클럭 기반의 이더넷 속도 제한 메커니즘

류현기°+이시영*류상률**김승호+
+경북대학교 컴퓨터 공학과, *㈜ 테크바일, **청운대학교
hkryu°@mmlab.knu.ac.kr+, sylee@techbile.com*, rsr@mail.chungwoon.ac.kr**, shkim@knu.ac.kr+

Rate Calculation: Clock-based Ethernet Rate-limiting Mechanism

Hyunki Ryu°+, S.Y Lee*, S.R Ryu**, S.H Kim+
+Dept. of Computer Engineering, Kyungpook National University
*TechBile Co. Ltd. , **Dept. of Computer Science Chungwoon University

요 약

최근 이더넷(Ethernet) 보급과 사용자의 증가 그리고 다양한 응용 서비스들의 등장으로 인하여 점차 네트워크 트래픽이 폭증하고 있다. 이에 따라서 SLA(Service Level Agreement)를 위한 대역폭 조절(Bandwidth의 Regulation)의 중요성은 더욱 높아지고 있다. 그러나 기존에 제시된 대역폭 조절 방식들은 이더넷에 적합하지 않거나 또는 너무 복잡하여, EFM(Ethernet for First Mile)에 적용하기 어려웠다. 그래서 본 논문에서는 이더넷에 적용하기 적합한 간단하면서도 효과적인 대역폭 조절(Bandwidth Regulation) 방법인 속도 계산(Rate Calculation)방식을 제안하며, 제안한 메커니즘에 대한 시뮬레이션을 통한 성능 분석을 수행하였다.

1. 서 론

기존에 LAN의 구축에만 사용되던 이더넷은 전 이중 전송 방식의 도입과 광전송 기술의 발전으로 거리에 제한이 사라지고 회로 기술의 발달로 말미암아 저가의 고속 스위칭 장비의 제공이 가능해 졌다. 이에 따라 메트로 광 이더넷이라는 새로운 분야를 개척하면서, MAN을 넘어 WAN을 구축하는 대안으로 떠오르고 있다. 특히 서비스 가입자와 제공자 사이의 EFM [1] 구간에서 이더넷은 고속 대용량의 트랙픽을 저가로 제공할 수 있는 방안으로 각광을 받고 있다. 급격히 늘어나는 사용자 트래픽을 효율적으로 관리하고, 다양한 사용자와의 SLA를 만족시키기 위해서는 송신측이나 수신측에서 필요한 만큼의 트래픽의 통과만을 허용할 수 있는 대역폭의 제한이 필요하다. 그러나 이더넷은 10/100/1000/10000 Mbps의 전송 계위만을 제공하므로 다양한 사용자들의 요구 사항을 만족시키기가 어렵다. 이에 따라 제공된 전송 계위 상에서 실제 대역폭을 제한할 수 있는 방법들이 연구되어 왔다.

제공 대역폭에 대한 사용자의 다양한 요구를 충족시키고, 효율적인 망 구축 및 관리를 위한 대역폭 제한은 제어를 수행하는 위치의 관점에서, 트래픽의 유입점에서의 대역폭 제한을 수행하는 트래픽 쉐이핑(Traffic shaping)과 트래픽의 진입점에서 조절이 이루어지는 트래픽 정책(Traffic Polishing)이 있다.

트래픽 쉐이핑은 리키 버킷(Leaky bucket) [2] 만을 사용한 방식과 토큰 버킷(Token bucket) [3]만을 사용하는 방식 그리고 두 가지 방식을 동시에 사용하는 방식으로 나뉘어 진다. 리키 버킷 방식은 데이터 버퍼인 리키 버킷에 쌓인 데이터를 정해진 속도로 균등 전송함으로써 대역폭을 제한한다. 토큰 버킷 방식은 제한하고자 하는 트래픽에 비례하는 토큰의 생성 주기와 생성된 토큰을 저장하는 버퍼인 토큰 버킷을 추가로 가진다. 이 방식은 토큰의 생성 속도를 조정함으로써 쉽게 동적으로 대역폭을 제한할 수가 있고, 입력단과 출력단의 전송 속도가 동일한 구조에서도 적용이 가능하다. 그러나 이 방식은 두 개의 버퍼를 유지해야 하므로 구성이 복잡해지고, 정확한 토큰의 생성 주기를 맞추기가 힘들 뿐만이 아니라, 긴 기간의 평균(Long term average) 개념에 따른 SLA를 초과하는 트래픽의 생성 가능성을 항상 가지고 있다.

