TCP 혼잡제어를 위한 RTT(Round trip time) 측정

김 은 기†

요 약

TCP 프로토콜의 혼잡 제어 알고리즘은 인터넷 망 내로 들어오는 트래픽 량을 조절하여 망이 혼잡 상태에 빠지는 것을 방지하는 기능을 수행한다. 따라서, 망에서는 어떤 TCP에서 발생하는 트래픽이 표준 TCP 흐름 제어 알고리즘을 따르고 있는지 감시할 필요가 있다. 이러한 기능을 수행할 수 있는 몇 가지 방안들이 제안되었으나 이들은 TCP 플로우 (flow)의 RTT (round trip time)를 알지 못하여 실제로 사용될 수 없는 문제를 갖고 있다. 본 연구에서는 인터넷 망 내 라우터에서 각 TCP 플로우의 RTT 값을 측정할 수 있는 알고리즘을 제안하고, 시뮬레이션을 통하여 알고리즘의 올바른 동작을 확인하였다.

Measurement of RTT for TCP Congestion Control

Eun-Gi, Kim†

ABSTRACT

TCP congestion control algorithm prevents network congestion through the control of outgoing traffic size. The network, therefore, should monitor the incoming traffic size of a TCP to determine whether or not a TCP follows standard congestion control algorithms. Some TCP friendly test algorithms are proposed. But, these algorithms cannot be used in real environments because a router in a network does not know the RTT of a TCP flow. In this study, we propose a new RTT determination algorithm that can be used in a router. Our proposed algorithm is validated through the simulation studies.

1. 서 론

인터넷의 활용이 활발해짐에 따라서 인터넷 망 내부에서 처리하여야 하는 트래픽(traffic)의 량이 급속히 증대되고 있다. 그런데, 망 내부로 들어오는 트래픽 량이 일정 수준 이상으로 증가하면 망이 혼잡 상태(congestion)에 빠져 성능이 급속히 감소된다[1]. 따라서 각 종단 단말에서는 망 내의 혼잡 상태 정도에 따라서 망 내부로 보내는 트래픽 량을 조절하여야 한다. 종단 단말기에서 이러한 기능은 TCP 프로토콜에 의해서 수행된다[11].

TCP 프로토콜은 망이 혼잡 상태에 있는 경우 망으로 데이타를 보내는 속도를 줄여 망이 혼잡 상태에서 빠져 나올 수 있도록 지원하고 있다. 그러나, TCP 프로토콜을 구현할 때 이러한 기능을 약화시키거나 제거하면 단말의 사용자에게는 사용하는 TCP 프로토콜의 성능이 우수한 것으로 인식될 수 있다. 이러한 단말기의 수가 많아지게 되면 망 내부의 혼잡을 가속화하여 전체 인터넷 망이 일시적으로 붕괴되는 상황으로 발전할 수 있다[1, 2]

따라서 망 내부에서는 데이터를 전송하는 각 TCP 프로토콜이 표준 혼잡 제어 방식을 따르고 있는지 감시할 필요가 있으며, 이에 관련된 여러 가지 연구가 수행되고 있다. 이들 중 대표적인 것이 Sally Floyd의 연구로써, 이 연구에서는 TCP 프로토콜이 표준 혼잡

※ 본 논문은 대전산업대학교 '99년 교내 연구비의 지원을 받아 수행되었음.
† 정 회 원 : 대전산업대학교 정보통신컴퓨터공학부 교수
논문접수 : 1999년 12월 17일, 심사완료 : 2000년 5월 17일

제어 방식을 지원하고 있는가를 감시할 수 있는 방식을 제안하였다[2]. 이에 따라서 망 내 라우터는 각 TCP 트래픽을 조사하여, 망이 혼잡 상태에 빠질 가능성이 있는 경우 표준 혼잡 제어 방식을 따르지 않는 트래픽을 먼저 제거함으로써 종단 단말기의 TCP에서 혼잡 제어 알고리즘을 충실히 이행할 수 있도록 하고 있다. 그러나, 이 연구 결과가 실제 상황에서 사용되지 못하는 이유는 망 내부에서 각 TCP 플로우의 RTT(round trip time)를 알지 못하기 때문이다.

