Ad-Hoc 네트워크 환경에서 무선 TCP 성능 향상 기법

이동재° 송주석
연세대학교 컴퓨터산업시스템공학과
{dongjae°, jssong}@emerald.yonsei.ac.kr

Performance Enhancement Scheme of Wireless TCP in Ad-Hoc Networks

DongJae Lee° JooSeok Song
Dept. of Computer Science and Industrial System Engineering,
Yonsei University

요 약

Ad-hoc 네트워크 환경은 자유롭고 예측할 수 없는 통신 노드들의 이동으로 인해 기본적인 TCP 혼잡 제어가 상당히 나쁜 영향을 미치게 된다. 기본적인 기존의 TCP에 대한 에러 감지(detection) 및 복구(recovery)는 새로운 환경에 맞추어 개선되어야 한다. 이 논문에서는 잦은 경로변경 현상을 네트워크의 피드백 없이도 패킷의 순서가 바뀌어 전달됨을 이용하여 알아내고, 이를 처리하여 성능을 향상시킬 수 있는 기법을 제안한다. 또한 오랜 시간의 네트워크 단절에 따른 성능감소문제를 수신자(Receiver)가 검사하여 해결할 수 있는 기법을 제안한다.

1. 서 론

오늘날 무선 환경에서의 인터넷 사용이 크게 증가하였다. 또한 현재, 활발하고 다양한 이동성과 멀티 홉간의 무선 통신을 지원하는 이동 Ad-Hoc 네트워크 환경에 대한 연구가 활발히 진행되고 있다.[1][2] Ad-Hoc 네트워크는 고정된 기반 망의 의존함 없이 무선 이동 노드들에 의해서 자율적으로 구성되는 망을 의미한다. 이 망은 네트워크의 이동성과 자율성을 부여하며 독립적으로 운영 가능하다. 이러한 Ad-hoc 네트워크는 현재 구조 활동이나 전쟁지역 통신 등에 제한적으로 사용되고 있으나, 기술의 진보에 따라 인터넷 환경에서 중요한 부분을 차지할 것으로 예측된다. 또한 간단하고 빠르면서도 다양한 연결과 수행이 가능함으로써, 미래의 무선 IP 통신 애플리케이션의 새로운 대안을 제시한다 하겠다. Ad-Hoc 네트워크에서의 TCP(Transmission Control Protocol)는 대부분의 IP 애플리케이션이 수행된다는 점에 있어서 상당히 중요한 의미를 가지고 있다. 기존의 TCP는 유선환경을 대상으로 개발되고 사용된 것으로, 지금의 무선 환경에서 사용하는 데는 한계를 가지고 있다. 기존의 TCP는 에러의 특성에 대한 검출이 없고, 패킷 손실의 원인을 네트워크 혼잡(congestion)으로만 생각하고 윈도우 크기(window size)를 줄임으로써, 에러 회복(recovery)에 대해 효과적으로 작동하지 못한다.[3] 더욱이 Ad-Hoc 네트워크 환경에서 발생하는 자유롭고 예측할 수 없는 움직임에 대해 더욱 심각한 성능 저하가 일어날 수 있다. 이 논문에서는 기존에 연구되어진 Ad-Hoc 네트워크에서의 TCP 성능향상 기법에 대해서 살펴보고, 잦은 경로 변경과 경로 실패에 따른 네트워크 분리(partition)현상을 고려한 새로운 TCP 성능 향상 기법을 제안하고자 한다.

2. Ad-hoc 네트워크 환경에서의 TCP 연구

2.1. TCP와 Ad-hoc Network

이 논문에서는 Ad-hoc 네트워크 환경에서의 무선 TCP 성능향상 기법을 제시한다. 이를 위해서는 기존의 환경과 Ad-Hoc 환경에서의 차이를 인식하여야 한다. 기존의 TCP가 무선환경에서 문제점을 가지고 있다고 앞에서 서술한 바 있다. 더욱이, Ad-Hoc 네트워크 환경에서의 성능 저하는 더욱 심각하다.

