효율적인 웹문서 처리를 위한 HTTP 지연 개선에 관한 연구

A Study on Improving HTTP Latency for the Efficient Web Document Processing

고일석
충북과학대학 전자상거래과 교수

최우진
동원대학 e-비지니스과 교수

나윤지
대전보건대학 초빙교수

류승렬
(주)아심정보기술 CEO

Il-Seok Ko
Professor, Dept. of Electronic Commerce, Chungbuk Provincial Univ.

Woo-Jin Choi
Professor, Dept. of e-business, DongWon College

Yun-Ji Na
Invited Professor, Dept. of Computer Engineering, Daejeon Health College

Seung-Ryul Ryu
CEO, Asim Information Tech.

중심어 : HTTP Performance, Web Document, Latency, P-HTTP, GETALL, GETLIST

요약

인터넷 사용의 증가는 네트워크 부하를 급격히 증가시키고 있으며 이러한 네트워크 부하의 증가는 인터넷 사용자들의 응답속도를 늦추는 요인이 되고 있어 이를 해결하기 위한 각종 연구가 필요하다. P-HTTP는 파이프라인 구조를 통해 HTTP의 성능을 개선한 모델이나 TCP 기반의 동작에서 P-HTTP와 TCP의 상호작용에서 발생하는 문제점으로 인해 성능이 저하된다. 본 연구에서는 웹문서의 효율적인 처리와 인터넷 응답 속도의 저하를 막기 위해 P-HTTP와 TCP의 상호 작용으로 인해 발생하는 문제점을 분석하고 이를 개선한 모델을 제시하였다. 본 논문에서 제시한 모델은 기존의 파이프라인 방식의 장점과 GETALL 메쏘드가 가지는 장점을 유지하면서 클라이언트 측의 캐시의 효용성을 높일 수 있어 각종 이미지와 멀티미디어 자료를 포함한 웹문서의 처리에서 그 효용성이 있을 것이다.

Abstract

Recently, network overload is greatly increased with explosive use of internet. So the Hyper-Text Transfer Protocol(HTTP) is required improve of performance for decreasing of latency on the web document processing. The P-HTTP is one of the improved model of the HTTP and has pipeline structure, but performance of the P-HTTP is decreased on interaction between the TCP and P-HTTP. Modification of structural design of the HTTP is not enough to improvement this problem. In this paper, we analyse performance of the HTTP and P-HTTP, and propose a new method on improving HTTP latency for the efficient web document processing.

Ⅰ. 서론

전자상거래를 비롯한 각종 웹 기반 서비스는 국내외를 막론하고 이미 활발하게 진행되고 있으며 그 거래 규모 또한 계속 증가하고 있다. 이와 같은 웹 사용의 급격한 수요 증가는 네트워크 부하의 급속한 증가를 가져왔고 이러한 네트워크 부하의 증가는 웹의 기본 프로토콜인 HTTP(Hypertext Transfer Protocol)의 성능 개선을 위한 각종 노력을 필요로 하게 되었다[1]. HTTP는 TCP(Transmission Control Protocol)를 기반으로 하고 있어 TCP가 가지는 문제점을 웹에서도 가지게 된다. 또한 HTTP의 구조적 문제점은 웹 성능을 저해하는 요소이며 이런 HTTP의 결점을 보완하여 네트워크의 부하를 줄이기 위한 연구가 활발히 진행되었다. 그 결과로 파이프라인 방식의 P-HTTP[2],[3]와 기존의 프로토콜을 확장하여 성능을 개선하기 위한 방안으로 PEP(Protocol Extension Protocol)[4] 등

이 제안되었다. 그러나 HTTP의 구조적 설계의 수정으로 HTTP와 TCP의 상호작용(interaction)에서 발생하는 문제점들을 모두 해결할 수 없다. 이에 따라 HTTP와 TCP의 상호작용으로 인한 성능저하를 줄이려는 연구 또한 활발히 진행되었다[5].

본 연구에서는 P-HTTP와 TCP 사이의 상호 작용으로 인해 발생하는 문제점을 분석하고 이를 개선한 방법을 제시하고 있다. 본 연구에서 제시한 모델은 Padmanabhan과 Mogul[2]에 의해 제시된 GETALL과 GETLIST 메쏘드를 사용하여 기존의 P-HTTP가 GETALL을 계속해서 요구하는 파이프라인 방식 프로토콜의 문제점을 개선한 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 통해 TCP와 HTTP를 분석하였고 3장에서는 기존의 모델을 개선한 모델을 제시하였고 4장에서는 결론을 살펴보았다.

II. 관련 연구

1. GETALL과 GETLIST 메쏘드

Padmanabhan과 Mogul은 GETALL과 GETLIST 메쏘드를 제안하였다[2]. 이 두 가지 메쏘드의 동작 방식은 다음과 같다.

