무선 망을 위한 다중 파일 전송 프로토콜 설계

윤민홍° 류은석 유 혁
고려대학교 컴퓨터학과 운영체제 연구실
{mhyun, esryu, hxy}@os.korea.ac.kr

Design and Implementation of Multi-file Transfer Protocol for Wireless Network

Min-Hong Yun° Eun-Seok Ryu Hyuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

휴대폰의 성능이 급격히 향상되며 많은 사람들이 휴대폰이 PC 가 할 수 있는 많은 일을 처리할 것이라는 전망을 한다. 휴대폰이 편리한 이유는 움직이면서 네트워크에 접근할 수 있다는 것이다. 그러나 휴대폰이 PC 에 비해 열악한 자원(resource)을 가지고있기 때문에 데이터 교환에 있어서 제한이 존재한다. 또한 휴대폰 사용자의 특성은 여러 개의 파일을 각각 전송 받은 후에 하나씩 접근하는 것보다, 조금씩 받으면서 파일에 접근하는 식의 방법을 사용하는 패턴을 가지고 있다. 본 논문에서는 여러 개의 파일을 열악한 대역폭과 메모리의 한계를 갖는 휴대폰을 위해 동시에 전송하는 데에 적합한 프로토콜을 설계하면서 겪은 과정과 실험결과, 설계한 여러 프로토콜들 중 가장 성능이 좋은 프로토콜로 증명된 ' UDP best block with NAK set' 을 소개한다.

1. 서론

휴대폰 사용자의 수가 늘고 그것을 통해 정보를 얻으려 하는 사용자의 수가 늘어남에 따라 무선 망에서 데이터를 전송하는 분야에 대한 연구가 진행되고 있다. 중요한 쟁점들로는 속도와 에러 율이 있겠지만, 이에 못지않게 사용자의 패턴을 파악해 불필요한 데이터를 전송하지 않고 필요한 데이터의 전송을 앞당기는 것 또한 사용자가 짧은 응답시간을 가지고 있다고 생각하기 때문에 중요하다.[1][2]

사용자가 원하는 정보는 한번에 하나가 아니라, 두 가지 이상의 정보다. 웹 문서를 보는데 한번에 문서 전체를 통독하는 사용자는 없다. 대신 각 문서의 제목과 같은 앞부분 조금씩 만을 읽은 후 관심 있어 하는 내용, 필요한 내용으로 생각되는 문서를 정독 한다.

휴대폰을 사용해 정보를 얻으려는 사용자의 특성은 다음과 같다. 적은 양의 정보를 먼저 요구한 후 필요에 따라 전체의 정보를 요구한다는 특성, 유선 망에 이미 익숙해져 사용자가 느끼기에 많이 느린 무선 망에 쉽게 실망한다는 특성이다. 따라서 본 논문에서는 이에 대한 해결책으로 일부 자료만을 미리 전송한 후 사용자와의 상호작용을 통해 사용자가 필요로 하는 자료를 파악하여 이를 전송하는 데 적합한 프로토콜을 제안하고 이의 구현을 통한 실험 데이터를 제시한다.

2. 실험 환경

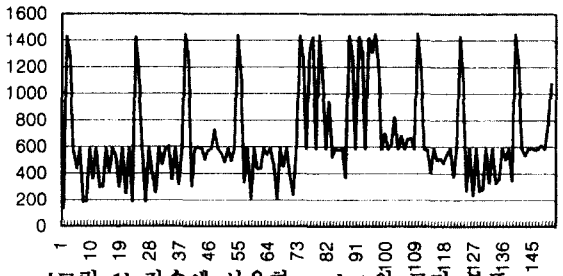
실험의 측정은 단말기의 역할로 PC 를 사용하고 휴대폰을 모뎀으로 사용해 시뮬레이션 했다. 측정에 사용된 휴대폰의 대역폭은 144Kbps 이지만 PC 에서 전화기로의 연결이 115Kbps 로 되기 때문에 이 수치를 사용해 계산했고 전송하는 데이터의 크기는 100,000bytes 이다 휴대폰이 PC 에 비해 갖는 한계 중 가장 큰 것이라 여겨지는 메모리의 한계를 시뮬레이션 하기 위해 버퍼(buffer)로 사용하는 메모리로 20Kbytes 와 데이터를 저장하는 메모리로 100,000Bytes 만이 존재한다고 가정했다. 데이터의 저장은 순차적으로만 저장이 가능하게 했다. 즉, 파일 seek 의 기능이 없다고 가정했다.

