2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

Active 라우터의 피드백 메커니즘을 이용한 혼잡제어

*최 기 현, *신 호 진, **김 병 기, *신 동 렬
*,**성균관대학교 전기전자 및 컴퓨터 공학과
** 에스넷시스템㈜
*e-mail : tomahawk@nova.skku.ac.kr

Active Congestion Control Using Active Router's Feedback Mechanism

*Ki-Hyun Choi, *Ho-Jin Shin, **Byong-Ki Kim, *Dong-Ryeol Shin
*,**Dept. of Electrical and Computer Engineering, Sungkyunkawn University
**S Net Systems Inc.

요 약

기존의 end-to-end방식에서는 네트워크 내부에서 혼잡(congestion)이 발생했을 경우 각 endpoint에서 즉시 알아 낼 수 없기 때문에 일정시간 동안 수신된 packet의 순서에 대한 정보로 혼잡이 발생했는지에 대해 추론하는 것이다. 따라서 네트워크 내부에서 직접적으로 정보를 얻거나 처리를 해 줄 수 있다면 기존 방식보다 처리율이 향상될 것이다. 본 논문에서는 Active Network 메커니즘을 이용하여 라우터를 구성하고 관련 모의실험결과를 보인다. 이를 통해서 기존의 방식보다 향상된 혼잡제어를 제시한다.

1.서론

혼잡 제어에 있어 Active Networking(이하 AN)기술의 이용은 네트워크 혼잡에 보다 신속하게 응답하게 해준다. 기존의 end-to-end 피드백(피드백) 혼잡제어는 오직 단말노드에서만 혼잡을 감지하고 제어할 수 있다. 따라서 AN 기술을 이용하여 혼잡을 네트워크 내부에서 직접적으로 처리해 줄 수 있다면 네트워크 성능을 향상 시킬 것이다[1].

AN 기술은 packet내에 프로그램이나 처리에 필요한 정보를 담아 전송하게 되는데 네트워크상에 있는 노드는 이 packet(active packet 또는 smart packet[7])을 읽어 들여 필요한 처리를 할 수 있다. 또한 네트워크 내부 노드들은 서로에게 필요한 네트워크 상의 정보를 공유할 수 있다. 이 기술을 사용할 경우 혼잡이 발생하면 네트워크 내부에서 이를 감지하여 주변 라우터에 혼잡에 대한 정보를 알려줄 수 있으며 직접적으로 혼잡을 제어할 수 있다[3]. 반면 기존의 혼잡제어 방식은 수신측과 송신측과의 지연 즉 RTT(Round Trip Time)가 클 경우 그 만큼 혼잡을 제어하는데 시간이 더 오래 걸리게 된다. 또 네트워크의 대역폭 클 경우 혼잡이 감지되어 어떤 처리를 시작할 때까지 이미 전송된 packet으로 인해 혼잡이 더욱 가중되게 된다. 따라서 본 논문에서는 혼잡처리를 네트워크 내부로 가져와 처리를 하기위해서 active 라우터를 구성하고 혼잡제어를 보이겠다. 2장에서 AN 기술을 이용한 혼잡제어에 대한 개념을 제시하고, 3장에서는 Active 라우터의 구성과 동작 방법을 제시하고, 4장에서는 모의실험의 결과 및 분석을 보이고 5장을 결론으로 끝을 맺는다.

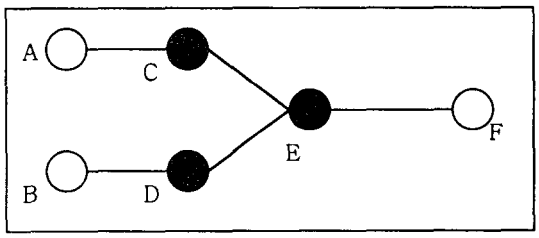
2. Active Congestion Control의 개념

혼잡이 발생했을 경우 라우터가 혼잡에 참여하는

트래픽에 대한 정보를 저장하고, 주변 라우터에 혼잡발생에 대한 정보를 전송하여 혼잡에 대한 처리를 해줄 수 있다면 endpoint에서 즉각적인 처리를 하는 것과 같은 효과를 얻을 수 있을 것이다.

