HTTP에서 서버 자원의 효율적인 이용를 위한 연결 관리 연구

이윤정*, 김태윤*
*고려대학교 컴퓨터학과
e-mail:genuine@netlab.korea.ac.kr

A Study on Connection Management for Efficient Use of Server's Resource in HTTP

Yoon-Jung Rhee*, Tai-Yun Kim*
*Dept of Computer Science & Engineering,
Korea University

요약

현재 사용하고 있는 HTTP/1.0은 각각의 트랜잭션마다 별개의 TCP 연결 설정을 해야함으로서 야기되는 잦은 RTT지연과 Slow Start로 인해 성능상의 문제점을 야기하고 있다. IETF의 HTTP-NG은 이런 문제점을 포함하여 다각적인 요구를 수용할 수 있도록 HTTP/1.1(RFC 2616)을 발표하였다. HTTP/1.1은 이 문제점을 지속적인 연결(Persistent Connection) 개념을 도입하여 개선하고 있다. 그러나, 연결 해제 시점을 명확하게 정의하고 있지 않기 때문에 이로 인해 서버의 자원 낭비를 줄이는 효율적인 연결 관리 방법을 제시하지 못하고 있다. 본 논문은 하나의 TCP 연결 위에 다중의 요구들을 실행할 수 있는 HTTP/1.1의 지속적인 연결 개념에, 클라이언트 측에서 적절한 TCP 연결 해제 시점을 지원하여 서버 자원을 효율적으로 사용할 수 있는 알고리즘을 제안한다.

1. 서론

HTTP 프로토콜은 WWW에서 HyperText Markup Language(HTML)문서를 송수신하기 위해 사용하고 있는 애플리케이션 프로토콜로서 TCP를 수송 계층 프로토콜로 이용해서 이루어지는 애플리케이션 계층 프로토콜 가운데 하나이다.

오늘날, 전형적인 WWW 페이지는 HTML문서와 많은 삽입 이미지들을 포함하고 있다. HTML 한 페이지에 20개 이상의 삽입 이미지를 갖는 것이 보통일이다. 웹 상에서 이 이미지들 각각은 독립적이며, 개별적으로 검색되고 바뀐다. 따라서, 웹 클라이언트는 보통 기본 HTML문서를 전송한 후, 그 안에 삽입되어 있는 이미지들을 불러온다. WWW의 HTML 문서는 HTTP 프로토콜을 통해서 전송되는데, HTML문서와 그 안의 삽입 이미지들은 전형적으로 동일한 서버에 위치한다.

삽입되는 개체 수의 증가는 HTTP/1.0[1] 프로토콜이 디자인되었던 환경으로부터 많은 변화가 있었음을 나타낸다. 그 결과, HTTP/1.0은 동일한 서버로부터 각각의 개체에 대하여 개별적인 TCP 연결을 생성하기 때문에, 다중의 요구를 비효율적으로 처리한다. HTTP/1.0에서는 각각의 문서마다 별도의 연결을 만들어야 하는 추가적인 부담이 발생한다. 따라서 같은 장소에서 많은 문서를 갖고 오려고 할 때 HTTP 프로토콜은 성능상의 저하를 발생시키게 된다. IETF의 HTTP-NG은 이런 문제점을 포함하여 다각적인 요구를 수용할 수 있도록 HTTP/1.1(RFC 2616)[2]을 발표하였다. HTTP/1.1은 이 문제점을 지속적인 연결(Persistent Connection) 개념을 도입하여 개선하고 있다. HTTP/1.1 표준은 하나의 연결 위로 여러 개체를 전송함으로써 이 문제점을 해결할 수 있게 디자인되었다. 동시에 WWW 컨텐츠에서 기대되는 변화는 삽입 개체의 수가 줄어드는 것이다. 이렇게 되면 네트웍 성능을 증가시킬 수 있을 것이다. 그러나, 연결 해제 시점을 명확하게 정의하고 있지 않기 때문에 이로 인해 서버의 자원 낭비를 줄이는 효율적인 연결

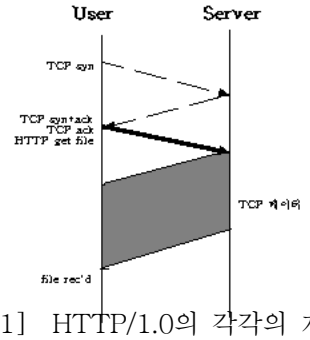
관리 방법을 제시하지 못하고 있다.