데이터를 전송할 때, 최대 전송 크기를 정했는데 이더넷(Ethernet)의 MTU 사이즈를 고려해 최대 1500bytes 이다. 어떤 네트워크를 얼마나 오랫동안 거치는지 알 수 없으므로 이것에 대한 계산은 제외하고, 대역폭만 사용해 서버측의 전송 간격을 계산했다. 서버측에서는 100Mbps 의 대역폭으로 최대 전송단위를 전송하는데, $\frac{1500 \times 8\,bit}{100 \times 1000 \times 1000\ bit/\sec} \approx 0.1(ms)$ 이 소요되지만 클라이언트 측은 115Kbps 의 대역폭으로 전송하기 때문에 $\frac{1500 \times 8\,bit}{115 \times 1000\ bit/\sec} \approx 104.3(ms)$ 이 소요된다. 이 결

과로 서버는 100ms 의 간격으로 패킷을 전송하게 했다.

파일의 전송단위는 1500bytes 로 일정하게 하지 않고, 패킷 헤더에 패킷 사이즈 정보를 넣음으로써 서버측에 유연성을 제공했다. 본 논문에 사용된 모든 실험 결과는 시시각각 변하는 무선 환경 때문에 대조군(TCP)과 실험군(UDP)을 교대로 실험했다.

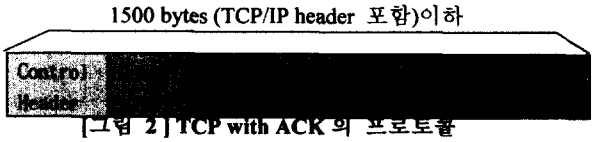
실험에 사용한 데이터는 [그림 1]에서와 같은 분포를 가지고 있다. 패킷당 평균 크기는 약 663bytes 이고 최소 136 bytes, 최대 1452bytes 의 크기이다.



[그림 1] 전송에 사용한 packet 의 크기 분포

3. TCP with ACK 와 UDP with ACK

TCP with ACK 프로토콜은 가장 간단하고 기초적인 프로토콜로 계속 소개할 다른 3 개의 프로토콜의 대조군이 되는 프로토콜이다. 이때 휴대폰은 서비스를 받는 입장이므로 클라이언트에 해당되고, 전송할 데이터를 가지고 있는 PC 는 서버에 해당된다.



[그림 2] TCP with ACK 의 프로토콜

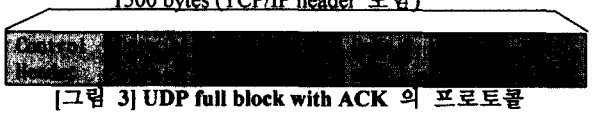
이 프로토콜은 [그림 2]같은 모습으로 설계됐다. 실제 전송하려는 데이터는 마지막에 위치하며, data size 는 실제 데이터의 크기를 의미하고, control header 는 ACK 를 전송하는 데에 필요한 패킷(packet)의 일련번호 등을 포함한다. 이 모델에서 서버는 전송이 시작되면 8 개의 패킷을 연속해서 전송한 다음 클라이언트측으로부터 각각의 패킷에 대한 ACK 를 받은 후에 다음 패킷을 보내게 된다. Transport 계층과 중복해서 ACK 를 사용 하는 이유는 실험시 TCP 전송이 되지 않는 경우가 실제로 발생했기 때문이다. 서버측에서는 일정시간동안 패킷에 대한 ACK 를 받지 못하면 가장 마지막으로 확인된 ACK 부터 8 개를 다시 연속적을 보낸다. 이 모델의 장점은 하나씩 패킷을 전송하고 모든 패킷에 대한 ACK 를 받기 때문에 완벽히 데이터를 전송한다는 점이다. 그렇지만 이 모델은 너무 속도가 느리다는 단점을 가지고 있다.

