2002년도 대한전자공학회/대한전기학회 시스템 및 제어분야 합동 추계학술대회 논문집(2002.11.30)

TCP 단일 혼잡 망에서의 ECN을 이용한 공정한 윈도우 제어

변 희 정, 임 종 태
한국과학기술원 전자전산학과
전화 : 042-869-8041 / 핸드폰 : 017-416-0300

Fair TCP Window-based Control in Single Congested Network with Explicit Congestion Notifcation

Hee-Jung Byun and Jong-Tae Lim
Dept. of Electrical Engineering & Computer Science, KAIST
E-mail : crasiva@mail.kaist.ac.kr

Abstract

In this paper, we propose a window-based congestion control algorithm to achieve a fair sharing of the available bandwidth in ECN capable heterogeneous TCP networks with a single bottleneck link. The proposed algorithm is based on extracting the network status from the successive binary congestion information provided by ECN. From the explicit network information, we estimate the fair window size proportional to the propagation delay. Through simulations, the effect and performance of the proposed algorithm are shown for the heterogeneous networks.

I. 서론

TCP(transmission control protocol) 알고리즘은 귀환(feedback)을 토대로한 윈도우 제어 알고리즘이며 각각의 TCP 호스트(host)는 timeout이나 중복 ACK(acknowledgement)를 받음에 따라 독립적으로 윈도우 크기를 결정한다. 이러한 윈도우 제어 알고리즘은 end-to-end 방식으로 이루어지고 망에 대한 유효한 자원이나 상태에 대해 알 수 없으므로 각 호(connection)간의 공정한 윈도우 크기 할당이 힘들다. 특히 TCP의 윈도우 제어 알고리즘은 AIMD(additive increase /multiplicative decrease)방식의 한 종류이고 이것은 TCP 불공정성(unfairness)의 주요한 요인이 된다 [1]. 이를 해결하기 위해 RED(random early detection) [2]와 ECN(explicit congestion notification) 등을 이용한 여러 윈도우 제어 기법들이 제안되었다 [3]. ECN은 망 내에서 혼잡이 일어났을 경우 패킷을 버리는 대신 귀환을 통해 호스트에게 혼잡이 일어났음을 알려주는 방법이다 [4]. 하지만 이는 혼잡 발생의 유무만을 알려주므로 호스트는 정확한 망의 상태를 알 수 없고 이로 인해 망에서 허용할 수 있는 공정한 윈도우 크기를 예측하는데 큰 어려움이 존재한다. 따라서 기존의 TCP 프로토콜을 유지하면서 망의 상태를 정확하게 파악하고 각 호간의 공정한 윈도우 크기를 예측하는 새로운 기법이 필요하게 된다.

본 논문은 TCP 단일 혼잡 망에서의 ECN을 이용한 공정한 윈도우 제어 알고리즘을 제안하고자, 먼저 연속적인 ECN 정보를 통해 망의 상태를 알아내는 기법을 제시하고 이를 토대로 각 호의 윈도우 크기를 조절하는 알고리즘을 제안한다. 또한 제안한 알고리즘을 검증함과 동시에 시뮬레이션을 통하여 TCP 단일 혼잡 망에 대한 그의 효과와 결과를 보이고자 한다.

II. 윈도우 제어 알고리즘

2.1 망 모델링

먼저 각각의 호는 서로 다른 전파지연(propagation delay)을 가지고 하나의 링크(link)를 공유하는 망을 고려해 보자. 위의 망을 정의 하기 위해 다음과 같은 사항을 가정한다.

- 호스트는 자신의 RTT(round-trip time)마다 윈도우 크기를 갱신한다.
- 망의 혼잡이 일어난 RED 게이트웨이(gateway)의 링크용량은 C [packets/s]이다.
- 소스 트래픽(source traffic)은 이산 유체흐름(deterministic fluid flow)으로 표현된다.

