무선 환경에서의 TCP 성능향상 방안

강 인석[U] 문 영성
숭실대학교 컴퓨터학과
tuneful@sunny.ssu.ac.kr, mun@computing.ssu.ac.kr

Mechanism for Improving TCP Performance over Wireless Links

Inseok Kang[U] Youngsong Mun
School of Computing, Soongsil University

요 약

현재 가장 널리 쓰이는 수송계층 프로토콜인 TCP는 패킷 손실의 원인을 망의 혼잡 때문에 일어난다고 가정하고 있으므로 기존의 유선망과 고정 호스트로 이루어진 전통적인 네트워크에 적합하다. 그러나 무선 링크에서의 패킷 손실은 대부분 혼잡에 의해서가 아니라 높은 에러율과 핸드오프에 의해 발생하게 되므로 기존의 TCP를 그대로 사용하면 불필요한 혼잡제어 메커니즘의 호출로 성능의 저하를 가져온다. 현재까지 무선환경에 적합한 TCP를 위한 많은 방안이 제시되고 있지만 근본적인 해결책을 제시하지 못하고 있다.

본 논문에서 제안하는 기법은 패킷손실이 유선링크에서 일어나는 것인지, 무선링크에서 일어나는 것인지를 판별하여 패킷손실이 유선링크에서 일어난 경우는 기존의 혼잡제어 메커니즘을 호출하여 재전송하고, 무선링크에서 일어난 경우는 혼잡제어 메커니즘을 호출하지 않고 재전송 하여 성능을 개선한다.

1. 서론

노트북, PDA 같은 휴대형 컴퓨터는 물론 이동 전화기를 이용해 인터넷에 무선으로 접속하고자 하는 욕구가 증가하고 있다. 이러한 무선망은 신호의 페이딩 및 간섭 등과 같은 신호 잡음과 핸드오프 등으로 유선망 보다 오류율이 높다. 기존의 TCP는 무선 환경을 고려하지 않은 프로토콜로서, 무선망과 연동될 경우 많은 성능 저하를 가져온다. 본 논문에서는 무선환경에 적합한 TCP로서 패킷 손실이 유선링크에서 일어난 것인지, 무선 링크에서 일어난 것인지를 구별하여 서로 다르게 처리해 주는 방안을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 무선환경에서 TCP 성능향상을 위해 기존의 제안된 기법들을 살펴보고, 3장에서는 새롭게 제안하는 기법에 대해 설명한다. 4장에서는 본 논문의 결론과 향후 연구 계획에 대해서 기술한다.

* 본 연구는 학술진흥재단 선도연구자 지원 2000-041-E00266의 지원을 받았음.

2. 무선 링크에서 TCP 성능향상을 위한 기존의 제안된 기법

2.1 Indirect TCP (I-TCP)

I-TCP는 고정 호스트와 이동 호스트간의 End-to-End 연결을 고정 호스트와 기지국과 같은 Mobile Support Router(MSR)와의 유선망 연결과 이동 호스트와 MSR간의 무선망 연결로 분리하여 각각의 망에 망 특성을 고려한 TCP를 적용시켜 전송 성능의 향상을 이루고자 하는 방법이다[1]. 그러나 이 경우 매 패킷이 TCP 프로세싱을 두 번씩 해야 하고, MSR에서는 고정 호스트와 이동 호스트 각각의 socket 정보를 유지해야 하는 오버헤드가 발생한다.

또한 실제로 패킷이 이동 수신자에 도착하기 전에 ack가 먼저 송신자에게 도착하는 경우가 발생할 수 있으므로 TCP의 종단간 작동 의미(end-to-end semantics)에 어긋나는 단점이 있으며 이 방법은 핸드오프 시에 더 큰 지연을 갖는다고 알려졌다[2].

2.2 TCP SACK

송신자와 수신자의 종단간에 TCP SACK 옵션[3]을 지원하는 TCP를 사용하게 되면 무선 링크에서 생긴 버스티한 패킷 손실에 대해서 TCP Reno[5]나 New-Reno를 사용할 경우보다 더 좋은 성능을 나타낸다는 연구가 있다.[2] 그러나 종단간 TCP SACK 만을 사용할 경우에는 무선 링크에서 발생한 패킷 유실의 재전송을 위해 종단간 재전송을 실시하여 유선 링크까지 영향을 받는 것이 단점이다. 즉 고정 호스트에서 MSR까지 잘 도착하고 무선 링크에서 손실된 데이터의 재전송을 위해서 다시 유선 링크에서의 대역폭이 사용된다.