2.1.1. 잦은 경로 변경 문제

먼저 Ad-Hoc라우팅의 경우 자주 경로가 바뀌고 경로 실패도 자주 일어난다. 특히 다중 경로 라우팅(Multiple path routing)의 경우 상당량의 순서가 바뀐 패킷이 발생한다. 예를 들면, 순서가 빠른 패킷을 긴 경로를 통해 전송하고, 잠시 루트에 문제가 생겨서 다음 순서가 늦은 패킷을 짧은 경로를 통해 전송한다면 수신자(Receiver) 측에서는 순서가 바뀐 패킷을 받게 된다. 즉 이러한 순서가 바뀐 패킷전달의 주원인은 경로 변경이다. 경로 변경에 의해 순서가 변경된 패킷 전달은 유선 네트워크에서 자주 일어나지 않으므로 무시 될 수 있었다 그러나 높은 수준의 이동이 일어나는 Ad-Hoc 네트환경에서 이러한 경로 변경은 상당히 자주 발생하게 된다. 이와 같은 순서가 바뀐 패킷이 늦게 들어온다면 수신자는 그 패킷을 잃어버린 것으로 간주하고 재전송을 요구하게 된다. 특히 빠른 재전송(fast retransmission)을 사용하는 경우 세 패킷의 시간차 후에 전송이 되더라도 재전송이 요구되며, 이때 슬로우 스타트(slow start)의 임계치(threshold)는 반으로 줄어들고, 재전송 타이머(RTO)는 재설정이 일어난다.[4] 심하게 순서가

바뀌어 전송된다면, 재전송 타이머 역시 타임아웃(time out)이 일어난다. 결국 잘못된 순서로 보내진 패킷에 의해 잘못된 혼잡회피 동작이 일어나는 것이다. 무선 Ad-hoc네트워크 환경에서 순서가 바뀐 패킷 전달이 자주 일어날 수 있고, 이는 TCP성능을 크게 떨어뜨리는 원인이 된다.

2.1.2. 네트워크 분리(partitioning) 문제

다음으로 기본적으로 Ad-Hoc 네트워크는 여러 개의 다중 홉 무선링크에 의해 구성되므로, 예측하지 못한 노드의 움직임으로 인해 연결된 루트가 갑자기 상실될 수 있다. 이 경우 연결되었던 노드들간에 장시간 동안, TCP에서 네트워크의 분리(partition)현상이 일어날 수 있다. 이 경우 또한 기존의 네트워크 혼잡과는 다른 이유임에도 불구하고, 혼잡회피 동작이 일어난다. 더욱이 네트워크 분리는 RTO의 연속적 타임아웃이 일어나고, 결국 TCP 지수적 백오프(exponential backoff)기법이 유발한다. 재전송 타이머(RTO)의 타임아웃 시간이 계속적으로 두 배씩 늘어나게 된다. 그 후에 링크가 회복되어 재전송이 일어난다 하더라도 결국 다시 통신이 일어나기 위해서는 상당한 시간이 흘러 가버린다. 결국 잘못된 혼잡 회피 동작과 링크 회복 후 전송에 필요한 시간이 상당히 증가함으로써, TCP성능이 크게 저하된다. 통신 기회자체가 줄어들며, 상대적으로 통신 시간이 증가하게 된다.

2.2. Ad-hoc 네트워크 환경에서의 TCP기법

최근 Ad-hoc 네트워크 환경에서의 TCP 성능향상에 대한 몇 가지 기법들이 소개되고 있다. TCP-F(TCP-Feed back)[5] 는 중간 노드를 이용한 일종의 피드백 메커니즘이다. 설정된 경로가 잘못된 경우 중간 노드가 관찰을 통해 경로가 실패를 감지하여, 경로 실패 공지(Route Failure Notification)을 통해 Sender에게 전달하게된다. 반대로, 중간 노드는 경로가 재 설정되면, 경로 재설정 공지(Route Re-establishment Notificaiton)를 Sender에게 역시 전달하여 재설정을 처리한다. ELFN(Explicit Link Failure Notification)[6]은 역시 경로 실패를 감지해내기 위해 DSR 라우팅 프로토콜을 이용한다. 결과적으로 경로 연결 실패를 감지하면, ELFN 메시지를 통해서 해당 노드에서 Sender로 경로가 연결이 실패되었음을 알리게 된다. Fixed RTO[7]의 경우는 연속적인 타임아웃이 발생하면, 이것은 경로 실패로 보고, 재설정이 일어날 때까지 RTO를 더 이상 증가하지 않고 멈추어 버린다. ATCP(Ad hoc TCP)[8]는 기존의 TCP에 대한 개선 없이, 새로운 레이어(ATCP)를 삽입하고 다양한 상태를 가정하여, 성능을 개선한다. 또한 잘못된 순서로 전달되는 패킷 전달을 감지하는 메카니즘을 소개한 TCP DOOR[4]도 있다. TCP DOOR기법은 헤더에 한 바이트 또는 두 바이트의 옵션을 추가하여 작동된다. 이것은 일종의 오버헤드(overhead)가 되는 것이다. 이러한 기법들은 구현이 어렵거나, 특정 상황에 부합되는 특수한 피드백 기법에 크게 의존하고 있다.

