제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

유무선 게임 네트워크에서 짧은 TCP 트래픽의 성능개선 기법

진교홍*, 김진덕**
*동의대학교 멀티미디어공학과 ** 동의대학교 컴퓨터공학과
*e-mail:khjin@dongeui.ac.kr

Performance Improvement Method for Short TCP Traffic in Wired/Wireless Game Network

Kyo-Hong Jin
Dept of Multimedia Engineering, Dong-Eui University

요 약

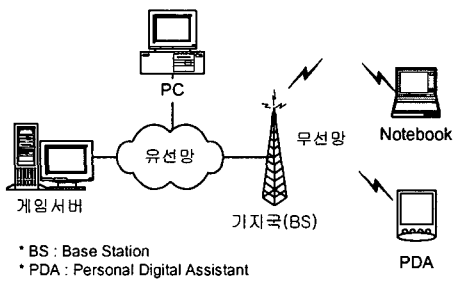
인터넷의 폭발적인 발전과 더불어 온라인 게임은 인터넷 산업을 주도하는 원동력이 되고 있다. 최근 온라인 게임은 유선통신망에서 무선통신망으로 사업범위를 넓혀 핸드폰이나 PDA 등의 무선단말기에서도 게임을 즐길 수 있게 되었다. 그러나, 온라인 게임의 전송프로토콜로 이용되고 있는 TCP 프로토콜은 무선통신 환경의 높은 비트 오류율로 인하여 폭주제어 알고리즘이 오동작을 일으키며 이로 인하여 전체적인 게임 서비스의 성능이 저하된다. 따라서 본 논문에서는 무선 온라인 게임 네트워크에서 TCP 프로토콜의 성능을 개선시키기 위한 응답패킷 분할 기법을 제안하였다. 제안된 기법은 컴퓨터 시뮬레이션을 통해 성능을 분석하였으며, 그 결과 기존의 TCP 프로토콜에 비하여 성능이 향상되었음을 확인하였다.

1. 서론

1969년 ARPANET의 등장 이후로 인터넷은 급속히 발전하여 현재 국내의 인터넷 사용자만 하더라도 2002년 6월 자료에 따르면 2,500만명을 넘어섰다[1]. 인터넷을 이용하기 위한 서비스 단말기는 주로 일반 PC가 중심이 되어왔지만, 현재는 노트북, 셀룰러폰, 및 PDA(Personal Digital Assistant) 등의 다양한 무선 단말기가 선보이고 있다.

한편, 한국인터넷정보센터의 인터넷 이용자 설문조사 자료에 따르면 자료정보검색을 위한 인터넷 이용률이 49%로 가장 높게 나타났으며, 게임이 25.7%, 전자우편이 13.5%로 나타났다[2]. 이 조사에서도 알 수 있듯이 인터넷을 통한 온라인 게임이 큰 비중을 차지하고 있으며, 앞으로도 이 비율은 더욱 커질 것으로 예상되고 있다. 또한 기존의 PC 중심의 유선통신 기반 게임방식에서 셀룰러폰이나 PDA를 이용한 무선통신 기반 게임방식도 더욱 확산될 전망이다.

본 논문은 2002년 부산테크노파크 사업(BTP-BDE-9)의 일환으로 연구되었음



(그림 1) 유무선 통합 게임 네트워크

인터넷을 이용한 온라인 게임은 모두 통신을 기반으로 게임서버에 접속하여 원거리의 게임 클라이언트와 게임을 즐기는 방식이다. 일반적으로 게임서버는 고성능의 컴퓨터로 유선망에 연결되어 있으며, 게임 클라이언트는 유선통신망을 이용하거나 핸드폰이나 PDA 등의 무선단말기로 무선통신망을 이용하여 게임을 하게 된다. 게임서버와 게임 클라이언트간은 인터넷을 통해 연결되며 TCP/IP 프로토콜을 이용하여 게임정보를 주고받게 된다. 그림 1은 전형적인 게임 네트워크의 구조

를 보여주고 있다.