이에 앞서 ACC[1]에서 제시한 혼잡제어를 이해하고 이보다 확장된 혼잡제어를 위한 Active 라우터의 구성을 보이겠다.

ACC[1]는 보다 신속하게 혼잡에 반응하기 위해 혼잡이 발생한 라우터와 주변 라우터와의 통신을 이용하는 방법을 제안하였다. 하지만 모의실험에서는 라우터와의 통신을 이용해서 혼잡을 제어하지 않고 단지 혼잡이 발생한 라우터에서 " Drop and Notice" 방법을 이용하여 혼잡을 제어 하였다.



<그림 1> ACC에서 제안한 혼잡된 네트워크 구성

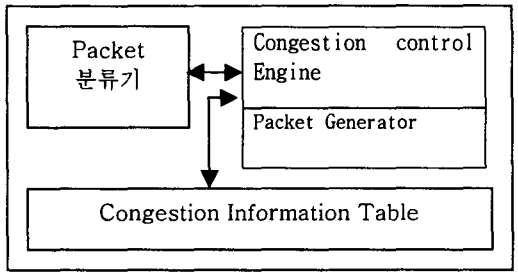
<그림 1>에서 노드 E는 혼잡이 발생하는 라우터이며 노드 C,D는 노드 E로부터 혼잡정보를 받아 혼잡에 관계된 트래픽을 필터링하게 된다. A에서 F까지의 RTT가 100ms이고 A~C까지의 지연 시간이 10ms라고 하면 혼잡 발생시에 기존의 end-to-end 처리방식보다 80ms정도 빠르게 처리가 가능하게 된다. 최초 E에서 C또는D로 혼잡정보를 전송할 때를 제외한다면 노드 C와D는 그 이후에 들어오는 트래픽을 20ms의 지연시간으로 처리가 가능하게 된다. 모의실험에서 라우터 E는 혼잡이 발생하면 E로 들어오는 packet에서 현재의 윈도우의 크기를 active packet내에 저장하여 각 endpoint로 피드백하게 된다. 각 endpoint는 피드백된 active packet에서 윈도우 크기를 읽어들여 새로운 윈도우 크기를 계산하게 된다.[4] 새로운 윈도우크기를 계산하여 보내기 전까지 혼잡이 발생한 라우터에서는 drop될 당시의 윈도우크기의 절반의 수까지 그 트래픽에 대해서 필터링(filtering)을 하게 된다. 주변 라우터 C와D는 혼잡제어에 어떤 영향도 주지 않는다. 비록 주변 라우터와 통신을 하지 않더라도 18%정도의 성능향상을 보였다[1]. 본 논문에서는 TCP Tahoe[9]이외에 TCP Reno[9]와 TCP Vegas[6]에도 같은 메커니즘을 적용하여 성능상의 이점을 보이겠다.

3. Active 라우터 Architecture

3.1 Active 라우터의 구성요소

① Congestion Control Engine
② Packet 분류기
③ Congestion Information Table
④ Packet Generator

packet 분류기는 active packet과 일반 packet을 분류하여 active packet일 경우 그 정보를 CIT (Congestion Information Table)에 저장한다. 이 정보를 이용하여 주변 라우터에 혼잡에 대한 정보를



<그림 2> Active 라우터의 구성

제공할 수 있다. Packet Generator는 라우터간에 정보를 공유할 때 사용하게 된다.

<그림 2>에서 구성한 각 모듈은 모의실험에서 하나의 함수로 구현되었다. 범용으로 혼잡을 제어하기 위해 각 active 라우터는 CCE(Congestion Control Engine)를 갖는다. 비록 현재의 작업은 단순히 drop된 정보를 이용해서 혼잡을 제어하게 되지만 차후에 QoS 정책에 따른 혼잡제어나 각 라우터에서의 통계정보를 이용할 수 있는 발판이 될 것이다.