3. 제안하는 성능향상 기법 및 분석

3.1. 순서가 바뀐 패킷 전달 검사 알고리즘

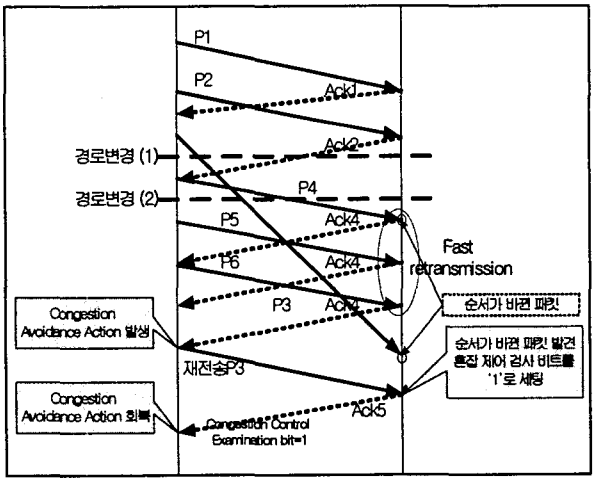
패킷의 순서가 바뀐 경우, 이에 대한 주된 원인은 Receiver와 Sender사이에 전달 경로가 바뀐 것이다. 이로 인해 발생되는 재전송이나 재전송 타임아웃을 막기 위해, 경로가 바뀐 경우 이에 대한 정보를 알려주거나 전달해 줄 수 있는 기법이 필요하다. 다음 알고리즘(1)과 같은 알고리즘에 의해서 단순히 순서가 바뀐 패킷을 검사해 낼 수 있다. TCP 패킷의 시퀀스 넘버는 기본적으로 증가하는 것이므로 이것을 비교하면 순서가 바뀌었는지 아닌지를 알아낼 수 있다. 그러나 여기서 고려해야 할 것이 재전송 패킷이다. 재전송 패킷은 시퀀스 넘버와는 무관하게 재 전송이 된다. 따라서 이 패킷을 제외한 패킷을 비교해야한다. 이를 해결하기 위해 헤더에 1bit의 재전송 비트를 추가하여 이를 검사한 후에, 순서를 검사하는 방법을 취하면 된다.

[알고리즘 1] 순서가 바뀐 패킷 전달 검사 알고리즘

```
last_seqno = last packet sequence number;
if retransmission bit == 0 then
        now_seqno = now packet sequence number;
        if last_seqno < now_seqno then
                result = order_ok;
        else
          if last_seqno > now_seqno then
                result = out_of_order;
          end if
        end if
end if
```

3.2. 순서가 바뀐 패킷 전달에 대한 처리

위와 같은 알고리즘의 결과, 순서가 바뀐 것이 확인되면 기존의 TCP 혼잡제어 모드에 의해 앞에서 말한 성능의 감소를 가져올 수 있다. 성능 감소를 막기 위해, 잘못 발생된 혼잡제어가 발생한 경우 이를 멈추고, 혼잡윈도우를 원래 크기로 회복 시켜 주어야함을 알리는 혼잡 제어 확인 비트가 필요하다.<그림1>



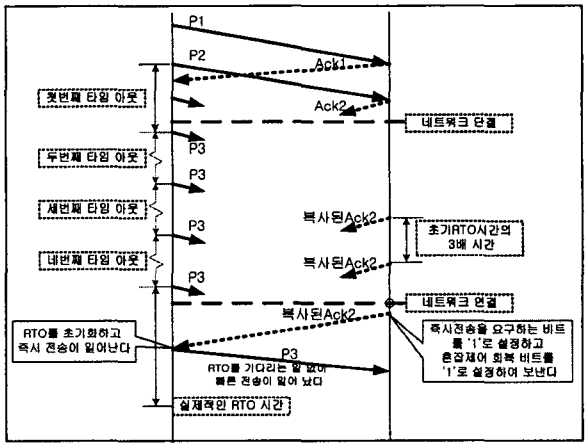
<그림 1 > Fast Retransmit에 의한 잘못된 동작의 해결

Receiver는 알고리즘(1)에 따라 순서가 바뀌어 온 패킷을

찾아내고, 이에 대한 ack의 헤더에 혼잡제어 확인 비트를 첨가하여 보낸다. 이를 받은 Sender는 최근 일정한 시간 T 동안(4×RTT)에 TCP가 혼잡 회피 동작(congestion avoidance action)에 들어가 있었다면서 이 혼잡 회피 동작 상태에서 빠져나오며, 기존의(혼잡 회피 동작 전에) 윈도우 사이즈를 회복한다. 이는 순서가 잘못 전달된 패킷에 의해 발생하는 잘못된 혼잡 제어를 회복하여 성능 향상을 꾀하는 것이다. 이를 위해서는 기본적으로 혼잡 회피 동작 상태로 들어 갈 때의 윈도우 사이즈를 기억하고 있어야 한다.