일반 유선망에 연결되어 있는 경우와는 달리 무선망을 사용하는 경우 신호의 페이딩이나 간섭현상으로 인하여 비트 오류율이 높다. 즉 비트 오류율이 10^{-3}~10^{-5}으로 기존 유선망의 10^{-6}~10^{-12}에 비해 상당히 높으며, 이로 인하여 패킷손실이 빈번히 발생될 수 있다. 또한 셀룰러 망에서는 이동단말기가 한 셀에서 다른 셀로 이동할 경우 새로운 주파수 채널을 할당하는 핸드오프(Handoff) 과정을 수행하게 된다. 이 핸드오프는 수십 msec에서 길게는 수초가 걸리는 작업이므로 일시적으로 연결이 단절될 수 있다. 따라서 무선망은 높은 비트 오류율과 연결단절과 같은 열악한 환경으로 인하여 유선망에서와는 달리 패킷손실이 자주 발생된다.

이러한 높은 패킷 손실율로 인하여 대부분의 인터넷 서비스에서 사용되는 TCP 프로토콜의 폭주제어(Congestion Control) 알고리즘은 오동작을 일으키게 되어 인터넷 기반 게임 서비스의 성능을 저하시킨다.

따라서 본 논문에서는 무선통신망 환경에서 TCP 프로토콜의 성능향상 기법을 제안하였으며, 온라인 게임을 위해 사용되는 10KB 정도의 짧은 메시지에 대한 TCP 프로토콜의 성능을 분석하였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 기존의 연구결과를 비교분석하고 3장에서는 제안된 기법을 설명하였다. 그리고 4장에서는 성능분석 결과를 제시하였고, 마지막으로 5장에는 결론을 기술하였다.

2. 기존의 연구

기존의 연구결과들은 크게 End-to-end 기법, Split-connection 기법, 및 Link Layer 기법 등의 3 종류로 분류된다.

2.1 End-to-end 기법

End-to-end 기법은 기존 TCP 프로토콜과 동일하게 종단간 연결을 설정하고, 무선통신 환경에 적합하도록 TCP 프로토콜을 수정하는 기법이다. 관련된 연구로는 빠른 재전송[4], 및 TCP-R[5] 등이 있다. 이 절에서는 대표적인 빠른 재전송 기법에 대해서만 설명한다.

빠른 재전송 기법의 동작과정은 다음과 같다. 먼저 무선단말기의 핸드오프로 인해 데이터가 손실되면, 무선단말기의 TCP는 즉시 서버에게 세 개의 동일한 응답패킷을 보내준다. 세 개의 동일한 응답패킷을 수신한 서버의 TCP는 재전송 타이머의 타임아웃을 기다리지 않고 Fast Retransmit 알고리즘을 수행하여 손실된 패킷부터 재전송을 실시한다. 한편, 무선단말기에서 송신한 데이터가 핸드오프로 인해 손실되면 무선단말기는 서버의 TCP에게 핸드오프 완료신호를 보내어, 서버에서 손실된 패킷에 대한 세 개의 동일한 응답패킷을 보내도록 한다. 이에 따라 무선단말기에서는 Fast Retransmit 알고리즘이 수행되어 손실된 패킷부터 재전송한다.

이 기법은 TCP 프로토콜의 End-to-end Semantics를 유지하는 장점을 가지지만, 무선단말기의 TCP 프로토콜을 수정해야 하는 문제점이 있다.

2.2 Split Connection 기법

Split Connection 기법은 유무선망의 특성을 고려하여 종단간의 TCP 연결을 서버와 기지국간의 연결과 기지국에서 무선단말기간의 연결로 분리하여, 서버에서는 무선망 상에서 발생되는 패킷 손실을 인식하지 못하도록 하는 기법이다.

Split Connection 기법을 이용한 기존의 연구에는 I-TCP(Indirect-TCP)[6], M-TCP[7], 및 METP[8] 등이 있으며, 각각은 적용된 메커니즘이 조금씩 다를 뿐 TCP 연결을 두 부분으로 나눈다는 점에서는 유사하다. 이 절에서는 Split Connection 기법의 대표적인 I-TCP의 동작원리에 대해서 설명하였다.