본 연구에서는 망 내부의 라우터가 각 TCP 플로우의 RTT를 측정할 수 있는 알고리즘을 제시하고, 시뮬레이션을 통하여 제시된 알고리즘의 타당성을 검증하였다. 본 논문은 2장에서 제안된 RTT 측정 알고리즘을 제시하고, 3장에서 제시된 알고리즘을 적용한 시뮬레이션과 그 결과에 관하여 기술하며, 마지막으로 4장에서 결론을 기술한다.

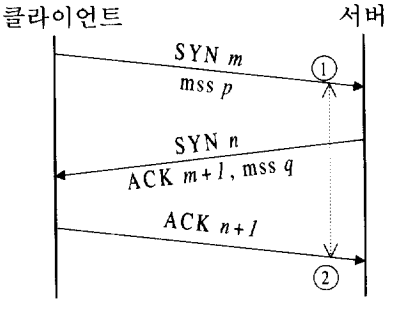
2. RTT 측정 알고리즘

[2]에서 floyd는 TCP 혼잡 제어 알고리즘을 충실히 수행하는 연결의 경우 트래픽 발생량이 식 1을 만족시키는 것을 증명하였다.

$$T \le \frac{1.5\sqrt{2/3}\times B}{R\times\sqrt{p}} \quad \cdots \quad (1)$$

식 (1)에서 T는 정상적인 TCP 플로우에서의 최대 전송률(Bps), B는 최대 패킷의 크기(byte), R은 최소 라운드 트립 시간(RTT, round trip time) 이며 p는 패킷의 드롭률(drop rate)이다. 따라서, 어떤 TCP 플로우에서 식 (1) 이상의 트래픽이 발생하는 경우 이 연결은 TCP 프로토콜의 혼잡 제어 알고리즘을 따르지 않는 트래픽으로 판단할 수 있다. 식 (1)에서 B, p 등은 쉽게 측정될 수 있으나, R 값은 쉽게 측정되기 어렵다. 본 논문에서 제안된 RTT 값 측정 방식은 TCP의 연결 설정 과정을 이용하는 것으로 망 내 라우터에서 어떤 TCP 플로우의 RTT 값을 측정하는데 이용될 수 있다.

TCP 프로토콜의 연결 설정 과정에서 클라이언트와 서버 사이에 세그먼트(segment)들이 교환되는 과정을 간단히 살펴보면 (그림 1)과 같다. (그림 1)에서 SYN으로 표시한 것은 TCP 세그먼트의 SYN(Synchronize sequence numbers) 비트가 1로 세트된 것을 나타내며, ACK로 표시한 것은 ACK(Acknowledgment field significant) 비트가 1로 세트된 것을 의미한다.



(그림 1) TCP 프로토콜의 연결 설정 과정

(그림 1)에서 알 수 있듯이 연결 설정 과정이 시작되면 클라이언트는 서버에게 자신이 사용할 초기 순서 번호(m)와 최대 세그먼트 크기(p)를 전송하며, 서버는 이에 응답하여 자신이 사용할 순서 번호(n)와 ACK 정보($m+1$), 세그먼트의 크기(q) 등을 전송한다. 이를 수신한 클라이언트는 ACK 정보($n+1$)를 서버에게 전송함으로써 연결 설정이 완료된다.

이러한 과정은 모든 TCP 연결에서 동일하게 발생하며, 망 내의 라우터는 클라이언트가 서버에게 전송하는 첫 번째 세그먼트를 수신한 후 연결 설정 과정의 마지막 세그먼트를 수신할 때 까지의 시간을 측정함으로써 RTT를 알 수 있게 된다. 예를 들면, (그림 1)에서 점선으로 표시된 시간 간격이 RTT로 측정될 수 있다.

망 내의 라우터는 인터넷 데이타그램(IP 패킷)의 데이터 영역에 있는 TCP 프로토콜의 헤더 부를 검사함으로써 이러한 과정을 수행할 수 있다.