2.3 Snoop

Berkeley 대학에서 제안한 Snoop 프로토콜[4]은 무선망에서의 높은 비트 에러율을 해결하기 위해서 MSR에 Snoop 모듈을 적용하였으며, 핸드오프 시 발생하는 데이터 손실을 줄이기 위해서 새로운 방식의 라우팅 프로토콜을 제시하였다. Snoop 모듈의 수행 방법은 다음과 같다. 고정 호스트에서 이동 호스트로 데이터를 전송할 경우, 고정 호스트가 전송한 패킷이 새로운 패킷이라면 이를 MSR에 저장(caching)하고 이동 호스트로 전송한다. Snoop 모듈은 이동 호스트에서 보내는 모든 ack를 관찰하고 있다가 패킷 손실이 발견되면 이미 MSR에 저장해 둔 패킷을 이동 호스트로 재 전송한다. MSR은 이동 호스트에서 보낸 중복된 ack를 고정 호스트로 보내지 않음으로써 고정 호스트에서 불필요한 혼잡 제어 메커니즘의 호출을 방지한다.

Snoop 모듈에는 이런 일련의 작업을 관리하는 snoop_data()와 snoop_ack()라는 2개의 procedure가 있다. snoop_data()는 이동 호스트로 전송되는 패킷 중 아직 ack를 받지 못한 패킷을 MSR에서 저장하고 이를 이동 호스트로 포워딩 하는 역할을 하며, snoop_ack()은 이동 호스트로부터 들어오는 ack를 처리하고 MSR에서 이동 호스트간의 지역 재전송을 관리한다. 이러한 지역 재전송 방법은 송신자 측에서의 불필요한 혼잡 제어나 회피 메커니즘의 호출을 막아서 전체적인 성능의 향상을 가져올 수 있다.

3. 제안하는 기법

Snoop 기법[4]에서 MSR의 캐쉬가 오버플로우 되면 전체 연결의 성능이 떨어진다. MSR에서 저장하지 못한 패킷이 무선 링크에서 손실되어 재전송이 요청되면 혼잡에 의한 패킷손실이 아님에도 불구하고 송신 쪽에서는 윈도우 사이즈를 줄이는 혼잡 제어 메커니즘을 호출한다. 이 단점을 극복하는 방안으로 해당 TCP connection에서 전송되는 패킷과 sequence number(SN)를 각각 저장하여 MSR에서 재 전송할 데이터가 없더라도 SN이 저장되어 있으면 유선 상에서는 손실이 없음을 알고 기존의 혼잡 제어 메커니즘이 수행되지 않도록 한다.

[그림 1]에서는 sequence number cache table(SNCT)과 data packet cache table(DPCT)의 구조를 보여주고 있다. SNCT의 두 번째와 세 번째 항목이 null로 되어 있는 것은 DPCT에서 오버플로우가 발생하여 데이터를 저장하지는 못하고 SN 값만을 저장했기 때문이다.

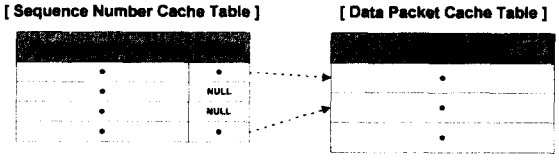

그림 1. cache table의 자료구조

MSR에서는 이동노드가 보내는 ack패킷과 자신이 저장해 놓은 SN을 비교하여 유·무선 링크 모두에서 손실이 없는 경우인지, 유선링크 상에서의 손실인지, 무선링크 상에서의 손실인지를 판단하게 된다. 이를 판별하는 알고리즘을 도식화하면 [그림 2]와 같다.

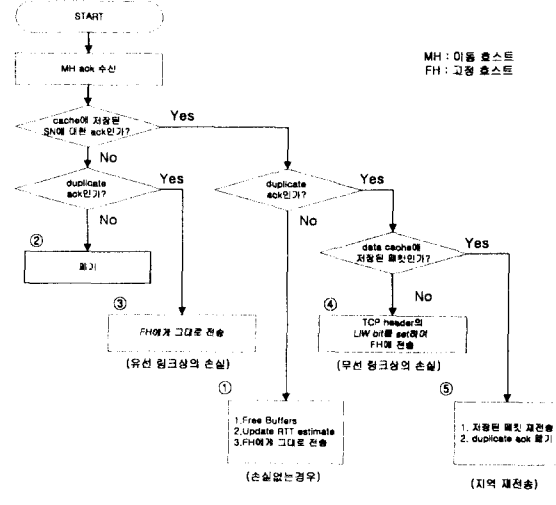

그림 2. decision 알고리즘

① 손실이 없는 정상적인 경우 :
저장된 SN에 대한 ack이고 중복된 ack가 아니면 SNCT과 DPCT에서 ack된 모든 패킷들의 항목을 삭제한다. 무선 구간에 대한 RTT 측정도 이때 갱신된다. 또한 이 ack는 유선망의 고정 호스트로 그대로 전송된다.