전체 호의 수는 M이고 τ_m은 호 m의 최소 RTT라고 하자. 여기서 최소 RTT는 망의 혼잡이 없을 때의 얻어진 RTT이다. 그리고 다음 식을 만족하며 τ_m에 비례하는 양수 $\Delta_m (1 \leq m \leq M)$이 존재한다고 하자.

$$\frac{\tau_1}{\Delta_1} = \frac{\tau_2}{\Delta_2} = \cdots \frac{\tau_{M-1}}{\Delta_{M-1}} = \frac{\tau_M}{\Delta_M} = \tau_s$$

$$(\tau_1 < \tau_2 < \cdots < \tau_{M-1} < \tau_M)$$

여기서 τ_s는 상수이다. 만약 RED 게이트웨이에서의 패킷 지연이 무시할만큼 작다고 하면 시스템은 τ_s가 타임슬롯의 길이인 이산시간 모델로 표현될 수 있다 [5]. 즉 $w_m(k)$를 호 m의 윈도우 크기이고 $q(k)$는 RED 게이트웨이의 버퍼길이라고 할 때 $(k+1)$ 슬롯에서의 q는 다음과 같이 나타낼 수 있다.

$$q(k+1) = [q(k) + \sum_{m=1}^{M} U(\Delta_m, k) w_m(k) - C\tau_s]^+ \quad (1)$$

여기서 $U(\Delta_m, k) = \begin{cases} 1 & \text{if } mod(k, \Delta_m) = 0 \\ 0 & else \end{cases}$

그리고 $[\cdot]^+ = max(0, \cdot)$이다.

RED 알고리즘은 패킷이 도착할 때마다 저역통과 필터를 이용하여 평균 버퍼길이를 계산하게 된다. 만약 q가 급격히 변하지 않는다면 평균 버퍼길이 $\bar{q}(k)$는 다음과 같은 식으로 표현된다.

$$\bar{q}(k+1) = (1-w_q)^{\sum_{m=1}^{M} U(\Delta_m, k) w_m(k)} \bar{q}(k) + (1-(1-w_q)^{\sum_{m=1}^{M} U(\Delta_m, k) w_m(k)}) q(k) \quad (2)$$

여기서 w_q는 가중율(weighting factor)이다.

2.2 RED 알고리즘과 윈도우 제어

이제 RED 게이트웨이에 대한 변형된 알고리즘과 윈도우 제어 알고리즘을 설명하겠다. 먼저 망의 명시적인(explicit) 정보를 추출해 내기 위한 기법을 설명하고자 한다. 제안하는 알고리즘에서 RED 게이트웨이는 오직 평균 버퍼길이과 버퍼 용량만을 이용하여 패킷 마킹 확률(packet marking probability) $p_b(k)$ [6]를 결정한다. 즉 변형된 RED 알고리즘에서의 $p_b(k)$는 다음과 같다.

$$p_b(k) = 1 - \frac{1}{B^2}(\bar{q}(k) - B)^2 \quad (3)$$

여기서 B는 버퍼용량이다.

기존의 ECN이 제공하는 1비트 혼잡유무 정보는 윈도우를 적절히 제어하는데 충분하지 않으므로 우리는 RTT동안에 들어온 ECN 비트 개수를 통해 RED 게이트웨이에서의 평균 버퍼길이 정보를 추출한다. 즉 각 호스트는 RTT 동안 들어온 N개의 ACK 중 ECN 비트가 표시된 N_e개의 ACK 수를 다음과 같이 산출한다.

$$e(k) = \frac{N_e}{N} \quad (4)$$

RED 알고리즘은 들어오는 패킷들을 각각의 호의 윈도우 크기에 비례하여 표시하게 되므로 [2] N_e와 N의 비율은 RED가 랜덤하게 패킷을 표시하는 확률과 동일하게 된다 [7]. 즉

$$e(k) = p_b(k) \quad (5)$$

(3)-(5)로부터 호스트는 구체적인 망정보, 즉 RED 게이트웨이 평균 버퍼길이를 다음과 같이 추출해 낼 수 있다.

