Link-layer Trigger를 통한 유무선 통합 환경에서의 TCP 혼잡제어

김유미*, 홍충선**
경희대학교 컴퓨터 공학과

e-mail:yumi@networking.khu.ac.kr
cshong@khu.ac.kr

TCP Congestion Control Algorithm Using L2 Trigger in Heterogeneous Networks

Yu-Mi Kim*, Choong Seon Hong**
Dept of Computer Engineering, Kyung Hee University

요 약

유무선 통합 환경에서의 기존의 TCP가 가지는 문제점은 혼잡 제어 알고리즘이 무선환경의 에러를 모두 congestion 으로 인식하는 것에 있다. 무선환경에 의한 에러일 경우 네트워크의 상황이 좋음에도 불구하고 전송률을 낮춰서 네트워크의 성능을 떨어뜨리게 된다. 따라서 본 논문에서는 Link-layer가 발생시킨 이벤트를 통해 에러의 원인을 명확이 구분하여 기존의 TCP의 큰 수정이 없이도 성능을 크게 향상 시킬 수 있는 혼잡 제어 알고리즘을 제안한다.

1. 서론

무선 통신 기술의 발달로 인터넷 환경이 점차 유무선 통합 환경으로 바뀌어가고 있다. 인터넷의 기반을 이루고 있는 TCP는 유선환경 기반 하에 만들어 진 것이므로 인터넷 환경에 따른 기존의 TCP의 문제점이 나타나고 있다. 기존의 TCP에서는 패킷 손실이 주로 congestion에 의한 경우여서 혼잡 제어 방법에 의한 전략으로 이를 해결했다.[1]

유무선 통합 환경에서의 에러는 link fail 에 의한 패킷 손실이나, 잦은 핸드오프에 의한 랜덤 로스 발생 등의 무선환경에서의 에러 가 주원인이므로 이러한 상황을 고려한 제어가 필요하다. 그러나 기존의 혼잡제어 전략은 무조건 에러를 congestion으로 간주하여 congestion window(cwnd)의 크기를 줄이는 방식이다. 송신측은 윈도우 사이즈에 의해 전송할 데이터의 양을 결정 하는데, 기존의 제어 방법은 무선환경의 에러인 경우에도 congestion으로 잘못인식하여 네트워크의 상황이 좋음에도 불구하고 적은 데이터만을 보내게 되므로 당연히 네트워크의 성능이 떨어진다.

따라서 이를 명확히 구분한다면 성능이 크게 향상 될 수 있다. 이에 대한 연구는 Wireless TCP 구간과 Wired TCP 구간의 명확한 구분을 해주거나, 수신측이나 송신측 어느 한 곳에서 네트워크의 상황을 고려하려 혼잡 윈도우의 크기를 조절하는 방식으로 연구가 진행되고 있다. 그러나 이러한 방법 또한 에러의 원인을 정확히 파악하지 못한 방법이므로 네트워크의 성능을 크게 향상 시키지는 못한다.

최근 Mobile IP 분야에서 이슈가 되고있는 Link layer Trigger는 링크 레벨에서 trigger가 event를 발생시켜서 IP 계층에 핸드오프 과정 중 Link layer의 상황을 알려주는 역할을 하는 것이다. 링크의 상태를 파악하면 에러의 상황을 예측할 수 있다. 따라서 본 논문에서의 Link-layer Trigger를 이용한 유무선 통합 환경에서의 에러의 명확한 구분을 통해 네트워크의 성

This work was supported by University ITRC project of MIC.

능을 향상시키고자 한다.

2. 관련 연구

2.1 Transmission Control Protocol

TCP에서의 데이터 전송방법에는 Flow Control 기법과 Congestion Control 기법이 있다.[2][3]

◆Flow Control algorithm

sliding window에 의해 수신측에서 감당할 수 있는 만큼의 크기를 결정하여 데이터를 전송하고 ACK를 받은 만큼 크기를 늘여서 다음 세그먼트를 전송한다.

◆congestion control

수신측의 네트워크 상황에 따른 Advertised receive window(rwin)와 congestion에 의한 congestion window(cwnd) 의 크기에 의해서 데이터를 전송한다. TCP에서는 아래 식과 같이 이 두 윈도우 크기의 최소값에 의해 데이터를 전송하게 된다.

$$Send_{Max} = \min\{cwnd, rwin\}$$

2.2 Congestion Control Algorithm

congestion이 발생하면 TCP는 다음과 같은 전략으로 에러를 복구한다.[4][5]