3.3. 장시간동안 네트워크 분리(partitioning)에 대한 처리 기법

일정시간동안 네트워크 분리현상이 일어나는 경우, Sender가 보내는 재전송패킷이 계속적으로 전달되지 못함에 의해서 재전송 타이머(RTO) 시간이 상당히 길어진다. 이런 형상을 막기 위해 RTO값을 초기값으로 회복시켜줄 뿐만 아니라 즉시 전송이 일어나도록 해 줄 필요가 있다. 또한 재전송에 의해 발생한 잘못된 혼잡 회피 동작에서 빠져나와 네트워크 분리전의 상태로 전송이 회복되어야 한다. 이를 위해 2 비트를 헤더에 추가했다. 즉 즉시 전송 비트, 혼잡제어 회복 비트를 ACK의 헤더에 추가한다.<그림2>

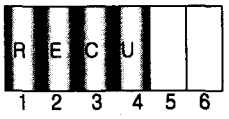


<그림 3>

통신 중단후, 일정 시간(RTT×5) 후부터 이렇게 추가된 비트를 가지고 있는 Ack패킷을 초기 RTO에 세배만큼(전형적인 초기 RTO타임이 6초이므로 18초면 적당하다)의 시간마다 복사하여 전송한다. 즉시전송비트와 혼잡제어비트가 '1'로 설정된 Ack를 받은 Sender는 재전송 타이머를 초기화하고, 더 이상 타이머를 기다리지 않고 바로 요구하는 패킷을 전송하게 된다. 이와 같은 동작에 의해 RTO에 의해 기다려야하는 시간을 크게 단축할 수 있다.-이론적으로 통신후 최대18초 후면 통신이 가능하다. 또한 TCP가 혼잡 회피 동작(congestion avoidance action)에서 빠져나오며, 기존의(혼잡 회피 동작 전에) 윈도우 사이즈를 회복하도록 만든다.

3.4. Ad hoc 비트들

이제까지 제시한 추가 비트들을 모아서 정리하면 다음과 같다. (그림3) 여기서 사용하는 비트들은 모두 TCP헤더에 추가하면 된다. TCP헤더에서 사용 할 수 있는 예약된 비트(Reserved bit)가 6비트 있으므로 이것을 이용하면 된다. 이 논문에서는 이해가 쉽도록 4 비트를 그대로 사용하였으나, 이것은 단순한 조합으로 2 비트면 충분히 기능을 발휘할 수 있다.



<그림 3>

1 bit : Retransmission bit
2 bit : cogestion control Examination bit
3 bit : Congetsion control recovery bit
4 bit : Urgent transmission bit

5. 결 론

Ad-hoc 네트워크의 TCP 환경에서 기존의 해결기법과 같이 특수한 프로토콜에 의존하거나 피드백 메커니즘에 의해 검출한다면, 구현이 어려울 뿐만 아니라 효율적이지 못하다. 제시한 기법은 지금까지의 혼잡 제어를 Sender에게 맡기던 방법에서 벗어나 Receiver가 직접 참여하여 공지함으로써, 잘못된 혼잡 제어 행동을 회복하고, 통신 시간의 증가를 최소화하는 방법이다. 복잡한 방법의 제어에서 벗어나 상당히 단순한 접근을 가능하게 하였으며, Ad-Hoc 노드들의 예측할 수 없는 움직임에 의해 TCP에서 발생하게 되는 성능 감소를 상당히 줄일 수 있다.

[1] http://www.ietf.org/html.charters/manet-charter.html
[2] http://w3.antd.nist.gov/wctg/manet/index.html
[3] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In Proceedings of ACM SIGCOMM'96, Aug. 1996.
[4] F. Wang and Y. Zhang. Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response. In Proceeding of Mobihoc'02, June 2002.
[5] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback-based scheme for improving TCP performance in ad hoc wireless networks. IEEE Personal Communications Magazine, page 34~39, Feb. 2001.
[6] G. Holland and N. Vaidya, Analysis of TCP Performance over Mobile Ad Hoc Networks, ACM Mobile Communications Conference(MOBICOM'99), Aug. 1999.
[7] T. D. Dyer and R. V. Boppana. A comparison of TCP performance over three routing protocols for mobile ad hoc networks. ACM International Symposium on Mobile Ad Hoc Networking & Computing, Oct. 2001.
[8] J. Liu, S. Singh. ATCP: TCP for Mobile Ad Hoc Networks. IEEE Journal on selected areas in communications, vol. 19, No. 7, July 2001.