$$\bar{q}(k) = (1 - \sqrt{(1-e(k))})B \quad (6)$$

이렇게 추출된 평균 버퍼길이를 이용하여 각 호스트는 자신의 윈도우 크기를 다음과 같이 조절한다.

$$w_m(\Delta_m(k+1)) = [\alpha(max_{th} - \bar{q}(\Delta_m k))\Delta_m]^+ \quad (7)$$

$$(1 \leq m \leq M)$$

여기서 α는 제어 이득이며 $\max_{th}$는 버퍼길이의 최대 기준값이다. 즉 각 호스트의 윈도우 크기는 그의 전파지연에 비례하여 증가하게 된다. 따라서, 느린 호의 자신의 RTT마다 내보내는 윈도우 크기는 상대적으로 RTT가 작은 호에 비해 크게 되므로 TCP의 불공정한 현상은 개선될 수 있다.

2.3 평형점과 점근적 안정

2.1의 망 모델로부터, 윈도우 제어 방식은 각 호스트의 RTT마다 w_m윈도우를 전송하는 것이다. 이러한 망 모델을 평균적인 입장에서 접근하여 근사화하겠다. 즉

호 m은 하나의 타임슬롯마다 평균적으로 w_m/Δ_m의 윈도우를 전송한다고 가정한다. 그렇다면 (7)은 다음과 같은 식으로 다시 쓸 수 있다.

$$w_m(k+1)=[\alpha(max_{th}-q(k))]^+ \quad (8)$$

그 결과 변형된 q와 $\bar{q}$는 다음과 같이 주어진다.

$$q(k+1)=[q(k)+\sum_{m=1}^{M} w_m(k)-C\tau_s]^+ \quad (9)$$

$$\bar{q}(k+1)=(1-w_q)^{\sum_{m=1}^{M} w_m(k)}\bar{q}(k) + (1-(1-w_q)^{\sum_{m=1}^{M} w_m(k)})q(k) \quad (10)$$

q_s, $\bar{q}_s$와 w_{ms}을 각각 $q(k)$, $\bar{q}(k)$와 $w_m(k)$의 평형점이라 하자. 평형점 부근에서의 포화 비선형특성은 무시한다. 만약 $\max_{th}$를 $max_{th} > C\frac{\tau_s}{\alpha M}$가 되도록 선택한다면 (8)-(10)식의 평형점은 다음과 같이 얻어진다.

$$w_{ms}=C\frac{\tau_s}{M},\quad q_s=\bar{q}_s=max_{th}-C\frac{\tau_s}{\alpha M} \quad (11)$$

즉 $\max_{th}$를 $max_{th} > C\tau_s/\alpha M$이 되도록 선택한다면 시스템은 (11)의 평형점을 갖게 됨과 동시에 링크는 완전히 사용되고 각 호의 공정한 대역 분배가 이루어지게 된다. 게다가 버퍼의 길이는 $\max_{th}$보다 작은 값으로 유지된다.

이제 (11)의 평형점의 점근적 안정을 살펴보기로 한다. $\boldsymbol{x}(k)$를 다음과 같은 상태벡터라고 하자.

$$\boldsymbol{x}(k)=[w(k)-w_s \quad q(k)-q_s \quad \bar{q}(k)-\bar{q}_s]^T$$

(8)-(10)으로 표현되는 시스템을 평형점 주위에서 선형화한 결과, 시스템은 다음과 같이 선형시스템으로 나타난다.

