Advanced Peer-to-Peer Network에서의 초고속 통신망의 성능연구

황명상, 류제영, 주기호, 박두영
배재대학교 정보통신공학과

A Performance Study on The Advanced Peer-to-Peer Network for Broadband Communications

MyungSang Hwang, Je-Young Ryu, Gi-Ho Joo, Dooyeong Park
Division of Information and Communications Engineering
Paichai University

요 약

본 논문은 leaky bucket을 이용하여 초고속망의 폭주를 제어하는 Advanced Peer-to-Peer Network(APPN)의 성능을 분석하였다. 제안된 논문에서는 손실 및 오류패킷에 대하여 재전송하는 오류제어 방식을 병행하여 사용하였다.

APPN 모델의 성능 분석을 통하여 사용자 차원의 오류제어와 망 차원의 폭주제어 방식간의 상호작용을 연구하여 윈도우 크기와 leaky bucket의 토큰 생성속도가 end-to-end delay에 영향을 미치는 중요한 파라미터들임을 알 수 있었다.

Abstract

In this paper, we carry out a performance study related to the Advanced Peer-to-Peer Network(APPN). For this particular network, it has been proposed to use the leaky bucket as a way of controlling congestion within the network. On the top of leaky bucket type rate based congestion control scheme for high speed networks, a user will typically operate an error control scheme for retransmitting lost and erroneous packets. We propose a performance model in order to study the interaction between a user's error control scheme and the leaky bucket congestion control scheme for high speed networks. Simulation results show that parameters such as the window size and the token generation rate in the leaky bucket are key factors affecting the end-to-end delay.

I. 서 론

APPN의 중요한 특징중의 하나는 광대역 전송을 위해 여러 사용자로부터 발생되는 다양한 형태의 트래픽에 대한 서비스를 동시에 제공할 수 있어야 한다는 것이다. 특히 대역폭이 작은 서비스뿐만 아니라 대역폭이 큰 서비스도 제공해야 한다. 어떤 특정한 트래픽의 형태는 대역할당 방법에 따라 더 그룹화 되어 질 수 있다. 비통계적 대역할당에서는 트래픽의 최고치에 따라 대역이 할당되고 이미 할당된 대역은 사용되고 있지 않더라도 다른 호(call)에게 할당될 수 없다. 반면에, 통계적 대역할당에서는 어떤 호는 특정한 서비스 등급(Grade of Service : GoS)을 보장하기 위해 다른 호들과 할당된 대역을 공유한다. 초고속통신망에서는 모든 패킷이 수신기에 오류나 손실 없이 도착한다고 가정할 수 없기 때문에 사용자 차원의 윈도우 흐름제어[1][2] 역시 망과 사용자/수신자 버퍼에서의 폭주를 방지하기 위해 필수적이다. 따라서 본 논문은 APPN에서 위의 2가지 방식(사용자 차원의 윈도우 흐름제어와 망 차원의 Leaky Bucket을 이용한 폭주제어[3][4][5])의 적용 시에 발생될 수 있는 복잡한 상호작용을 시뮬레이션 수행결과를 통하여 이해함으로서 초고속통신망 설계 시 도움이 될 수 있도록 하는데 그 목적이 있다.

II. Model Description

본 연구에서 고려되어지는 시스템을 자세히 서술하면 다음과 같다. 이 시스템에서 송·수신기는 자신들의 고유한 오류제어 방식을 갖고 있고 송신기에서 발생되는 버스트한 트래픽은 패킷 단위로 전송되며 패킷들은 호 연결 시 설정된 GoS를 보장하기 위해서 Leaky Bucket을 통과하여 망으로 전달되고, 망에 있는 여러 중간노드들을 거쳐서 수신기에 보내진다. 송신기의 패킷은 전송시간에 해당하는 시간만큼의 서비스를 받은 후 Leaky Bucket을 통하여 망으로 보내진다. 이때, 서비스 시간은 패킷의 길이에 비례한다. 송신기는 패킷을 Leaky Bucket으로 보냄과 동시에, 그 패킷의 사본을 ACK를 받을 때까지 윈도우 버퍼에 보관하며 ACK를 받은 후에 그 버퍼에서 제거한다. 이 버퍼가 다 차게되면 송신기는 패킷의 전송을 중단하며 그 버퍼에 여유공간이 생길 때까지 새로운 패킷을 Leaky Bucket으로 전송하지 않는다.