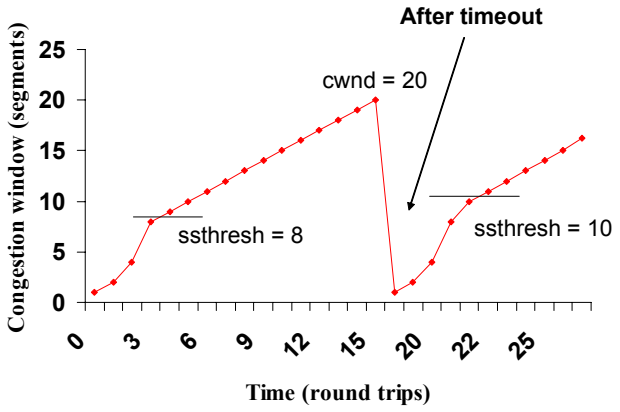

그림 1. congestion control

◆Slow Start

cwnd의 초기값은 세그먼트 하나의 크기로 초기화하고, ssthresh(slow start threshould) 의 값은 66535 bytes 로 정의한다. ACK를 수신할 때 마다 ssthresh에 도달하기 전까지 cwnd를 지수적으로 증가시킨다.

◆Congestion Avoidence

cwnd는 수신자측에서 알려주는 rwin과 달리 송신자측에서 congestion정도에 따라 크기를 조절해야한다. 따라서 에러를 congestion으로 보고 그 크기를 줄이게 된다. 이때 AIMD(Additive Increase Multiple Decrease)방식으로 congestion에 대응한다. 타임아웃이 없이 데이터가 보내지면 cwnd를 각 RTT마다 하나의 패킷만큼 늘이고, 타임아웃이 발생하면 congestion상황으로 인식하고 cwnd를 반으로 줄인다.

◆Fast Retransmission

타임아웃에 의해 패킷 손실을 인식하는데는 시간이 많이 걸리므로 세 개 이상의 연속된 Duplicate ACK를 받으면 바로 패킷 손실로 보고 재전송을 한다.

2.3 Wireless TCP

wireless TCP의 연구가 진행되면서 기존의 프로토콜을 수정한 TCP 버전이 소개되었다. 그 중 가장 대표적인 세 가지를 소개하자면 다음과 같다.[6]

◆Tahoe

기본적으로 slow start 와 congestion avoidence ,fast retransmission을 수행한다.

◆Reno

가장 널리 알려진 프로토콜로 기본적인 slow start,congestion avoidence, fast retransmission 과 함께 fast recovery를 수행한다. fast recovery는 패킷 손실 후 slow start를 수행하지 않고 잃어버린 세그먼트를 빠르게 재전송 하는 것이다.

◆Vegas

system clock을 따로 구동시켜 네트워크의 상황을 측정하여 데이터를 전송하는 방식이다.

$$Expected = WindowSize / BaseRTT$$

위의 식에 의한 예측값과 실제 네트워크의 상황을 비교해서 네트워크의 상황이 좋으면 데이터를 늘이고 그렇지 않은 경우는 보내는 데이터의 크기를 줄인다. 즉 다음과 같은 알고리즘을 수행한다.[7]

$$Diff = Expected - Actual$$

$$Diff < \alpha \rightarrow \text{increase window linearly}$$

$$Diff > \beta \rightarrow \text{decrease window linearly}$$

2.4 Link-layer Trigger

Link-layer Trigger는 L2에서 발생한 특정한 이벤트를 알려주는 역할을 한다. 현재 정의된 L2 trigger 이벤트는 다음과 같다.

◆Source trigger

◆Target trigger

◆Mobile trigger

◆Link up trigger

-링크가 연결되었음을 알리는 이벤트

◆Link down trigger

-링크의 연결이 끊어졌음을 알리는 이벤트

3. 제안 사항

기존의 TCP 혼잡제어 방식의 가장 큰 문제점은 유선 환경에 적합한 알고리즘의 사용으로 혼잡제어 알고리즘의 잘못된 에러 분석에 의한 성능저하이다. 유무선 혼합환경에서 TCP의 성능을 높이기 위해서는 명확한 에러의 구분이 필요하다.따라서 본 논문에서는 이에 적합한 혼잡 제어 전략을 제시하고자 한다. TCP에서는 congestion이 생기거나 잦은 핸드오프로 인한 bursty한 에러,무선 링크가 깨졌을 때 주로 에러가 발생한다. 비록 에러의 원인을 정확하게 구분 할 수는 없지만 무선 링크의 특정한 에러 상황을 즉시 전달 받을 수는 있다. 이러한 이벤트를 알려주는 것이 L2 trigger이다.[8]

L2 trigger에 의해 발생한 이벤트는 TCP 계층에 전달하도록 한다.(그림2) TCP계층에서는 전송 중 발생한 trigger 이벤트를 살펴보고 링크의 상태와 congestion이 아닌 에러를 찾아 낼 수 있다.[9]

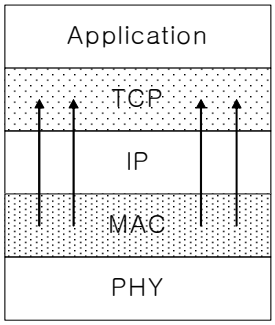

그림 2. trigger 이벤트 전달 구조

Link down Trigger메세지가 오면 link fail을 의미한다. 그러나 단순히 Link down trigger가 에러 발생을 의미하는 것은 아니다. Mobile IP에서의 핸드오프 과정에서 link down 메시지는 아주 짧은 시간 정상적으로 일어날 수 있다.