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)$$

여기서

$$\boldsymbol{A}=\begin{bmatrix} 0 & 0 & -\alpha \\ M & 1 & 0 \\ 0 & 1-(1-w_q)^{C\tau_s} & (1-w_q)^{C\tau_s} \end{bmatrix}$$

Jury's criterion으로부터, 제어 이득 α가 다음과 같이 선택되어진다면 시스템은 점근적 안정하다:

$$0<\alpha<\frac{1-(1-w_q)^{C\tau_s}}{M} \quad (12)$$

III. 모의실험

제안한 TCP 윈도우 제어 알고리즘의 모의실험 결과를 보이고자 한다. RED 게이트웨이의 처리속도 C는 1500[패킷/s]이며 TCP 호의 수는 5이다.

호	TCP1	TCP2	TCP3	TCP4	TCP5
지연(ms)	50	100	150	200	250

표 1 TCP 호의 최소 RTT

$\max_{th}$와 $\min_{th}$는 각각 500, 20 패킷이다. 각 TCP 호의 최소 RTT, τ_m은 표 1에 나타내었다. 타임슬롯의 길이 τ_s는 50ms이고 따라서 $\Delta_m(1 \leq m \leq 5)$는 1, 2, 3, 4, 그리고 5가 된다. w_q는 0.01이며 (12)에 따라 제어이득 α는 0.07로 선택했다. 제안한 알고리즘과 [3]의 알고리즘에 대해 윈도우 크기, RED 게이트웨이에서의 버퍼길이 그리고 ACK수의 관점에서 성능을 비교할 것이다.

그림 1은 [3]의 알고리즘에 대한 윈도우 크기, RED 게이트웨이에서의 버퍼길이, 그리고 ACK수를 보여주며 그림 2는 제안한 알고리즘의 (8)-(10)을 바탕으로 한 근사화된 모델의 결과를 보인다. 그림 1에서 보다시피 버퍼길이는 $\max_{th}$와 $\min_{th}$ 사이에 존재한다. 하지만 누적된 ACK수를 보게 되면 각 호스트들의 공정한 윈도우 할당이 이루어지지 않음을 알 수 있다. 또한 윈도우 크기와 버퍼의 길이의 진동이 주기적으로 일어남을 보여준다.

이에 반해 그림 2는 모든 호스트이 그들의 서로 다른 전파지연에 상관없이 같은 윈도우 크기로 전송함을 보여준다. 또한 윈도우와 버퍼길이에 있어서 진동현상은 배제됨과 동시에 버퍼길이 역시 $\max_{th}$보다 작은 어느 한 값에 유지됨을 알 수 있다.