Leaky Bucket에 도착한 패킷은 망으로 들어가기 위해 허가를 얻어야 하고 망 내에서 여러 중간노드를 거쳐서 수신기에 도착하게 된다. 수신기는 도착한 패킷의 일련번호를 체크하여 정상적으로 도착한 패킷을 위한 ACK를 송신 측으로 보낸다. 반면에 손실이나 오류에 의한 비정상적인 패킷에 대해서는 NAK를 송신 측으로 보낸다. 이 과정에서 ACK나 NAK는 일정한 망 전달지연을 갖고 송신 측으로 보내지게 되고 손실이나 오류는 발생하지 않는다고 가정한다. NAK을 수신하게 되면 송신기는 해당 패킷의 재전송을 시도한다. 이때, 재 전송되는 패킷은 우선권을 가지고 송신 버퍼의 제일 앞쪽에 위치하게 된다.

초고속통신망의 단일 가상회선(virtual circuit)을 모델링 하기 위해 직렬로 연결된 여러 노드들로 나타낼 수 있다. 그리고 각 노드들은 유한용량의 단일 서버 큐와 직렬로 연결된 무한 서버 큐로 나타낼 수 있다. 입력 큐는 패킷이 전송되기 전에 서비스를 기다리는 버퍼를 나타내며 서버는 외부 링크를 나타낸다. 여기서, 패킷의 서비스 시간은 geometric 분포를 갖는다고 가정한다. 서비스를 받은 패킷은 전파지연(propagation delay)을 나타내는 무한 서버를 거쳐 다음 노드로 전달된다. 초고속통신망에서는 전파지연은 전송시간에 비해 무척 크기 때문에 end-to-end delay에 많은 영향을 미치므로 무시할 수 없다.

각 노드에서 패킷에 관한 손실은 패킷이 입력 버퍼에 도착하는 시점에 그 버퍼에 저장될 빈 공간이 없는 경우와 패킷에 오류가 발생하여 서버는 패킷을 인식하지 못하고 폐기 처분하는 경우에 생긴다.

더 현실적인 환경을 구현하기 위해 본 연구에서는 중간노드를 공유하는 또 다른 가상회선의 작용을 나타내는 외부 트래픽을 각 노드에 첨가한다. 이 외부 트래픽은 Bernoulli 프로세스이며 각 노드에서 서비스를 받은 후에 곧바로 시스템을 떠난다고 가정한다. 각 노드에 도착한 패킷은 그 패킷의 세그먼트 수만큼의 공간을 차지하게 되고 버퍼의 공간이 충분하지 못하면 폐기된다.

1. Arrival Process

이산 시간적 환경에서는 버스트 트래픽의 모델로 Interrupted Bernoulli Process(IBP)를 많이 사용된다[6]. IBP에서는 휴지상태(Idle State)와 활동상태(Active State)를 가지며, 각 상태의 유지시간은 geometric 분포를 갖는다. 또, 그 프로세스가 활동상태일 때 패킷이 Bernoulli 분포를 갖고 발생하고 휴지상태에서 패킷이 발생하지 않는다. 또한, 여러 버스트한 트래픽들이 중첩된 프로세스의 모델링을 위해서 Markov Modulated Bernoulli Process(MMBP)라는 더 일반적인 프로세스를 고려한다. IBP는 MMBP의 특수한 경우이며, 본 연구에서는 도착 프로세스를 2-state MMBP (2-MMBP)로 가정한다. 2-MMBP는 geometric 분포를 가지고 교대로 발생하는 두 가지 상태가 있으며 각 상태에서 서로 다른 패킷 발생률을 갖는 Bernoulli 프로세스이다[6]. 또한, 본 연구에서는 3가지 크기의 패킷(50, 1000, 2000byte)이 발생하며, 1 세그먼트(segment)는 50 byte이고, 각 패킷의 세그먼트 수만큼의 버퍼 공간을 차지한다고 가정한다.

2. Leaky Bucket

Leaky Bucket은 end-to-end preventive 폭주제어를 위해 사용되며 미리 정해진 트래픽의 평균 흐름 율과 특정한 버스트를 유지하기 위해 송신기로부터 받아들여진 패킷을 망으로 전달하는 기능을 갖고 있다. 토큰을 가진 패킷만 망으로 전달되며 만일 토큰이 Token-pool에 존재하지 않으면 그 패킷은 Leaky Bucket의 입력 버퍼에 저장되어 토큰이 생성될 때까지 기다린다. Leaky Bucket은 유한용량의 Token-pool을 갖고 있어 일정한 속도로 생성된 토큰들은 Token-pool에 저장되고, 만일 Token-pool이 다 차게되면

Token-pool에 빈 공간이 생길 때까지 계속 생성되는 토큰들은 버려진다.

또한, APPN에서는 ATM망과는 달리 패킷이 요구하는 토큰의 양은 각 패킷의 크기에 따라 달라진다. 패킷들이 망으로 들어가기 위해 요구하는 토큰의 양은 그 패킷의 세그먼트 수와 같다. 남아있는 입력 버퍼의 공간이 패킷의 세그먼트 수 보다 적으면 그 패킷은 버려지게 된다.