3. 시뮬레이션

본 연구에서 제안된 RTT 측정 알고리즘의 시뮬레이션은 가상 네트워크 테스트 환경을 구축하려고 시

도되고 있는 VINT 과제에서 구현한 ns-2 시뮬레이터를 이용하였다[7]. 이를 위하여 ns-2의 원시 코드를 약간 수정하였으며 이에 관련된 자세한 내용은 부록에 나타나 있다.

Ns-2를 이용한 시뮬레이션 환경은 (그림 2)에 나타나 있다. (그림 2)에서 라우터들 사이의 연결이나 송신측/수신측 노드와 라우터와의 연결에 사용된 모든 링크는 10Mb의 대역폭과 10ms의 전송 지연 시간을 갖는 것으로 구성하였다.

(그림 2) 시뮬레이션 환경

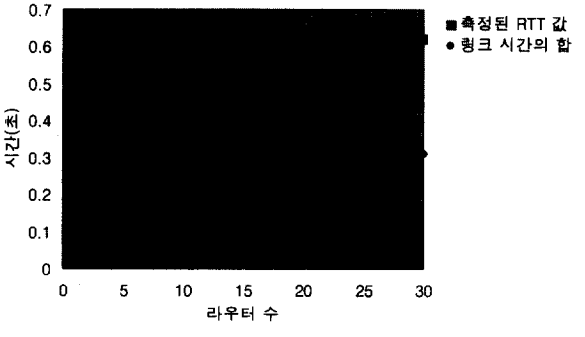
본 연구의 시뮬레이션은 RTT 값 측정에 관련된 내용과 측정된 RTT 값을 식 (1)에 대입하여 계산된 값과 실제의 시뮬레이션 결과와 비교하는 두 가지 내용으로 이루어졌다. (그림 3)은 RTT 값의 측정에 관련된 시뮬레이션 결과를 보여 주고 있다.

(그림 3a)는 라우터 수가 증가함에 따라서 RTT 값이 선형적으로 증가하는 모습을 보여주고 있다. 이 실험에서는 하나의 응용에서만 트래픽이 발생하며 링크 대역폭보다 적은 량의 트래픽이 발생되도록 구성하였다. RTT 값 측정은 첫 번째 라우터에서 이루어졌으며, 다른 라우터에서 측정한 결과나 다른 종류의 TCP와 큐를 이용하여 측정한 결과도 (그림 3a)와 거의 동일한 결과를 나타내었다. (그림 3b)에서는 망 내의 트래픽 량을 증가시키면서 측정한 RTT 값을 보여 주고 있다. 이 실험에서는 전체 라우터 수를 15개로 하였으며, 11번 라우터에서 측정이 이루어졌다. 망 내의 트래픽이 증가함에 따라서 RTT 측정에 필요한 시간 간격은 길어졌으나, 그 값은 항상 일정하였다.

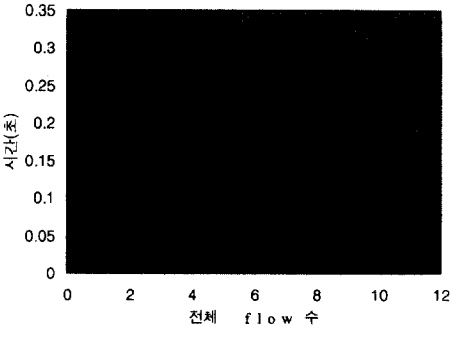
(그림 4)에서는 라우터 수를 20개로 하여 RTT 값을 측정하고 이 값을 식 (1)에 대입하여 계산한 트래픽 량과 시뮬레이션에서 실제로 발생한 트래픽 량을 비교하여 보여 주고 있다. 그림에서 standard로 표현된 것은 측정된 RTT 값을 식 (1)에 대입하여 얻은 결과이다.

