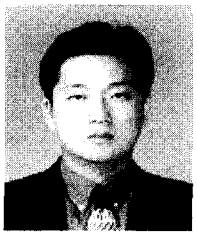
[20] C. Zhu, Oliver W. W. Yang, J. Aweya, M. Ouellette and D. T. Montuno, "A Comparison of Active Queue Management algorithms Using the OPNET Modeler," IEEE Communications Magazine, pp. 158-167, June 2002.

유 인 태(In-Tae Ryoo) 정회원

1987년 2월 : 연세대학교 전자공학과 학사
1989년 2월 : 연세대학교 대학원 전자공학과 석사
1994년 2월 : 연세대학교 대학원 전자공학과 박사
1997년 9월 : 동경대학 정보통신공학 박사
1999년 3월~현재 : 경희대학교 전자정보대학 부교수

<주관심분야> 인터넷, 네트워크 보안, 무선 LAN

홍 인 기(Een-Kee Hong) 정회원



1989년 2월 : 연세대학교 전기공학과 학사
1991년 2월 : 연세대학교 전기공학과 석사
1995년 8월 : 연세대학교 전기공학과 박사
1999년 3월~현재 : 경희대학교 전자정보대학 조교수

<주관심분야> 이동 통신, CDMA

서 덕 영(Doug-Young Suh) 정회원

1980년 2월 : 서울대학교 핵공학과 학사
1986년 6월 : Georgia Tech. 핵공학과 석사
1990년 6월 : Georgia Tech. 전기 및 컴퓨터공학 박사
1992년 3월~현재 : 경희대학교 전자정보대학 교수

<주관심분야> Networked Video, Internet QoS