② 오류가 있는 ack :
SNCT에 저장되어 있지 않은 SN에 대한 ack이고 중복된 ack도 아닌 경우로 거의 발생하지 않는 상황이다. 이 ack는 MSR에서 마지막으로 수신된 ack보다 작은 ack에 해당하며, 이 ack

는 폐기시키고 다음 ack를 처리한다.

③ 유선 링크상의 손실 :
SNCT에 저장되지 않은 SN에 대한 중복된 ack를 받은 것은 고정호스트가 보낸 패킷이 MSR까지 전송되지 않은 경우이다. 이 중복된 ack를 그대로 송신측에 전달하여 기존의 혼잡 제어 메커니즘을 호출하고 윈도우 사이즈를 줄여서 재전송 한다[5].

마지막으로, SNCT에 저장되어 있는 SN에 대한 중복된 ack인 경우는 무선 링크 상에서 손실이 발생한 경우로 다시 다음의 두 가지로 세분된다.

④ 중복된 ack를 통해서 재전송이 요구되는 패킷이 MSR의 DPCT에 저장되어 있지 않은 경우 :
이 경우는 DPCT에 오버플로우가 생겨서 송신 측으로부터 전송된 패킷을 DPCT에 저장할 수 없었고, 무선 링크 상에서 그 패킷이 손실된 경우이다. 하지만 SNCT를 통해 유선 상에서는 손실되지 않은 패킷임을 알 수 있으면, 기존의 혼잡 제어 메커니즘을 호출하지 않고 재전송 한다.
이를 유선망의 고정 호스트에게 알리기 위해서는 중복된 ack의 TCP 헤더에 [그림 3]과 같이 새롭게 정의한 LIW(Loss In Wireless) flag를 1로 세팅하여 보낸다.

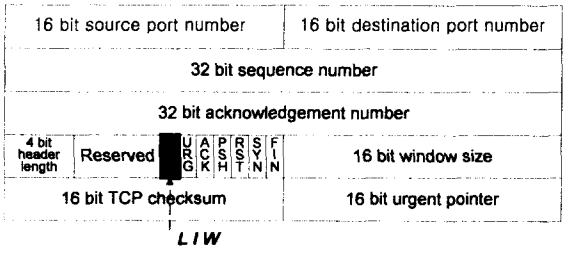

그림 3. 추가된 LIW flag

송신측에서는 중복된 ack를 수신하게 되면 먼저 LIW flag를 조사하여 1로 세팅되어 있으면 윈도우 사이즈와 slow start threshold 값을 기존의 값으로 유지한 채 재전송을 한다. 이를 pseudo code로 나타내면 다음과 같다.

```
tcp_recv()
{
   if ( (LIW == 1) && (fast retransmit) )
   { hold ssthresh;
     hold cwnd;
     retransmission_timer = 0;
     retransmit lost_pkt;
     return;
   }
   else if( (fast retransmit) || (timeout) )
   { update ssthresh;
     update cwnd;
     retransmission_timer = 0;
     retransmit lost_pkt;
     return;
   }
   /* Other packet processing */
}
```

⑤ 중복된 ack를 통해서 재전송을 요청하는 패킷이 MSR의 DPCT에 저장되어 있는 경우 :
저장된 패킷을 이동 호스트에 보내 지역 재전송을 수행하고 중복된 ack는 폐기시킨다. 재전송을 한 경우라 할지라도 바로 DPCT에서 삭제하는 것이 아니라 중복된 ack가 아닌 정상적인 ack를 수신하였을 때 비로소 DPCT에서 패킷을 삭제하게 된다.

4. 결 론

본 논문에서는 MSR의 캐쉬를 이용한 지역 재전송 기법을 최대한 이용하면서, 데이터 캐쉬가 오버플로우 됐을 때도 저장된 SN값을 이용해 패킷손실이 유선링크에서 일어난 것인지, 무선 링크에서 일어난 것인지를 구별하여 각기 다르게 처리해 주는 방안을 제시하였다. 향후 연구과제로는 무선환경의 에러 특성을 고려한 효율적인 MSR의 캐쉬 관리 기법과 핸드오프를 고려한 TCP 성능 향상 방안을 들 수 있겠다.

참고 문헌

[1] Ajay Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," Tech. Rep., Rutger University, May 1995.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms For Improving TCP Performance Over Wireless Links," IEEE/ACM Transactions on networking, vol. 5, no. 6, Dec. 1997.

[3] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "Selective acknowledgment options," RFC-2018, 1996.

[4] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, "Improving TCP/IP performance over wireless networks," In Proc. 1st ACM Int'l Conf. on Mobile Computing and Networking (Mobicom), Nov. 1995.

[5] W. Richard Stevens, TCP/IP Illustrated, Volume 1 : The Protocols, Addison-Wesley, Reading, Massachusetts, 1994.