I-TCP 프로토콜은 서버에서 무선단말기로 전송된 데이터는 기지국에서 무선단말기로 중계해 주며 이에 대한 응답패킷이 이동단말기로부터 도착되기 전에 기지국에서 대신 서버로 전달하여 준다. 그리고 무선망 상에서 패킷손실이 발생되면 이를 서버에게 알리지 않고 기지국에서 자체적으로 무선단말기에 재전송하여 서버의 TCP 프로토콜 성능에는 영향이 없도록 한다.

이 기법은 기지국과 무선단말기 간에는 TCP 대신 보다 간단한 새로운 프로토콜을 사용할 수 있는 장점이 있지만, 기지국에는 패킷손실에 대비하여 많은 양의 버퍼가 필요하며 TCP 프로토콜의 End-to-End Semantics가 파괴되는 문제점이 있다.

2.3 Link Layer 기법

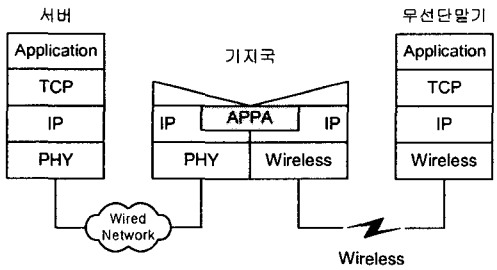
Link Layer 기법은 무선 망에서 패킷손실이 발생되면 이를 송신측 TCP에서 감지하기 전에 기지국의 데이터 링크 계층에서 재전송하는 기법이다. 대표적인 연구로는 Snoop 모듈[9]과 AIRMAIL[10]등이 있으며, 이 절에서는 대표적인 Snoop 모듈에 대해서 알아본다.

Snoop 모듈은 기지국의 데이터 링크 계층에 데이터 패킷을 저장하기 위한 버퍼를 마련한다. 그리고 서버에서 무선단말기로 데이터를 전송하는 경우, 기지국의 버퍼에 전달되는 데이터를 저장해 두고, 무선단말기로부

터 응답을 받지 못하면 무선망 상에서 패킷이 손실된 것으로 판단하고 기지국에서 버퍼에 저장된 데이터를 이용하여 재전송하여 준다. 한편, 무선단말기에서 서버로 데이터를 전송하는 경우에 패킷이 손실되면 기지국에서 무선단말기로 NACK(Negative Acknowledgement) 패킷을 송신하여 무선단말기에서 해당 데이터를 재전송하도록 조치한다. 이 기법은 TCP 프로토콜의 End-to-end Semantics를 유지하여 주고 이동단말기의 TCP 프로토콜을 수정할 필요가 없다는 장점이 있지만, 데이터링크 계층의 재전송과 TCP 계층의 재전송이 중복될 수 있는 문제점이 있다.

3. 제안된 기법

본 논문에서 제안하는 기법은 서버와 무선단말기에서는 TCP 프로토콜을 수정없이 그대로 사용하고, 대신 기지국에 APPA(Acknowledgement Packet Partitioning Algorithm) 모듈을 첨가하여 TCP 프로토콜의 성능향상을 도모하였다. 먼저 APPA기법에서 사용되는 프로토콜 구조는 그림 2와 같다.

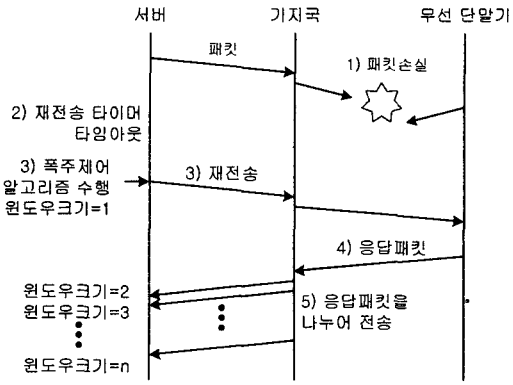


(그림 2) APPA 기법의 프로토콜 구조

그림 2에 제시된 바와 같이 서버와 무선단말기는 기존의 TCP/IP 프로토콜 구조를 그대로 사용하지만, 기지국에는 고유의 IP 라우팅 기능과 더불어 APPA 모듈이 첨가된다. 이러한 프로토콜 구조에서 서버와 무선단말기간의 데이터 송수신을 위해 종단간 호설정이 이루어지고, 평상시 기지국은 서버나 무선단말기에서 보내온 데이터를 받아 중계하는 고유의 라우팅 기능만을 수행한다.