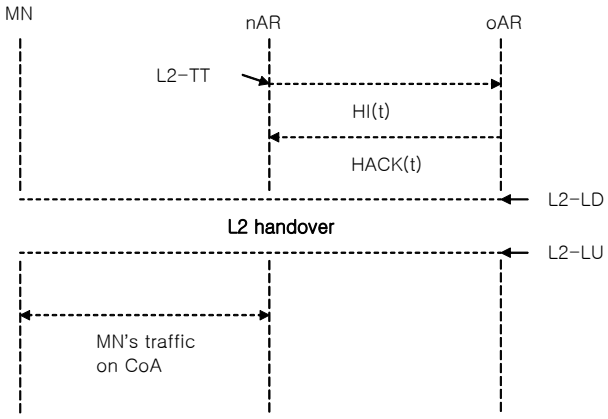

그림 3. Target Trigger Handover Timing

그림 3에서 보듯이 Mobile Node가 old Access Router에서 new Access Router쪽으로 이동 할 때 핸드오프 과정에서 순간적으로 Link down 과 Link up이 일어남을 알 수 있다.

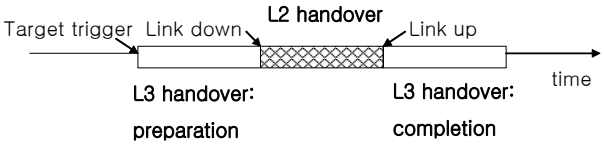

그림 4. L2 handover 에서의 trigger 이벤트

그림 3과 그림 4는 L2 핸드오프 과정을 순차적으로 도식했다.[10]Target Trigger가 발생한 후에는 Link down과 Link up 이벤트가 발생함을 알 수 있다. 이러한 과정이 없이 Link down 이벤트가 발생한 경우는 단순한 Link fail로 볼 수 있다. 이때는 congestion 알고리즘을 수행하지 않고 바로 재전송을 해야한다. 본 논문에서는 Link up 이 되는 순간 바로 재전송을 수행하는 기법을 Instant Retransmission이라 정의하였다. 그림 5에서는 본 논문에서 제시한 알고리즘을 자세히 보여준다.

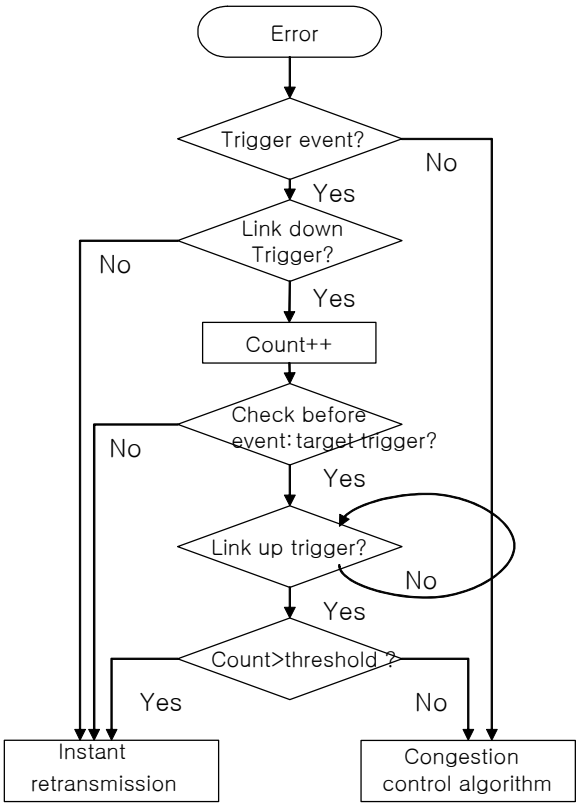

그림 5.Algorithm Flow Chart

그림 5의 알고리즘은 에러 발생 시 혼잡제어 알고

리즘을 제시하였다. 먼저 에러가 발생했을 때 trigger이벤트가 발생했는지를 살펴본다. 만약 trigger 이벤트가 없다면 단순한 혼잡제어로 가정하고 기존의 혼잡제어 알고리즘을 그대로 따른다. trigger 이벤트 발생시 먼저 link down 이벤트가 발생했는지 여부를 살펴본다. 만약 link down 이벤트가 발생되었음에도 불구하고 그 전에 다른 trigger가 없으면 이는 단순한 Link fail이다. 이 때는 즉시 재전송을 한다.