본 논문에서는 EFM 구간에 활용되는 이더넷 장비에 쉽게 적용이 가능하도록, 토큰 버킷 방식의 유용성과 리키 버킷 방식의 단순성을 가진 이더넷 트래픽 제어 방법인 속도 계산 방식을 제안한다.

논문의 구성은 2장에서 기존의 리키 버킷, 토큰 버킷 만을 사용한 방식과 리키 버킷과 토큰 버킷을 함께 사용한 방식에 대한 분석과 이더넷에 적용에 따른 장단점을 분석한다. 3장에서는 본 논문에서 제안하고 있는 속도 계산 방식에 대해 설명하며, 4장에서는 OPNET에서 속도 계산 방식을 이용한 이더넷 트래픽의 조절에 대한 시뮬레이션을 통해 제안된 방식을 검증하고, 5장에서 결론을 내린다

2. 관련 연구

2.1 리키 버킷과 토큰 버킷

리키 버킷은 그림 1과 같이 입력 단에서는 입력되는 패킷(Incoming Packet)을 순서대로 버퍼에 담고, 출력단에서 조절 되는 장비의 클럭 속도에 맞추어 패싱(Passing)

하는 방식이다.

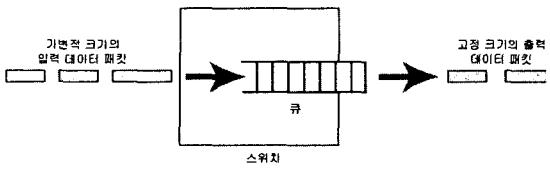

그림 1. 리키 버킷 방식

이러한 경우는 입력단과 출력단의 클럭 속도(Clock Speed) 자체가 다르므로 자동적으로 대역폭 조절이 이루어진다. 리키 버킷은 출력단의 전송 속도가 입력 단의 수신 속도보다 낮은 경우에 적용하기 좋고 패킷 버퍼 하나만 사용하므로 구현하기 쉽다. 그리고 버스트한 인터넷 데이터 트래픽(Internet Data Traffic)을 스태디 스트림(Steady Stream)으로 스무딩(Smoothing) 시킬 수 있는 점이 장점이지만 입력단과 출력단이 동일한 클럭 속도로 동작하는 경우에는 대역폭 조절 방법으로 적용할 수 없다. 왜냐하면 출력되는 패킷은 하나의 단위로 전송 속도에 맞추어 전송되어야 하므로, 하나의 패킷을 전송할 때 시간 간격을 두어 분할해서 전송하게 되면 전송된 패킷이 올바르지 않은 패킷이라 인식된다.

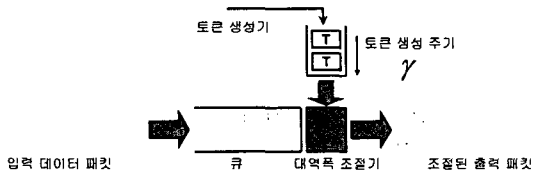

그림 2. 토큰 버킷 방식

리키 버킷과 유사한 트래픽 쉐이핑 기법으로 그림 2와 같은 토큰 버킷이 있다. 토큰 버킷 알고리즘의 핵심은 입력단에서 입력 되어지는 패킷을 순서대로 버퍼에 담는 것은 리키 버킷 방식과 동일하나 출력단에서의 행위는 서로 다르다. 토큰 버킷 방식의 경우에는 장 기간 동안 전송해야 되는 트래픽의 양을 계산하기 위해 조절된 대역폭에 비례하여 주기적으로 생성되는 토큰을 토큰 버킷에 저장하고, 그 토큰의 크레딧(Credit)에 맞추어 패킷을 전송한다. 즉, 특정 패킷을 전송할 수 있을 만큼의 토큰이 토큰 버킷에 충분히 존재하는 경우에는 토큰 버킷에서 필요량 만큼의 토큰을 제거한 후 패킷을 전송하고, 만일 충분하지 않다면 필요량 만큼 토큰이 생성될 때까지 대기하게 된다.

이러한 토큰 버킷의 장점으로는 입력단과 출력단의 전송 속도에 상관없이 적용할 수 있다는 것과 출력 단에서의 필요에 따라 필요한 만큼의 전송 속도 제어가 가능한 동적인 공급을 제공할 수 있다는 것이며 또한 장 기간의 평균 방식이므로 효율적인 대역폭의 활용이 가능하다는 것이다. 그러나 버스트한 데이터 트래픽을 허용하므로 여러개의 링크가 집합되는 지점에서 충돌(Congestion)을 발생시킬 소지가 있으며, 패킷을 위한 버킷과 토큰을 위한 버킷 두 가지를 별도로 관리해야 되므로 구현이 복잡하다는 단점을 지닌다.