본 논문은 하나의 TCP 연결 위에 다중의 요구들을 실행할 수 있는 HTTP/1.1의 지속적인 연결 개념에, 클라이언트 측에서 적절한 TCP 연결 해제 시점을 지원하여 서버 자원을 효율적으로 사용할 수 있는 알고리즘을 제안한다. 본 논문의 2장에서는 HTTP/1.0의 문제점과 HTTP/1.1에서의 개선된 내용을 다루고, 3장에서는 본 논문에서 제안하고 있는 CM-HTTP에 대하여 기술하며, 4장에서는 결론 및 향후 과제에 대하여 기술한다.

2. HTTP/1.0의 문제점

2.1 잦은 RTT지연

HTTP/1.0에서 클라이언트는 각각의 개체에 대한 요구를 위해 서버와 파일 전송에 대해 개별적인 TCP 연결을 맺고, 각기 다른 연결로 파일을 전송한다. 각각의 연결 설정을 위해서는 파일 채널 설정을 위한 각각의 RTT(round-trip time)가 요구된다.



[그림 1] HTTP/1.0의 각각의 개체 전송

[그림 2]에서 보는 바와 같이 채널 설정 시에 필요한 RTT가 각각의 request의 TCP 연결 시에 매번 요구된다.

또한 TCP는 연결이 설정된 뒤에야 초기 요구 메시지가 전달되기 때문에, 서버의 시각에서 볼 때, 요구 메시지가 어플리케이션까지 전달되기 위해서는 1.5 RTT가 소요된다.[3][4]

2.2 Slow Start로 인한 문제점

TCP는 Sliding Window 프로토콜을 사용하여 각 사용자 자신이 전송한 각각의 패킷에 대한 전송 확인(ACK)이 도착하기 전에 window 크기만큼의 패킷을 계속 전송할 수 있도록 하고 있다[4]. 만일, 연결 네트워크에 혼잡(congestion)이 탐지된다면 window 크기를 다시 산출하는 Slow Start 알고리즘이 실행된다. 이 알고리즘은 연결이 초기화되었을 때, 패킷을 잃었을 때, 연결에 심각한 휴지기간(idle period)이 생겼을 때 실행된다.

HTTP에서 요구되는 데이터의 길이는 FTP 등에서 요구하는 길이보다 비교적 짧다. HTTP헤더가 때로는 뒤에 이어지는 데이터보다 클 때도 있다. TCP는 스트림 데이터를 작은 세그먼트로 쪼개어 전송한다. 세그먼트의 크기는 네트워크에 따른 MSS(maximum segment size)에 달려있지만, MSS의 디폴트 값은 536바이트이다. HTTP 헤더는 MSS보다 더 길기 때문에 클라이언트 TCP는 이것을 두 개 이상의 세그먼트로 쪼갤 필요가 있다. 연결 초기에는 congestion window가 1로 초기화되어 있기 때문에, 두 번째 세그먼트를 보내기 전에 첫 번째 세그먼트가 확인될 때까지 기다려야하는데, 이 때 Slow Start 알고리즘이 실행되므로 추가적인 지연이 발생하며 각각의 요구에 대해 매번 이 과정을 실행해야 한다.[3][4]

3. HTTP/1.1 표준

3.1 HTTP/1.1의 개선 내용

기존에 사용되고 있는 HTTP프로토콜은 HTTP/1.0을 구현한 것이다. 현재, IETF의 HTTP-WG에서는 HTTP/1.0에서 야기된 앞서의 문제점들과 그 밖의 다양한 문제점들을 해결하기 위한 HTTP/1.1 표준을 발표했으며 추가작업과 검증작업이 진행중이다. HTTP/1.1은 전 버전인 HTTP/1.0의 많은 부분을 개선하려는 노력을 하고 있다. 개선사항은 크게 다음과 같다.

* Request methods
  - Safe & Idempotent, OPTIONS, PUT, HEAD, DELETE, POST, TRACE, CONNECT 등
* Persistent connection
* Request Pipelining
* Chunked encoding
* Access authentication
* Byte range operations
* Content negotiation
* Digest Authentication
* Caching in HTTP

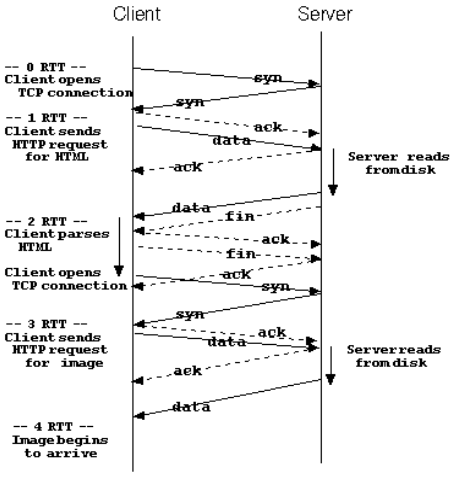
3.2 HTTP/1.1의 지속적인 연결

HTTP/1.1은 위에서 살펴보았던 문제점들을 해결하기 위해 지속적인 연결(Persistent Connection)을 포함하고 있다[2].