3. Error Control Scheme

광섬유를 전송매체로 하는 초고속통신망에서는 오류발생 확률이 매우 적다(10^{-10}). 이러한 관점에서 보면 오류제어는 end-end 차원에서 이루어지는 것이 타당하다. 본 연구에서는 HDLC나 OSI 계층 2에서 적용되는 Selective Reject ARQ(SREJ ARQ)방식[2]을 채택한다.

III. Simulation Model and Results

시뮬레이션을 통하여 고찰될 시스템의 상세한 모델을 서술해 보면 다음과 같다. 송·수신기는 SREJ ARQ 오류제어 방식을 사용하며 송·수신기의 버퍼는 무한용량을 갖고 망 내에서는 2개의 중간노드가 있다고 가정한다. 본 연구에서는 도착 프로세스, 윈도우크기, Token-pool 크기, 토큰생성속도, 각 버퍼의 크기, 전송속도 등을 시뮬레이션의 입력 파라미터로 사용함으로서 모든 가능한 망의 환경 변화와 각 파라미터 값의 변화에 따른 망의 동작 및 영향에 대한 분석이 가능하도록 하였다. 또한, 본 연구에서 송신기 및 노드에서의 전송시간은 각각 100Mbps, 150Mbps이고, 노드와 노드간의 전파지연은 5msec이라고 가정한다. 또 ACK/NAK는 송·수신기간의 전파지연(10msec)만 존재하며 처리 및 전송시간, 전송과정상의 손실이나 오류는 무시한다. 재 전송되는 패킷은 Non-preemptive head-of-line (HOL)방식의 우선권이 주어진다고 가정한다. 그리고 패킷의 크기가 일정한 일반적인 ATM망에서의 전송방법과 달리 해당 패킷의 모든 세그먼트가 존재할 때만 그 패킷의 전송이 시작된다. 이 가정은 노드-노드(Node-to-Node)에서의 전송뿐만 아니라 모든 전송과정에서 적용된다.

Table 1~4는 다음과 같은 조건하에서 Leaky Bucket에서의 토큰 생성속도와 윈도우크기에 따른 end-to-end delay, 각 버퍼에서의 지연 및 패킷손실확률의 변화를 보여준다.

- 망 내의 각 노드의 입력 버퍼의 크기
  = 160 segments
- Leaky Bucket의 입력 버퍼의 크기
  = 600 segments
- Token-pool의 크기 = 400 tokens
- 송신기의 평균대역폭
  = 0.01 packets/slot

본 연구에서 사용된 2가지 토큰 생성속도는 송신기의 평균 처리율에 1.1 (Table 1과 2) 또는 1.3 (Table 3과 4)을 곱하여 얻는다. 또한, Table 1~4에서 토큰생성속도와 윈도우크기가 각 버퍼에서의 지연 및 패킷 손실 확률에 미치는 영향에 대하여 보여준다. 이때, 사용된 윈도우크기는 70, 50, 40, 30 이다. Table 1과 4에서는 윈도우크기의 변화에 대한 end-to-end delay와 각 버퍼에서의 지연을 보여준다. 윈도우크기가 작아지면 송신 버퍼에서의 지연이 커지고, 반면에 윈도우크기가 커지면 Leaky Bucket의 입력 버퍼에서의 지연이 커진다. 이 결과는 각 큐에서의 패킷손실확률을 나타내는 Table 2와 3를 보면 더욱 명백해 진다. 윈도우크기가 증가하면 Leaky Bucket의 입력 버퍼에 패킷들이 쌓여 패킷손실확률이 증가한다.

IV. Conclusion

본 연구에서는 Leaky Bucket을 이용한 망 내부의 폭주를 제어방식을 고려하였다. 그리고, 각 송·수신기는 사용자 차원의 윈도우를 이용한 흐름제어 방식과 SREJ의 오류제어 방식을 가지고 구동된다. 송신기에서의 프로세스는 버스트 특성을 고려하여 MMBP로 모델링 하였고, 3가지의 크기의 패킷을 동일한 확률을 가지고 발생시킨다고 가정하였다. 또한, 보다 현실적인 망의 환경을 모델링하기 위해서 망 내부에서 성능에 영향을 미치는 외부 트래픽의 흐름을 첨가하였다.

시뮬레이션 결과를 통하여 end-to-end delay의 대부분은 망의 전파지연에 의한 것이며, 윈도우크기와 토큰생성속도에 많은 영향을 받는다는 것을 알 수 있었다. 즉, 윈도우크기가 작으면 송신 버퍼에서의 지연이 커지고 반면에, 커지면 Leaky Bucket에서의 지연이 커진다. 이 결과는 망 전체를 통하여 각 버퍼에서의 지연에 중대한 영향을 미친다.