UDP with ACK 프로토콜은 데이터의 전송을 UDP 로 한다는 것만 ' TCP with ACK' 와 다르다. 이로 인해 서버측에서는, 버퍼링(buffering)하는 TCP 와달리 UDP 를 사용해 함수 호출 즉시 전송한다는 점과, transport 계층에서의 ACK 와 application 계층에서의 ACK 가 중복되지 않는다는 장점 그리고, 라우팅 과정에서는, 라우터의 버퍼가 가득찰 때 victim 이 될 가능성이 줄어들 것이라는 추정의 장점을 갖는다. 실험 결과에서도

$\frac{TCP}{UDP} \approx 1.05$ 라는 미비한 속도향상의 결과가 나왔다

4. UDP full block with ACK

' 3. TCP with ACK 와 UDP with ACK' 에서 소개한 프로토콜은 한번에 하나의 파일에 대한 데이터만 전송한다는 문제점을 가지고 있다. 이 때문에 이더넷의 MTU 인 1500bytes 를 다 채우지 않고 전송 하게 되 Per Packet Overhead 가 커지는 비효율성의 문제가 있다. 이 비효율적인 전송을 막기위해 1500bytes 가득 채웠다 또 클라이언트의 경우 연속되지 않은 패킷을 받는 경우 ' 3. TCP with ACK 와 UDP with ACK' 의 프로토콜은 나머지 패킷을 버렸지만, 이 프로토콜부터는 버퍼링을 해 10 개의 패킷을 임시로 저장할 수 있게 했다.



[그림 3] UDP full block with ACK 의 프로토콜

[그림 3]과 같이 설계한 이 프로토콜은 마지막 블럭이 아니라면 반드시 1500bytes 로 채운다. data_2 의 경우 데이터가 완전히 포함되지 않고 잘려나갈 수 있지만 그런 경우 나머지 data 들은 다음 블럭에서부터 시작한다.

평균 성능향상은 $\frac{TCP}{UDP} \approx 1.6$ 이다.

TCP/UDP 가 2~3 이 되는 성능 향상이 있었지만 평균 성능 향상이 낮은 이유는, 크기를 알려주는 필드에 오류가 발생하면 오류가 발생한 시점을 찾을 수 없어 파일 전송이 무효화되기 때문이다. 즉, 한번 오류가 발생하면 이 오류는 연속적으로 전파되기 때문에 오류가 발생했다는 사실 조차 찾지 못할 수 있고, 찾는다 하더라고 발생한 시점을 찾는 것이 불가능하다는 것이다

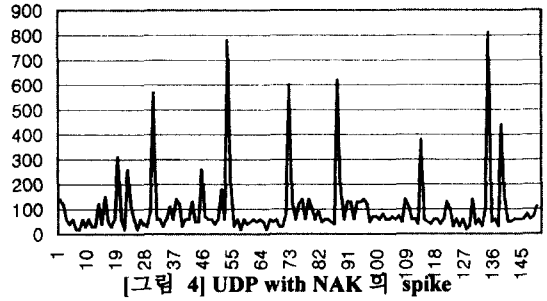
5. UDP with NAK 과 UDP best block with NAK set

이 프로토콜이 앞의 3 개의 프로토콜에 비해 가장 뛰어난 성능 향상을 보인 프로토콜이다. 소개하는 두개의 프로토콜은 휴대폰 메모리 제약을 고려해 10 개까지의 패킷을 저장할 수 있는 버퍼를 유지해 패킷 손실이 발생해도 10 개까지는 버퍼에 유지할 수 있다.

5.1 UDP with NAK

이 프로토콜은 한번에 하나의 파일에 대한 데이터만 전송하며 NAK 은 패킷 각각에 대해 하나씩 보낸다. NAK 을 사용해서 평균 $\frac{TCP}{UDP} \approx 1.45$, 최대 2.4 의 성능 향상이 있었고, spike 에서 속도향상 가능성이 존재한다.

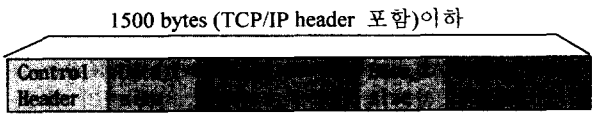
[그림 4]의 그래프는 패킷 전송간격을 나타낸 그림으로 X 축은 패킷의 일련번호이고 Y 축은 millisecond 단위의 시간이다. 속도저하의 원인은 약 10 번 정도에 거쳐 나타나는 spike 에 있으며 이것은 10 개 까지 저장 가능한 버퍼가 가득 차 일련번호가 맞지않는 나머지 패킷들을 drop 하는데 기인한다. 버퍼에 저장 가능한 패킷의 개수가 제한되어 있기 때문에 이 spike 를 완벽히 없앨 수 있는 방법은 없다. 최선책은 패킷을 여러 개 묶어 1500bytes 의 크기로 만들어 전송 횟수를 최대한 줄임으로써 drop 의 횟수를 줄이는 것뿐이다.