[그림 4]와 [그림 5]은 각각 여러 개의 이미지가 삽입된 HTML문서를 교환할 때의 HTTP/1.0와 HTTP/1.1 모델에서의 클라이언트와 서버의 상호작용을 도시하고 있다. 클라이언트로부터 시작하여 서버를 거쳐 다시 클라이언트로 향하는 하나의 왕복 실선은 하나의 RTT를 나타낸다. 점선의 대각선은 TCP 프로

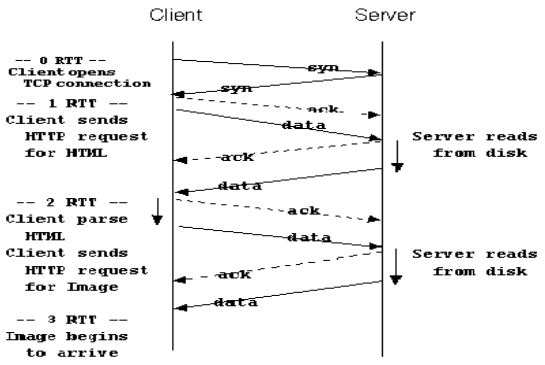
토콜에 의해서 요구되는 패킷이며 직접적으로 지연에 영향을 미치지는 않는다.

HTTP/1.0에서는 삽입된 첫 번째 이미지가 전송될 때까지 4번의 RTT를 요구한다. 사용자가 요구하는 문서에 대하여 TCP 연결 설정을 위한 한 번의 RTT와 HTML문서를 위한 하나의 RTT를 합쳐 두 번의 RTT를 요구한다. 문서 안에 이미지 파일이 삽입되어 있다면 해당 이미지 파일에 대하여도 각각 2번씩의 RTT를 요구한다.



[그림4] HTTP1/1.0 클라이언트/서버 상호작용

HTTP/1.1에서는 HTML 문서의 전송에 두 번의 RTT를 요구한다. TCP 연결 설정을 위한 것과 문서의 요구/응답을 위한 각각 한 번씩의 RTT이다. 현재의 TCP 연결을 계속 사용하기 때문에 삽입된 첫 번째 이미지부터는 해당 파일의 전송에 대한 요구/응답의 RTT 1번씩만 사용하게 된다. 따라서 클라이언트의 HTTP 요구에 대한 RTT를 각 문서 당 최대 한번으로 줄일 수 있다.



[그림 5] HTTP/1.1 클라이언트/서버 상호작용

이 모델은 HTTP/1.0서 생기는 잦은 RTT로 인한 문제점을 해결할 수 있다. 이를 위해 HTTP/1.1 모델에서는 HTML문서 안에 여러 개의 파일 요구가 포함되어 있을 때 클라이언트가 서버에게 문서를 처음 요구를 할 때 사용한 TCP 연결을 다음 요구 때에도 계속 사용한다.[5][6]

3. 연결 관리를 위한 CM-HTTP 프로토콜

3.1 연결 해제 시점 정의

인터넷에는 수많은 사용자와 서버, 많은 양의 트래픽이 존재하며 서버들은 수많은 클라이언트와 통신하게 된다. 수많은 클라이언트들로부터 연결 설정과 문서요청이 끊임없이 들어오고 있는 서버 측에서는 연결은 되어 있지만 더 이상의 데이터 이동이 없는 클라이언트와 '휴지상태(idle)'의 연결을 유지하게 위해 자원을 허용하는 것은 자원 사용의 효율성을 떨어뜨리며, 새로 연결을 설정하려는 클라이언트와의 연결 속도에도 커다란 영향을 미치게 된다[8]. 본 논문에서는 이 문제점을 해결하기 위하여 여러 개의 멀티미디어 개체가 삽입된 HTML문서의 전송을 하나의 트랜잭션으로 하여 트랜잭션이 종료되는 시점을 클라이언트와 서버와의 TCP 연결 해제 시점으로 정의하는 Connection Management HTTP(이하CM-HTTP) 프로토콜을 제안한다. 문서 안에 포함되어 있는 여러 개의 파일 중에서 마지막 파일의 전송이 끝났을 때 클라이언트는 서버 측에 연결 해제를 알리는 메시지를 보내고 서버와의 TCP 연결을 종료시킨다. 사용되는 method는 HTTP/1.1에서의 파일 전송을 위한 "GET"과, 클라이언트가 서버에게 TCP 연결 해제를 알리는 "CLOSE"로 제한하였다.