너무 잦은 핸드오프는 링크의 연결이 간헐적으로 일어나므로 bursty한 에러를 만들게 되고, 이 경우 패킷 손실이 일어난다. link down 이벤트를 체크하는 함수에 카운터를 두고 이벤트가 발생할 때 마다 카운터를 하나씩 증가시킨다. 따라서 link down 이벤트의 횟수가 일정 한계치를 넘어서면 이 역시 무선 구간에서의 잦은 핸드오프에 의한 에러로 가정하고 즉시 재전송한다.

비록 정상적인 핸드오프 과정에서 Link down 이벤트가 발생했다 하더라도 Link up 이 되지 않으면 이 역시 link fail이다. 순서도에서 보면 link up이 될 때 까지 루프를 돌게 되는데, 그림 5의 알고리즘은 기본적으로 RTT값에 의한 타임아웃을 기본으로 하기 때문에 계속해서 Link up 으로 가지 못한 경우에도 일정 시간 후 재전송이 된다. count 역시 일정 시간후 다시 초기화가 된다고 가정한다.

여기서 제안한 알고리즘은 기존의 TCP에 거의 수정을 가하지 않고 이벤트 발생만을 검사하는 형태이므로 기존의 혼잡제어 방식을 크게 벗어나지 않으면서 성능을 높일 수 있는 최적의 알고리즘이다. 비록 현재 Link layer Trigger의 표준화가 진행중이지만 본 논문에서 사용한 TCP 계층에서의 이벤트는 정의되어 있기 때문에 이를 이용한 알고리즘으로 가장 적합한 TCP 알고리즘을 구현할 수 있다.

4. 결론 및 향후 과제

본 논문에서는 기존의 TCP 혼잡제어의 문제점인 congestion 에러와 무선 에러의 명확한 구분으로 인한 혼잡 제어 전략을 제시하였다. Link layer Trigger 이벤트를 통해 Link fail을 바로 감지하여 즉시 재전송 함으로써 네트워크의 성능을 높일 뿐 아니라 congestion 에러로 잘못 인식되어 congestion window 의 크기를 줄여 성능을 낮추는 기존의 무선환경에 적합하지 않은 알고리즘 또한 회피하였다.

더구나 잦은 핸드오프에 의한 에러 역시 감지하여 이에 의한 에러도 즉시 재전송으로 극복할 수 있다.

TCP는 오랫동안 많은 사람들에 의해 쓰여진 인터넷의 가장 기본적인 프로토콜이다. 그러나 무선 인터넷의 발전에 따른 문제점이 있었다. 여기에 최소의 수정으로 기존의 TCP를 그대로 사용할 수 있다면 최소의 비용으로 최고의 성능을 보장할 수 있을 것이라고 본다.

향후 과제로는 본 논문에서 제안한 알고리즘을 바탕으로 실질적인 알고리즘 구현과 이를 통한 시뮬레이션으로 성능을 측정하여 네트워크의 성능개선을 평가하는 것이다.

알고리즘에서 단순히 잦은 핸드오프에 의한 에러의 한계값을 임의의 값 threshould로 정의하였는데 이에 대한 실험값으로 정확한 값을 얻어내야 하는 것 역시 남겨진 과제이다.

참고문헌

[1]C. Zhang and V. Tsaoussidis,"The Interrelation of TCP Responsiveness and Smoothness in Heterogeneous Networks",Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)

[2]Nitin H. Vaidya. Texas A&M University."TCP for Mobile and Wireless Hosts",1999

[3]Forouzan,"TCP/IP Protocol Suite", McGRAW-HILL

[4]W.Richard Stevens,"TCP/IP Illustrated, Volume1 - The Protocols", Addison-Wesley

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

[6]C. Zhang and V. Tsaoussidis,"TCP-Real: Improving Real-Time Capabilities of TCP over Heterogeneous Networks",The 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video IEEE/ACM NOSSDAV 2001, Port Jefferson, NY, June 2001

[7]Lawrence S. "Brakmo,TCP Vegas: New Techniques for Congestion Detection and Avoidance",1994 ACM SIGCOMM Conference

[8]A. Yegin,"Link-layer Triggers Protocol", draft-yegin-l2-triggers-01.txt,24-Jun-02

[9]James Kempf,"Supporting Optimized Handover for IP Mobility - Requirements for Underlying Systems",draft-manyfolks-l2-mobilereq-01.txt,November 2001

[10]G.Dommety,"Fast Handovers for Mobile IPv6",draft-ietf-mobileip-fast-mipv6-03.txt,Jun 2003