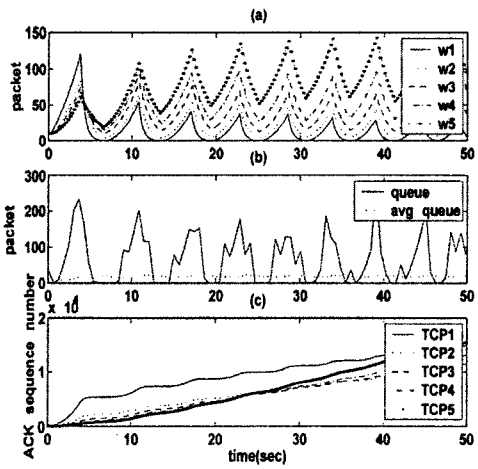

그림 1 알고리즘 [3] : (a) 윈도우 크기, (b) 버퍼길이와 평균버퍼길이, (c) ACK sequence number

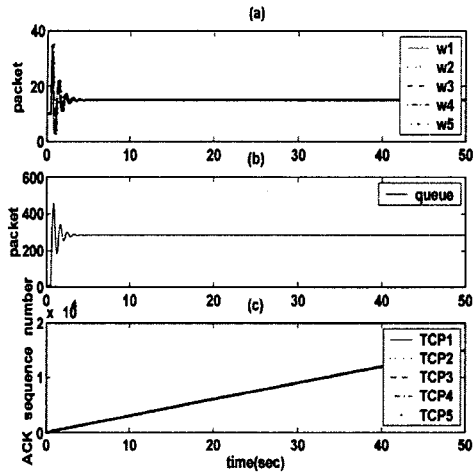

그림 2 (8)-(10)에 기인한 제안한 알고리즘 : (a) 윈도우 크기, (b) 버퍼길이, (c) ACK sequence number

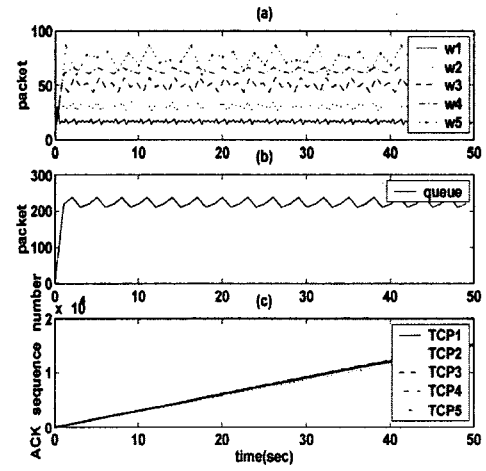

그림 3 (1)-(2),(7)에 기인한 제안된 알고리즘 : (a) 윈도우 크기, (b) 버퍼길이, (c) ACK sequence number

그림 3은 제안된 알고리즘의 (1)-(2),(7)에 기초로한 시뮬레이션 결과를 보여준다. ACK 결과를 보게 되면 그림 2의 근사화된 ACK 결과와 거의 비슷한 성능을 보인다. 하지만 윈도우와 버퍼길이에 있어서 약간의 진동이 있는데 이는 (8)-(10)에 의해 정의된 근사화된 망의 가정으로부터 기인한 것이라 볼 수 있다.

즉 호 m은 하나의 타임슬롯마다 평균적으로 w_m/Δ_m의 윈도우를 전송한다고 근사화하였기 때문에 약간의 차이가 존재한다.

위의 결과들을 종합해 볼때 제안한 알고리즘은 공정한 대역분배가 이루어짐과 동시에 링크효율도 개선시킬 수 있음을 보여준다.

V. 결론

본 논문은 단일 혼잡노드를 가진 TCP 이종망에서 ECN을 이용한 공정한 대역분할을 위한 윈도우 혼잡제어 알고리즘을 제안하였다. 또한 연속적인 ECN 혼잡표시를 이용하여 구체적인 망정보, 즉 RED 게이트웨이에서의 평균 버퍼길이를 얻었다. 이러한 망정보와 제안한 윈도우 제어 알고리즘을 이용하여 호스트는 자신의 공정한 윈도우 크기를 예측할 수 있었으며 링크효율 역시 개선되었음을 모의 실험을 통하여 보였다.

참고문헌(또는 Reference)

[1] G.Hasegawa, and M.Murata, "Survey on fairness issues in TCP congestion control mechanism," IEICE Trans. on Communications, vol. E84-B, no. 6, pp. 1461-1472, 2001.

[2] S.Floyd and V.Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Trans. on Networking, vol. 1, no. 4, pp. 397-413, 1993.

[3] T.Hamann and J.Walrand, "A new fair window algorithm for ECN capable TCP (new-ECN)," IEEE INFOCOM, pp. 1528-1536, Mar. 2000

[4] S.Floyd, "TCP and explicit congestion notification", ACM Computer Communication Review, vol. 24, no. 5, pp. 8-23, 1994.

[5] K.Takagaki, H.Ohsaki, and M.Murata, "Analysis of a window-based flow control mechanism based on TCP vegas in heterogeneous network environment", ICC2001, pp. 3224-3228.

[6] H.Ohsaki and M.Murata, "Steady state analysis of the RED gateway: stability, transient behavior, and parameter setting", IEICE Trans. on Communications, vol E85-B, no. 1, pp. 107-115, 2002

[7] D.Lapsley, and S.Low, "Random early marking for internet congestion control", IEEE Globecom, pp. 1747-1752. Dec. 1999.