3.2 CM-HTTP 서버 알고리즘

CM-HTTP서버는 CM-HTTP 클라이언트의 요청에 따라 TCP 연결을 설정하고 클라이언트가 요구하는 문서를 찾아 전송하며 마지막 트랜잭션 이후 일정시간이 흐르면 클라이언트와의 연결을 해제하는 동작을 취한다.

```
begin
    establish server.socket
    do
        if accept "SYN" from client
            then open client.socket
        do
            read stream
            if method in stream is "GET"
                begin
                    get filename from stream
```

```
                if file exist then response
                    requested file
                else response "file not found"
              end if
          else if method in stream is "CLOSE"
              then close client socket and break
        while (true)
    while (true)
end
```

[그림 6] CM-HTTP 서버 알고리즘

3.3 CM-HTTP 클라이언트 알고리즘

CM-HTTP 클라이언트는 사용자의 URL 요구에 따라 해당 서버와의 TCP 연결을 설정하고 서버에게 문서 요구를 한다. 해당 문서를 전송 받은 후 TCP 연결을 해제하지 않고 다음의 문서요구가 현재의 서버일 경우에는 기존의 TCP 연결을 계속 사용하지만, 새로운 서버로의 트랜잭션이 요구되면 현재의 서버에게 “CLOSE” 메시지를 보내고 TCP 연결을 해제한 뒤 새로운 서버와의 연결을 설정한다.

```
begin
   do
     get new URL
     open server.socket with new URL
     send "GET" method and URL.filename
     read stream from server
     display stream content
     search included other filename in received file
     while (other request file remain)
        begin
          send "GET" method and filename
          read stream from server
        end while
     send "CLOSE" method to server
     close server.socket
  while (true)
end
```

[그림 7] P-HTTP 클라이언트 알고리즘

3. 결론 및 향후과제

HTTP/1.1은 지속적인 연결 개념을 도입하여 HTTP/1.0의 문제점인 각각의 트랜잭션마다 별개의 TCP 연결 설정으로 인해 야기되는 잦은 RTT 지연과 Slow Start로 인한 성능상의 문제점을 개선하고 있다. 그러나, 연결 해제 시점을 명확하게 정의하고 있지 않고 있다.

본 논문에서는 HTTP/1.1에서의 하나의 TCP 연결 위에 다중의 요구들을 실행하기 위한 지속적인 연결 개념에, 적절한 TCP 연결 해제 시점을 클라이언트 측에서 지원할 수 있는 알고리즘을 제안하였다. 클라이언트에서 사용자에 의해 행해지는 한번의 요구를 하나의 트랜잭션으로 정의하며, 이런 하나의 트랜잭션을 하나의 TCP 연결 위에서 처리하여, HTML 문서와 그 안에 삽입되어 있는 이미지 등의 파일들의 요구를 수행하고 마지막 파일의 전송이 종료되었을 때를 TCP 연결 해제 시점으로 한다. 본 논문의 알고리즘은 HTTP/1.1의 지속적인 연결에 서버 자원을 더욱 효율적으로 사용하고 관리할 수 있는 메커니즘을 제공한다.

현재 본 논문에서 제안하는 프로토콜을 바탕으로 프로토타입을 구현중이며 성능분석과 클라이언트의 요구에 대한 파이프라인화(pipelining)의 추가 작업이 연구될 것이다.

참고문헌

[1] Hypertext Transfer Protocol (HTTP/1.0) - RFC 1945, Tim berners-Lee, R.Fielding, H.Frystyk. May. 1996

[2] HTTP/1.1- RFC 2616, T.Berners-Lee, R.Fielding. H.Frystyk Nielsen. Jun. 1999

[3] Analysis of HTTP Performance Problems, Simon Spero, http://sunsite.unc.edu/mdma-releas/http-prob.html

[4] Analysis of HTTP Performance, Touch,J., J.Heidemann, K.Obraczka, USC/Information Sciences Institute, June, 1996

[5] Improving HTTP Latency, Jeffrey C. Mogul, Padmanabhan.V, In Proc. 2nd International WWW Conf. '94: Mosaic and the Web. October. 1994

[6] Performance Interactions Between P-HTTP and TCP Implimentation, Heidemann,J. ATM Computer Communication Review, April 1997

[7] Internetworking with TCP/IP(Vol I), Douglas E. Comer Prentice-Hall. 1989

[8] HTTP Connection Management, Gettys,J., Freier,A. March 26. 1997