그리고 리키 버킷과 토큰 버킷을 함께 사용한 방식은 리키 버킷이 가진 버퍼 외에 트래픽에 비례하는 토큰의 생성 주기와 생성된 토큰을 저장하는 토큰 버킷을 추가로 가진다. 실제 동작인 대역폭의 제어는 토큰 버킷에 축적된 토큰의 양만큼 리키 버킷에서 데이터를 전송함으로써 이루어진다. 이 방식은 토큰의 생성 속도를 조정함으로써 쉽게 동적으로 대역폭을 제한할 수가 있고, 입력단과 출력단의 전송 속도가 동일한 구조에서도 적용이 가능하다. 그러나 이 방식은 두 개의 버퍼를 유지해야 하므로 구성이 복잡해지고, 정확한 토큰의 생성 주기를 맞추기가 힘들 뿐만이 아니라, 긴 기간의 평균 개념에 따른 SLA를 초과하는 트래픽의 생성 가능성을 항상 가지고 있다.

3. 속도 계산 방식

그림 3은 속도 계산 방식의 전체적인 타이밍 차트를 보여주고 있다.

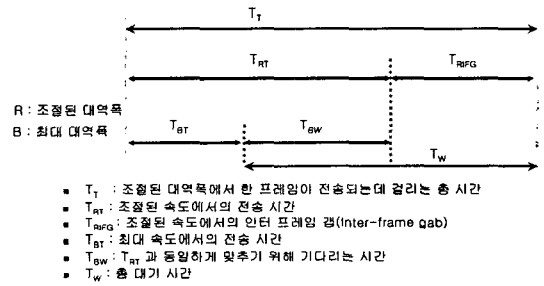

그림 3. 속도 계산 방식의 타이밍 차트

속도 계산 방식은 기다리다 보내는(Send And Wait) 방식으로 다음과 같은 순서로 대역폭의 제어를 수행한다. 먼저 하나의 이더넷 프레임을 전송하는 중간에 해당 프레임이 제한된 대역폭 상에서 전송될 때 걸리는 시간을 계산한다. 그리고 해당 이더넷 프레임의 전송이 끝난 후에 실제 전송된 시간과 계산된 전송 시간만큼 다음 프레임의 전송을 막음으로써 제한된 대역폭으로 전송된 것과 같은 효과가 나타나게 한다. 또한 패딩 존(Padding zone)이 부가된 프레임 버퍼의 관리 방법을 사용하여 가변 크기의 이더넷 프레임을 효율적으로 관리할 수 있도록 한다.

그림 3 에서 보면 비트 사이즈가 α 인 이더넷 프레임이 R Mbps 의 속도로 전송될 때 필요한 시간은 전송 시간 (Transmission Time) + 인터 프레임 갭(Inter Frame Gab)의 시간을 필요로 한다. 그러나 실제 전송은 B Mbps 의 속도로 일어나므로 실제 패킷의 전송은 T_T 에 종료하게 된다. 그러므로 전송은 B Mbps 로 했더라도 R Mbps 의 속도로 전송한 것과 같은 효과를 나타내기 위해서는 패킷의 전송 후에 $T_W + T_{IFG}$ 만큼 대기하여야 한다.

이러한 시간 값들은 B MHz의 PHY Chip 기준으로 보면 아래와 같이 값을 얻을 수 있다.

$T_{BT} = \alpha\ clock$ (α : preamble 부터 FCS 까지의 이더넷 프레임의 총 비트 길이)

$T_{BW} = \alpha \times (B/R - 1)\, clock$

$T_{IFG} = B/R \times 96\, clock$

상기의 수식을 실제 PHY Chip에 대한 입력인 $B/4$ MHz의 MII(Media Independent Interface) 기준으로 보면 아래의 값들을 얻을 수 있다.

$T_{BT} = \alpha/4\, clock$

$T_{BW} = \alpha \times (B/(R \times 4) - 1/4)\, clock$

$T_{IFG} = B/(R \times 4) \times 96\, clock$

이제 상위 식으로부터 T_W를 얻을 수 있다.

$$T_W = T_{BW} + T_{IFG} = \alpha \times (B/(4 \times R) - 1/4) + B/R \times 8 \times IFG\_BYTE = (\alpha + 8 \times IFG\_BYTES) \times (B/(R \times 4) - 1/4) + 2 \times IFG\_BYTES = P\_SIZE \times R\_RATE + \beta$$

여기서

$$P\_SIZE = \alpha + \gamma \quad , \quad \gamma : 8 \times IFG\_BYTES$$

$$R\_RATE = B/(4 \times R) - 1/4.$$

$$\beta = 2 \times IFG\_BYTES$$

이다.