Token Generation Time = 0.0356751 msec				
Window Size / Delay	70	50	40	30
End-to-end Delay	10.910401	10.844379	10.715546	10.628714
Delay at Sender 1	0.093104	0.106122	0.108277	0.414381
Delay at Sender 2	0.104817	0.112274	0.112537	0.414322
Delay at Sender 3	0.087883	0.095924	0.123209	0.433307
Delay at Leaky Bucket 1	0.511850	0.497512	0.369036	0.073945
Delay at Leaky Bucket 2	0.792743	0.735059	0.542548	0.098752
Delay at Leaky Bucket 3	0.386657	0.378739	0.329975	0.076092
Delay at Node 1	0.074017	0.073877	0.073871	0.072789
Delay at Node 2	0.073648	0.073613	0.073602	0.072636
Delay at Receiver 1	0.098528	0.033274	0.022226	0.000000
Delay at Receiver 2	0.104981	0.062598	0.025500	0.000000
Delay at Receiver 3	0.052043	0.052043	0.051100	0.005543

Table 1 : End-to-end delay와 각 큐에서의 delay
(토큰 생성시간 = 0.0356751 msec)

Token Generation Time = 0.0356751 msec				
Window Size / Blocking Prob.	70	50	40	30
Blocking prob. at Leaky Bucket 1	0.000150	0.000030	0.000000	0.000000
Blocking prob. at Leaky Bucket 2	0.000215	0.000082	0.000000	0.000000
Blocking prob. at Leaky Bucket 3	0.000000	0.000000	0.000000	0.000000
Blocking prob. at Node 1	0.000086	0.000086	0.000086	0.000011
Blocking prob. at Node 2	0.000005	0.000005	0.000005	0.000005

Table 2 : 각 큐에서의 패킷손실률
(토큰 생성시간 = 0.0356751 msec)

Token Generation Time = 0.03026482 msec				
Window Size / Blocking Prob.	70	50	40	30
Blocking prob. at Leaky Bucket 1	0.000000	0.000000	0.000000	0.000000
Blocking prob. at Leaky Bucket 2	0.000000	0.000000	0.000000	0.000000
Blocking prob. at Leaky Bucket 3	0.000000	0.000000	0.000000	0.000000
Blocking prob. at Node 1	0.000097	0.000097	0.000162	0.000022
Blocking prob. at Node 2	0.000005	0.000005	0.000005	0.000005

Table 3 : 각 큐에서의 패킷손실률
(토큰 생성시간 = 0.03026482 msec)

Token Generation Time = 0.03026482 msec				
Window Size / Delay	70	50	40	30
End-to-end Delay	10.338020	10.339859	10.399580	10.599486
Delay at Sender 1	0.087840	0.089232	0.124549	0.414517
Delay at Sender 2	0.088659	0.089087	0.102173	0.399300
Delay at Sender 3	0.087849	0.091794	0.132722	0.427383
Delay at Leaky Bucket 1	0.067753	0.064907	0.064902	0.057970
Delay at Leaky Bucket 2	0.066718	0.066714	0.065799	0.057639
Delay at Leaky Bucket 3	0.063492	0.059089	0.059037	0.057509
Delay at Node 1	0.074490	0.074473	0.074426	0.072856
Delay at Node 2	0.074144	0.074115	0.074011	0.072682
Delay at Receiver 1	0.028995	0.028995	0.063148	0.007807
Delay at Receiver 2	0.019604	0.019604	0.044707	0.000000
Delay at Receiver 3	0.046560	0.046560	0.068721	0.005543

Table 4 : End-to-end delay와 각 큐에서의 delay
(토큰 생성시간 = 0.03026482 msec)

참고문헌

[1] Data and Computer Communications, W. Stallings, Macmillan, 1995

[2] Data Communications, Computer Networks and Open Systems, F. Halsall, Addison -Wesley, 1992

[3] K. Bala, I. Cidon, and K. Sohraby,"Congestion control for high speed packet switched networks", Technical Report, IBM Watson Research Center, 1990.

[4] H. Ahmadi, R. Guerin, and K. Sohraby, "Analysis of leaky bucket access control mechanism with batch arrival process", Globecom'90, 1990.

[5] D. Holtsinger and H. G. Perros, "Performance analysis of leaky bucket policing mechanisms", Asian-Pacific Engineering Journal (Part A), vol 3, No. 3, pp. 235-264, 1993.

[6] W. Fischer and K. Meier-Hellstern, "The MMBP cookbook," Technical Report, Bell Lab., 1990.