3.2 Active 라우터의 동작 방법

Active 라우터는 drop이 발생하면 drop된 packet에 대한 정보를 CIT에 저장하고 이 정보를 이용하여 혼잡을 제어한다.

src	dst	window	Drop	Id	buffer		...	...
0	13	14	6	1	22			
3	16	22	2	2	20			
8	21	13	6	3	22			
					20			
4	15	14	5	8	24			
7	20	16	7	9	22			

<표 1> CIT에 저장되는 정보

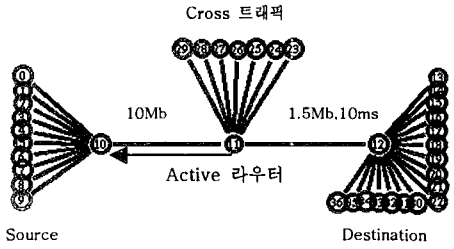
<표 1>은 <그림 1>에서 노드 E의 CIT에 저장되는 정보를 나타낸다. src와 dst는 트래픽의 출발지와 목적지를 나타낸다. 윈도우는 drop될 때의 윈도우크기이고 packet을 필터링할 때 사용되는 정보이다. Drop은 현재까지 노드 E에서 필터링되어 전송되지 않고 drop된 packet의 개수다. 이 drop된 개수가 일정 값을 넘게 되면 더 이상 필터링되지 않는다. 이

렇게 하는 이유는 drop되는 packet이 많아 질 경우 성능이 오히려 떨어지는데 그 이유는 네트워크의 대역폭이 클 경우 이미 네트워크에 들어와 있는 packet의 수가 많기 때문에 계속해서 drop을 할 경우 다시 전송하는데 많은 시간이 걸리기 때문이다. Buffer field는 노드 E의 큐(queue)에 들어있는 packet의 개수다. 이 정보를 주변 라우터 C,D에 일정 간격마다 피드백하면 이를 이용해서 혼잡을 미리 제어할 수 있다. 단순히 이 정보를 이용해서 주변 라우터에서 drop할 경우에는 오히려 drop되는 수가 많아져 성능이 떨어지기 때문에 노드 E는 피드백 후에 필터링을 하지 않고 노드 C와 D에서 하도록 한다. <표 1>에서 " … " 부분은 나중에 혼잡 제어에 필요한 정보를 저장하기위해서 남겨둔 공간이다.

4. 모의실험결과

4.1 모의실험환경

본 논문에서는 NS[5]를 이용하여 모의실험을 진행하였다. Active 라우터를 구성하기 위해서 NS에서 제공하는 Queue관련 모듈과 TCP 소스를 분석하여 Active packet에 대한 처리를 가능 하도록 수정하였다.