(그림 4a)에서는 라우터 수가 증가함에 따른 각 TCP와 큐 타입의 트래픽 량을 보여 주고 있다. 실제로 발생하는 트래픽 량은 TCP와 큐 타입에 무관하게 식 (1)에서 계산한 결과보다 적은 것을 알 수 있다. (그림 4b)에서는 패킷의 크기를 512 바이트로 한 경우 패킷 드롭률(drop rate)의 변화에 따른 각 TCP와 큐 타입의 트래픽 량을 보여 주고 있다. (그림 4c)에서는 각 TCP와 큐 타입의 패킷 크기가 변화할 때 발생하는 트래픽 량을 보여 주고 있다.

이상의 시뮬레이션 결과에서 볼 수 있듯이 본 연구에서 제안된 RTT 측정 방식은 망 내 라우터에서 어떤 TCP 플로우가 표준의 혼잡 제어 알고리즘을 충실히 따르고 있는지를 감시할 때 사용할 수 있을 것으로 생각된다.

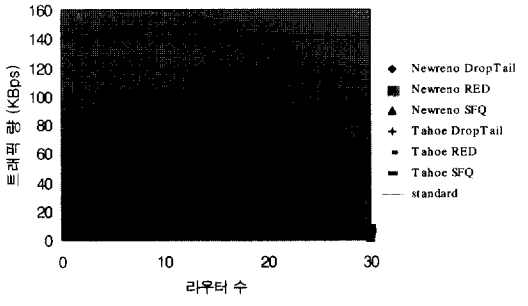


(a) 라우터 수의 증가에 따른 RTT 측정값
(Tahoe Full TCP, RED 큐)

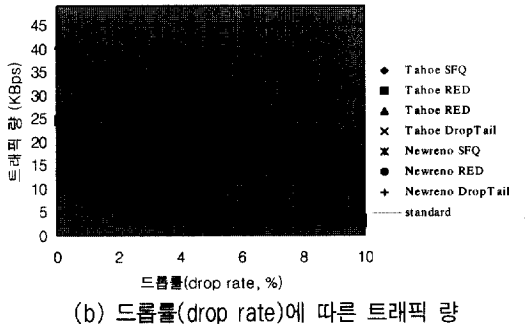


(b) 플로우 수의 증가에 따른 RTT 측정값
(Tahoe Full TCP, RED 큐)

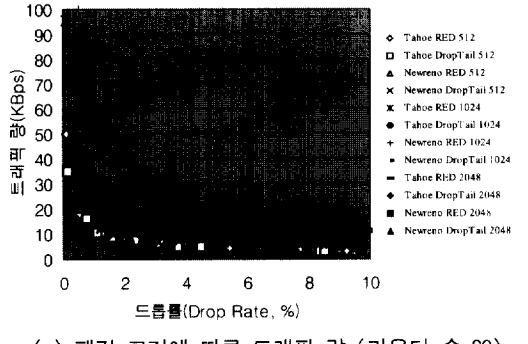
(그림 3) RTT 값 측정 결과



(a) 라우터 수에 따른 트래픽 량
(Drop rate=0.02, 패킷 크기=512)



(b) 드롭률(drop rate)에 따른 트래픽 량
(라우터 수 20, 패킷 크기 512 byte)



(c) 패킷 크기에 따른 트래픽 량 (라우터 수 20)

(그림 4) TCP와 큐 타입에 따른 트래픽 량

4. 결 론

본 연구에서는 망 내부의 라우터가 각 TCP 플로우의 RTT를 측정할 수 있는 알고리즘을 제시하고, 시뮬레이션을 통하여 제시된 알고리즘의 타당성을 검증하였다. 망 내 라우터는 본 연구에서 제시된 RTT 측정 알고리즘에 따라서 각 TCP 플로우의 RTT를 알아낼 수 있으며, 그 결과 TCP의 표준 혼잡 제어 알고리즘을 충실히 따르는 TCP 플로우와 그렇지 않은 TCP 플로우를 구분하여 망이 혼잡 상태에 빠지는 경우에 효율적으로 대처할 수 있을 것으로 생각된다.