한편, 무선망 상에 패킷 손실이 발생되면 서버에서는 마치 폭주현상이 발생된 것으로 오인하게 되며 윈도우 크기(W)를 1세그먼트나 W/2로 줄이고 TCP 프로토콜의 폭주제어 알고리즘을 동작시켜 패킷 전송 속도를 늦추게 된다. 그 이후 줄어든 윈도우 크기는 새로운 응답패킷이 서버에 도착되면 Slow Start와 Congestion Avoidance 알고리즘에 따라 증가된다.

본 논문에서는 TCP 프로토콜의 폭주제어 알고리즘은 수신된 응답패킷의 개수에 따라 윈도우 크기를 증가시킨다는 점에 착안하여, 기지국이 무선단말기로부터 재전송된 패킷에 대한 응답패킷을 수신하면 이를 여러 개로 나누어 서버에 전달하여 서버의 윈도우 크기를 빠르게 복귀시키는 방법이다. APPA 알고리즘의 동작 예는 그림 3과 같다.



(그림 3) APPA 알고리즘의 동작 예

그림 3에서 보는 바와 같이 서버에서 전송된 데이터 패킷이 1) 기지국과 무선단말기사이에서 손실되면 2) 얼마 후 서버의 재전송 타이머가 타임 아웃된다. 이에 따라 3) 폭주제어 알고리즘이 호출되고 서버의 TCP 프로토콜은 윈도우 크기를 1세그먼트로 줄인 후 재전송을 수행한다. 그 후 무선단말기에서 데이터 패킷을 제대로 수신하게 되면 4) 응답패킷을 서버로 전달한다. 이때 5) 기지국에서는 응답패킷을 그대로 서버에게 전달하지 않고 여러 개로 쪼개어 보내줌으로써 서버의 윈도우 크기는 빠르게 증가되어 TCP의 성능이 빠르게 복구된다.

4. 성능분석

시뮬레이션은 ns-simulator[11]를 사용하였으며 시뮬레이션 모델은 그림 4와 같다.

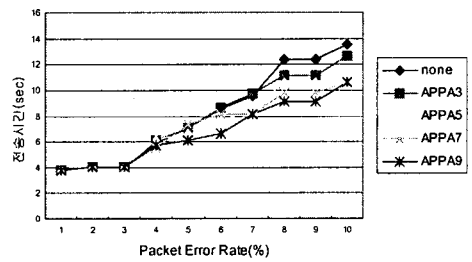
(그림 4) 시뮬레이션 모델

그림 4에서 보는 바와 같이 유무선 복합망의 특성을 고려하여 시뮬레이션 모델은 서버, 기지국, 및 무선단말기 등 총 3개의 노드로 구성하였으며, 서버와 기지국간의 링크는 100Mbps의 대역폭과 100msec의 전송지연 시간을 가정하였다. 또한 기지국과 무선단말기간은

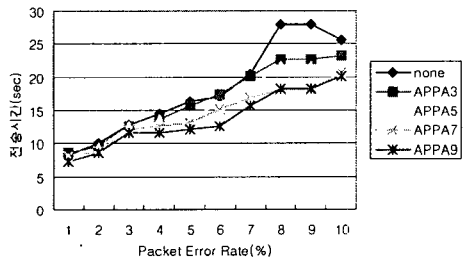
2Mbps의 대역폭과 10msec의 전송지연을 가지도록 하였으며, 이 구간에서만 패킷손실이 발생되는 것으로 가정하였다.