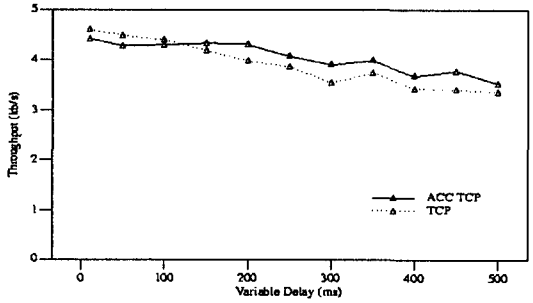
<그림 3> 모의실험에 사용한 topology

<그림 3>은 모의실험에서 사용한 네트워크 구성이다. 그림에서 ⑪는 Active 라우터로 동작하게 된다. ACC와 동일한 결과를 얻기 위해서 ⑩을 일반 라우터로 동작시키고 ⑩과 ⑪사이의 지연시간을 변화시켜 가면서 성능을 측정하였다. 여기서 endpoint ° ~ ⑨에서 ⑩까지의 지연 시간은 10ms이고 대역폭은 각각 1Mb로 고정시켜서 모의실험 하였다. 각 endpoint ° ~⑨는 모의실험 시작부터 burst하게 packet을 전송한다. 라우터 ⑪과 ⑫는 1.5Mb의 대역 폭을 갖게 하였다. 노드 ⑪의 queue에서 drop하는 방법에는 Droptail과 RED 방법을 사용하였다. Droptail은 큐가 다 찼을 경우 제일 마지막 packet을 drop시킨다. RED(Random Early Detection)[8]는 큐가 다 차기 전에 미리 무작위로 drop시키기 때문에 Droptail보다 처리율이 높게 나타났다. 모의실험에서 각 라우터의 큐길이는 25이고, 모의실험 시간은 200초이고 10번의 수행 결과를 평균 내었다. Cross 트래픽은 Drop에 대한 처리를 하지 않는 실시간 트래픽으로 본 논문의 혼잡제어 방식이 혼잡제어를 하지 않는 트래픽과 특정 node에서 만나서 혼잡을 발생 시킬 경우에 성능에서의 차이점을 관찰하기 위해서 구성하였다.

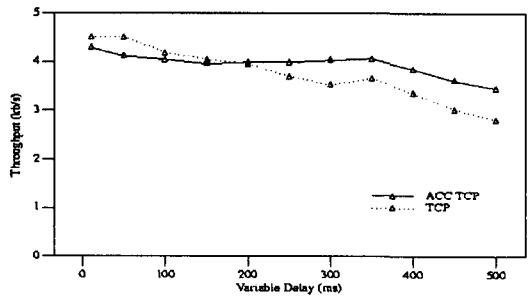
4.2 모의실험결과

Active 라우터를 이용해서 혼잡을 제어할 경우 Tahoe[9]나 Reno[9] 그리고 Vegas[6]모두 혼잡이 발생한 라우터에서 endpoint<▪~⑨>까지 지연시간이 짧을 경우 네트워크상에 전송되어있는 packet의 수가 상대적으로 적다. 따라서 Drop되는 packet이 차지하는 비율이 상대적으로 커지게 되므로 <그림 4>~<그림 9>와 같이 성능이 떨어지게 된다. 하지만 지연시간이 클 경우에는 혼잡을 발생시키는 트래픽의 양이 많아 지게 되므로 성능이 향상되었다.

• TCP Tahoe와 ACC TCP Tahoe

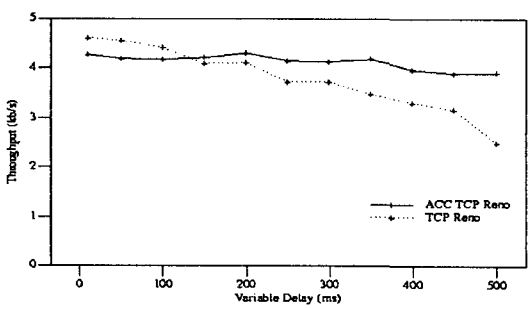


<그림 4> TCP Tahoe와 ACC TCP Tahoe(Droptail)

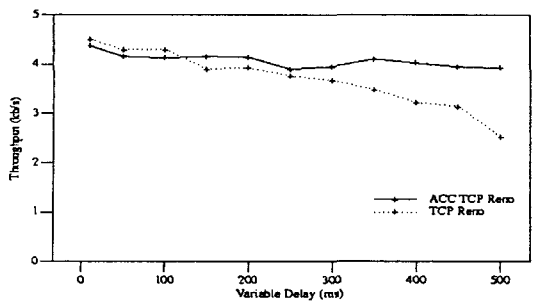


<그림 5> TCP Tahoe와 ACC TCP Tahoe(RED)

• TCP Reno와 ACC TCP Reno

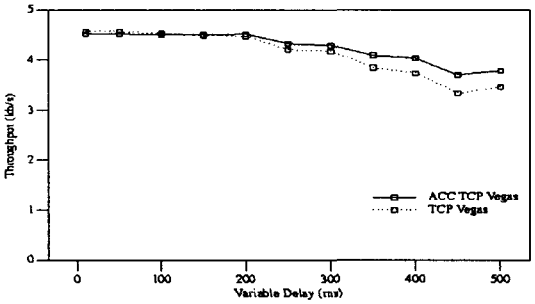


<그림 6> TCP Reno와 ACC TCP Reno(Droptail)