[그림 4] UDP with NAK 의 spike

5.2 UDP best block with NAK set

UDP best block with NAK set 은 UDP full block with ACK 의 문제점인 크기를 알려주는 필드에 오류가 발생하면 오류가 발생한 시점을 찾을 수 없다는 것을 보완한 것으로, 이런 오류를 방지하기위해 1500bytes 를 가득 채우는 것이 아니라 1500bytes 에 근접하게 채워 UDP 데이터그램을 전송하게 설계했다. 즉 온전한 데이터만이 블럭에 포함되게 했는데, 이 방법은 전체 UDP 데이터그램의 크기를 알 수 있고 블록 하나에 포함된 각각의 데이터에 대한 크기를 비교할 수 있으므로 우연히 실제 데이터 부분과 일치하는 경우만 제외하면, 오류가 발생한 것을 즉시 알아챌 수 있다.



[그림 5] TCP with ACK 의 프로토콜

[그림 5]에서 보는 것과 같이 data_2 는 data_2 의 size 필드에서 명시하는 온전한 크기 그대로를 갖게 했다. 반면 1500bytes 전체를 다 사용할 수 없다는 단점을 가지고있다. 그리고 패킷 손실이 burst 하게 발생하는 경우가 많아서 NAK 을 하나씩 보내지 않고, 연속된 NAK 에 대해서는 시작된 패킷의 일련번호와 개수를 보내서 piggy-backing 을 했다. NAK 을 TCP 로 전송하면 NAK 이 늦게 도착하는 현상이 발생해 UDP 로 전송하는 대신 서버측에서 NAK 을 받지 못할 가능성이 있기 때문에 해당 packet 이 재전송되지 않으면 클라이언트는 $RTT \times 1.5$ 의 간격으로 NAK 을 재전송한다.

이 프로토콜에 의한 성능향상은 $\frac{TCP}{UDP} \approx 2.24$ 의 뚜렷한 성능향상이 있어서 무선 망을 위한 다중 파일 전송 프로토콜에 가장 적합한 것으로 증명 됐다.

6. 결론

본 연구논문은 실험 결과를 토대로 휴대폰에서 여러 파일을 동시에 전송하는 시뮬레이션 때 설계한 프로토콜들을 소개했고, 무선 망을 위한 다중 파일 전송 프로토콜에 가장 좋은 성능을 보인 것이 ' UDP best block with NAK set' 라 결론지었다. NAK 으로 packet 의 전송 여부를 판단하고 이더넷 MTU 에 가장 근접한 size 를 갖는 data 를 전송했을 때,

평균 $\frac{TCP}{UDP} \approx 2.24$ 의 높은 성능 향상 을 보였다.

높은 성능향상의 원인은 서버측에서 일정간격 지속적으로 패킷을 전송해 네트워크의 사용 효율을 극대화했고, 손실된 패킷에 대해서는 NAK set 을 사용해 NAK 을 동시에 전송한 것과 한번에 전송하는 패킷의 크기를 MTU 크기에 가깝게 유지했기 때문이다.

다른 실험군으로, 최상의 환경인 휴대폰이 100,000 Bytes RAM 의 충분한 저장공간이 있다고 가정하고 buffer 를 사용하지 않고 비순차적으로 저장하면,(다른 환경은 UDP best block with NAK set 에서와 같다.)

$\frac{TCP}{UDP} \approx 2.56$ 의 성능향상 비율을 보인다.

이 두 결과로 보아 버퍼링보다는, 패킷전송을 MTU 에 가까운 크기로 전송하고 NAK set 을 하는 것이 더욱 성능 향상에 기여하는 것이라고 판단할 수 있다.

참고 문헌

[1] D. Tang and M. Baker, " Analysis of a Local-Area Wireless Network," Proceedings of the sixth annual international conference on Mobile computing and networking, August 6 - 11, p.1-10, 2000.

[2] D. Tang and M. Baker, " Analysis of a Metropolitan Area Wireless Network," Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking, August 15-19, p.13-23,1999