시뮬레이션 시나리오는 먼저 서버에서 무선단말기로 10KB~100KB 크기의 데이터를 전송하도록 하고, 세그먼트의 크기도 250B에서 1500B까지의 값으로 변화를 주었다. 그리고 기지국과 무선단말기 구간에서 패킷손실율을 1%에서 10%까지를 적용하여 시뮬레이션을 수행하였다. 또한 APPA 기법의 주요인자인 응답패킷의 개수도 여러 값을 주어 성능분석에 활용하였다.

시뮬레이션 결과를 보면 그림 5와 그림 6은 각각 전송데이타의 크기가 100KB인 경우에 세그먼트의 크기가 1000B, 500B인 환경에서 패킷 손실율과 전송시간과의 관계를 보여주고 있다. 이 두 개의 그래프에서 패킷손실율이 증가할수록 기존의 TCP보다 APPA 알고리즘을 적용한 TCP 프로토콜의 전송시간이 짧음을 알 수 있다.



(그림 5) 패킷손실율 vs. 전송시간(MSS=1000B)



(그림 6) 패킷손실율 vs. 전송시간(MSS=500B)

5. 결론

본 논문에서는 무선통신 환경에서 TCP 프로토콜의 성능을 개선하기 위한 APPA 기법을 제안하였다. 제안된 기법은 기지국에서 무선망의 패킷손실 이후에 수신된 응답패킷을 여러 개로 나누어 TCP 송신측에 전달한다. 이에 따라 TCP 프로토콜은 기존의 폭주제어 알고리즘에 따라 줄어든 윈도우 크기를 빠르게 복원하여 TCP 프로토콜의 성능을 향상시킬 수 있다. APPA 기법은 기존의 연구결과들과는 달리 TCP 프로토콜을 수정할 필요가 없으며, End-to-end TCP Semantics도 유지된다. 또한 기지국에 많은 양의 버퍼가 필요없을 뿐만 아니라 재전송이 중복되는 현상도 발생되지 않는 장점을 가지고 있다.

참고문헌

[1] 한국인터넷정보센터, "인터넷 이용자 통계," http://stat.nic.or.kr/iuser/korea_y.html

[2] 한국인터넷정보센타, "2002년 6월말 기준 인터넷 이용자수 설문조사 결과보고서," http://stat.nic.or.kr/stat_report.html, Aug., 2002

[3] W. Richard Stevens, *TCP/IP Illustrated*, Vol.1, Addison Wesley, 1994.

[4] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE Journal on Selected Areas in Communications, 13(5), June, 1995.

[5] Daichi Funato, Kinuko Yasuda and Hideyuki Tokuda, "TCP-R : TCP Mobility Support for Continuous Operation," Proceedings of International Conference on Network Protocols, pp.229-236, Oct. 1997.

[6] A. Bakre and B. R. Badrinath, "I-TCP : Indirect TCP for Mobile Hosts," Proceedings of 15th International Conference on Distributed Computing Systems, Vancouver, Canada, pp.136-143, May, 1995.

[7] K. Brown and Suresh Singh, "M-TCP : TCP for Mobile Cellular Networks," ACM SIGCOMM, Computer Communication Review, pp.19-43, July, 1997.

[8] Kuang-Yeh Wang and Satish K. Tripathi, "Mobile-End Transport Protocol an Alternative to TCP/IP over Wireless Links," Proceedings of INFOCOM'98, Vol.3, pp.1046-1053, April, 1998.

[9] Elan Amir, Hari Balakrishnan, Srinivasan and Randy H. Katz, "Efficient TCP over Networks with Wireless Links," Proceedings of 5th Workshop on Hot Topics in Operating Systems, pp.35-40, May, 1995.

[10] E. Ayanoglu, S. Paul, T.F. Laporta, K. K. Sabnani, and R. D. Gitlin, "AIRMAIL : A Link-Layer Protocol for Wireless Networks," ACM/Baltzer Wireless Networks Journal, pp. 47-60, Feb. 1995.

[11] "UCB/LBNL/VINT Network Simulator-ns (version2)," http : //www-mash.cs.berkeley.edu /ns.