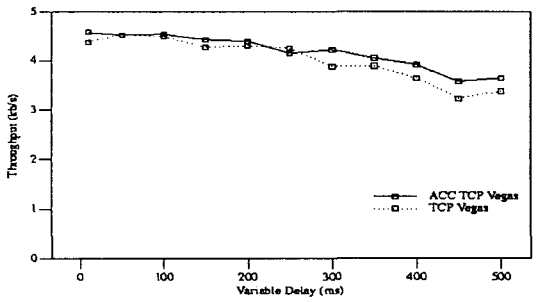


<그림 7> TCP Reno와 ACC TCP Reno(RED)

• TCP Vegas와 ACC TCP Vegas



<그림 8> TCP Vegas와 ACC TCP Vegas(Droptail)



<그림 9> TCP Vegas와 ACC TCP Vegas(RED)

5. 결론 및 향후과제

모의실험에서와 같이 Active 라우터를 이용해서 혼잡을 제어할 경우 기존의 혼잡제어의 비효율성을 제거하여 전체적으로 성능이 향상됨을 알 수 있었다. 본 논문에서 각 TCP 혼잡제어 방식마다 단순히 Drop 시키고 이 정보를 각 endpoint에 알려주는 " Drop and Notice" 을 이용하여 혼잡을 제어하였다. 이것은 네트워크 내부에서의 정보를 이용할 수 있다면 혼잡제어에 보다 신속하게 처리할 수 있다는 것을 보여주는 것이다. 만약 라우터에서 drop된 정보이외에 라우터 내부의 통계적 정보나 연결된 트래픽에 대한 정보, 큐에 저장된 packet의 개수 등을 이용해서 혼잡을 제어 한다면 보다 안정적이고 효율적인 제어가 가능할 것이다. 또한 실시간 트래픽과 비실시간 트래픽의 우선 순위를 정해서 혼잡을 제어한다면 QoS를 보장하는 혼잡제어를 할 수 있을 것이다. 실시간 트래픽은 일단 drop이 혼잡이 발생되어 drop되면 제대로 된 서비스를 받지 못하기 때문에 이 트래픽에 대해서는 다른 혼잡제어 방법을 사용하는 것이 좋을 것이다.

앞으로 본 논문의 모의실험 결과를 토대로 전체적인 라우터의 구성을 확장하여 모든 트래픽에 대해서 범용적으로 혼잡제어가 가능한 라우터 구성을 목표로 연구를 할 것이다.

참고 문헌

[1] University of Southern California, ISI, Theodore Faber, IEEE Network (pp.61~65), " ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanisms" .

[2] Y. Yemini and S. de Silva, " Towards Programming Networks" , IFIP/IEEE, (pp.5~18), 1996.

[3] S. Bhattachariee, K. Calvert, and E. Zegura, " On Active Networking and Congestion" Technical Report GIT-CC-96-02, College of Computing, Geargia Tech, Atlanta, GA 1996.

[4] V. Jacobson, " Congestion Avoidance and Control" , ACM SIGCOMM, Stanford (pp.314~329), Aug. 1998.

[5] S. McCanne, S. Floyd, and K. Fall, UCB/LBL Network Simulator NS 1996.

[6] L.S Brakmo,S.W.O' Malley,and L.Peterson, " TCP Vegas: New techniques for congestion detection and avoidance" , Proc. ACM SIGCOMM Symp. (pp.24~35), Aug. 1994.

[7] Alexander, S., Gunter, C., Keromytix, A., Minden, G., Wetherall, D., Braden, B. and Jackson, A. (1997) " The Active Network Encapsulation Protocol(ANEP)" .

[8] S. Floyd and V. Jacobson, " Random Early Detection gateways for Congestion Avoidance" , IEEE/ACM Trans. an Networking, vol.1, no. 4, (pp.397~413), Aug. 1993.

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