출력단에서 총 대기 시간(Total Waiting Time)의 계산은 특정 패킷의 전송이 종료되기 이전에 이루어져야 한다. 그런데 이더넷 프레임의 최소 크기는 Preamble과 SFD(Start of Frame Delimiter)를 빼고 64 Byte이므로, 이를 MII를 통해 전송하기 위해서는 72 x 2 = 144 MII 기준 Clock 만큼의 시간이 필요하다. 수식의 계산은 R_RATE 의 값이 테이블(Table) 등을 통해 제공된다면 충분히 이 시간 내에서는 이루어질 수 있다.

4. 실험 결과

이 장에서는 속도 계산 방식의 성능과 토큰 버킷 방식과의 비교를 OPNET 으로 시뮬레이션 하고 분석한다.

그림 4는 버퍼에 들어가는 입력 트래픽을 평균 12 Mbps 인 일정 스트링을 평균 6Mbps로 조절하여 출력한 결과이다. 이 실험에서는 단지 서비스 조절 만을 생각하여 버퍼 앞에서의 정책을 고려 하지 않고 무한 버퍼로서 실험을 하였다. 그림 4는 속도 계산 방식을 그리고 그림 5는 토큰 버킷 방식의 결과치이다. 그림 4와 그림 5의 실험 결과에서 보듯이 토큰 버킷 방식에 비해 속도 계산 방식은 출력 부에서 버스트 트래픽을 스태디 스트림으로 변환하여 조절된 트래픽만 통과 시키는 것이 가능하다. 또한 입력되는 트래픽 값에 상관없이 원하는 출력 값만의 설정으로 아주 쉽게 구현이 가능하다. 반면 토큰 버킷은 버킷의 크기와 토큰의 생성시간까지 모두 고려해 주어야 하기 때문에 구현상 많은 복잡한 점을 지닌다.

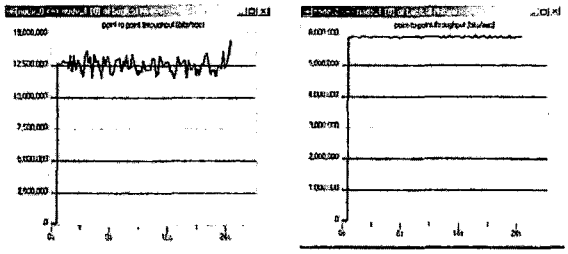

그림 4. 속도 계산 방식 결과

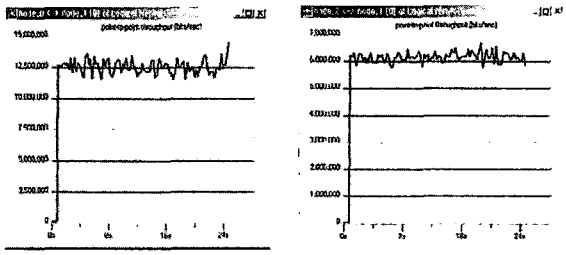

그림 5. 토큰 버킷 방식 결과

5. 결론

본 논문에서는 이더넷에 적용하기 적합한 간단하면서도 효과적인 대역폭 조절 방법인 속도 계산 메커니즘을 제안하였다. 속도 계산 방식은 정확한 결과 값과 단순함 그리고 조절하고자 하는 대역폭의 동적인 변화에 있어서 장점을 지닌다. 무엇보다 현재 라우터나 스위치등의 QoS 기기에 바로 적용 가능하다는 것이 가장 큰 장점이 된다.

본 논문이 제시한 메커니즘을 향후 QoS 기기에 적용시킨다면 네트워크 트래픽 관리 및 운영에 있어서 효율적으로 적용될 수 있을 것으로 기대된다. 본 논문에서는 단순히 대역폭을 제어하는 방식만을 제안 하였지만 차후에는 속도 계산방식에 맞는 버퍼 앞단에서의 큐 매니저(queue manager) 방식과 출력단에서의 흐름 제어 방식까지 순차적으로 연구되어야 할 것이다.

참고 문헌

[1] EFM working group. Home Page for EFM working group, 2001, URL:http://www.manta.ieee.org/groups/802/3/efm/

[2] N.Yin, and M.G.Hluchyj, "Analysis of the Leaky Bucket Algorithm for On-Off Data Sources," in Proc. IEEE GLOBECOM, pp.254-259,1991.

[3] D.S. Holtsinger and H.G.Perros, "Performance analysis of leaky bucket policing mechanisms," Asia - Pacific Engineering Journal, vol. 3, pp. 235-264, September 1993.