현재 멀티미디어 데이터 전송을 필요로 하는 인터넷 응용의 사용이 점차적으로 활발해지고 있으며 이러한 응용들은 일반적으로 인터넷 프로토콜 계층 구조의 UDP 프로토콜을 사용한다. 따라서, 인터넷 망의 전체 트래픽 중에서 UDP 트래픽이 차지하는 비중이 점차적으로 높아질 것으로 예상되며, 이 경우 망 내 혼잡 제어에 관련된 문제가 더욱 커질 것으로 예상된다. 이는 TCP 프로토콜의 경우 망 내의 혼잡 상태에 따라서 망으로 전송하는 트래픽 량을 조절하는데 반하여 UDP에서는 그러한 기능을 제공하지 않기 때문이다. 이에 따라서, 앞으로는 TCP와 UDP 트래픽이 혼재된 망 내의 혼잡 제어에 관련된 연구를 수행하려고 한다.

참 고 문 헌

[1] V. Jacobson, "Congestion Avoidance and Control," SIGCOMM Symposium on Communications Architectures and Protocols, pp.314~329, 1988

[2] Sally Floyd, Kevin Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," IEEE/ACM Transaction on Networking, May 3, 1999

[3] K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and Sack TCP," ACM Computer Communication Review, Jul. 1996

[4] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Trans. on Networking, Aug. 1993

[5] Vern Paxon, "Measurements and Analysis of End-to-End Internet Dynamics," candidate thesis for Ph.D, Univ. of California Berkeley, April 1997

[6] B. Braden, et. al., "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, April 1998

[7] Kevin Fall, Kannan Varadhan, "ns Notes and Documentation," Manual for LBNLs Network

Simulator, July 1999

[8] M. Mathis, J. Semke, J. Mahdavi, "The Microscopic Behavior of the TCP Congestion Avoidance Algorithm," ACM Computer Communication Review, Vol.27, No.3, July, 1997

[9] M. Allman, et. al., "TCP Congestion Control," RFC 2581, April 1999

[10] J. Postal, "Transmission Control Protocols," STD . 7, RFC 793, Sept. 1981

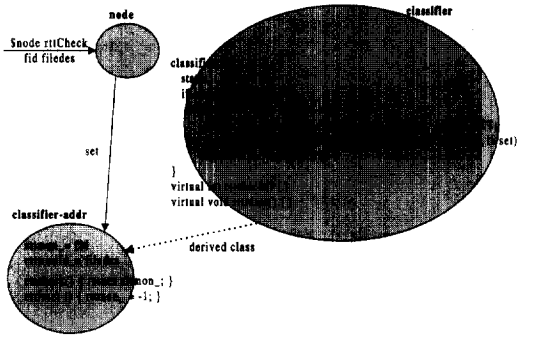
[11] W. Stevens, "TCP/IP Illustrated, Volume 1 : The Protocols," Addison-Wesley, 1994

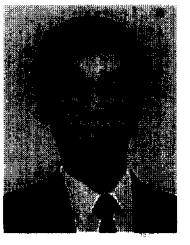
부 록

시뮬레이션을 위한 ns-2 코드의 수정

가. Interpreted class hierarchy의 node에서 RTT를 측정할 수 있도록 하는 proc rttCheck {fid filedesc} 를 추가(ns-2.1b5/tcl/lib/ns-default.tcl, ns-2.1b5/tcl/lib/ns-node.tcl 화일 수정)

나. 실제의 RTT 측정을 위하여 ns-2의 노드가 부록 (그림 1)과 같은 구조로 동작할 수 있도록 ns-2.1b5/classifier-addr.h, ns-2.1b5/classifier-addr.cc, ns-2.1b5/classifier.cc, ns-2.1b5/classifier.h 파일 수정





김 은 기

e-mail : egkim@tnut.ac.kr

1987년 고려대학교 전자공학과 (학사)

1989년 고려대학교 전자공학과 대학원(석사)

1994년 고려대학교 전자공학과 대학원(박사)

1995년~현재 대전산업대학교 정보통신컴퓨터공학부 조교수

관심분야 : 컴퓨터 네트워크, 이동통신