1.1. GETALL 메쏘드

GETALL 메쏘드는 클라이언트가 요청한 문서와 그 문서에 포함된 이미지를 한꺼번에 가져오는 방식이다. GETALL 요구를 받은 서버는 HTML 문서를 파싱하여 내부 이미지들의 URL을 찾아내고 HTML 문서와 찾아낸 이미지를 하나의 응답으로 보낸다. 이 경우 서버 측의 HTML 문서 파싱으로 인한 로드는 연속적인 HTTP 요구를 처리하는 것에 비해 비교적 부담이 적다. 또한 서버는 특정 HTML 파일과 연관된 URL에 대한 캐시를 두어 반복되는 동일한 요구에 걸리는 파싱의 부담을 줄일 수 있다. 그러나 GETALL 메쏘드는 클라이언트에가 최근 가져온 정보에 대한 캐시를 가지고 있으나 서버는 문서에 연결된 내부 이미지가 클라이언트 측의 캐시에 있는지를 알 수 있는 방법이 없다. 따라서 GETALL 메쏘드에서는 서버가 HTML 문서에 포함된 모든 이미지를 한꺼번에 보내주므로 클라이언트 측 캐시의 효과를 활용할 수 없게 한다. 그림 1은 3개의 이미지를 포함한 HTML문서에서 GETALL 메쏘드를 나타낸 것이다.

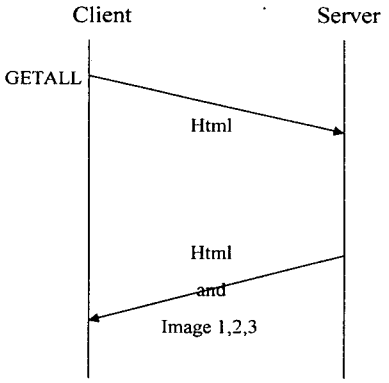

그림 1. 3개의 이미지를 포함한 HTML문서에서 GETALL

1.2. GETLIST 메쏘드

GETLIST 메쏘드는 GET으로 가져온 문서를 클라이언트가 파싱(parsing)하여 필요한 이미지들만 서버에 요구하게 된다. GETLIST 메쏘드는 클라이언트 캐시의 효율성을 그대로 살리면서도 서버의 문서나 이미지들을 한꺼번에 요청하기 위해 효율적이다. 클라이언트는 GET을 이용하여 HTML 파일을 가져오고 GETLIST를 이용하여 자신의 캐시에 있지 않는 이미지 모두를 하나의 메시지로 요청한다. GETLIST는 결과적으로 이전의 GET이 끝나기를 기다리지 않고 연속해서 GET을 보내는 것과 같아서 논리적으로 파이프라인의 효과를 거둘 수 있다. 클라이언트가 URL에 명시된 HTML 문서를 가져올 때 서버는 단순히 그 파일의 내용을 전달한다. 그러면 클라이언트는 내부 이미지에 대한 요구를 서버로 보낸다. 거의 모든 내부 이미지는 HTML 문서와 같은 서버에 존재하므로 서버가 HTML 문서와 같은 서버에 존재하는 내부 이미지를 모두 보낸다면 이미지 요구에 필요한 시간을 줄일 수 있다. 따라서 GETLIST 메쏘드는 파이프라인이 가지는 문제를 해결하면서도 성능을 향상시킬 수 있다. 그림 2는 3개의 이미지를 포함한 HTML문서에서 GETLIST 메쏘드를 나타낸 것이다.

그림 1과 2와 같이 3개의 이미지를 가지는 HTML 문서를 가져올 때 GETALL과 GETLIST의 동작에서 캐시 문제를 고려하지 않고 서버쪽 파싱과 클라이언트 쪽 파싱에 걸리는 시간이 동일하다고 보았을 때 GETALL이 GETLIST 보다 더 나은 성능을 보인다. 따라서 본 연구에서는 파이프라인 방식의 P-HTTP에 대해 GETALL과 GETLIST 메쏘드를 활용하여 성능을 개선하고 있다.

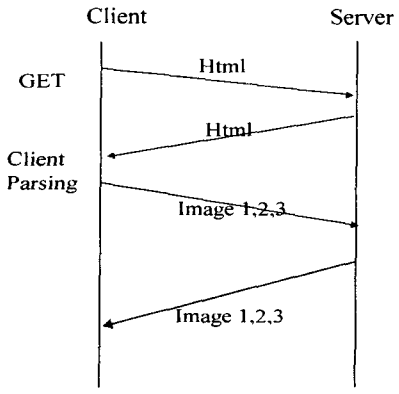

그림 2. 3개의 이미지를 포함한 HTML문서에서 GETLIST

2. HTTP와 TCP

지금까지 HTTP[10]의 개선과 TCP와의 상호 작용 문제를 해결하기 위해 다양한 연구가 있었으며 이들 연구들은 웹 서버의 부하를 측정하여 네트워크 성능을 평가하는 작업들과 함께 진행되었다. 웹의 동작은 기본적으로는 크기가 작고 수많은 요구에 대해 단방향(uni-direction)성의 요구-응답 방식이라 할 수 있다. 대부분의 웹 문서는 HTML 문서와 이들에서 사용되는 내부 및 외부의 이미지들로 구성되고 있다. 따라서 사용자가 요구한 웹 문서를 클라이언트 측의 웹브라우저로 완전히 가져오기 위해서는 HTML 문서 자체와 이에 포함된 내부 및 외부의 이미지 각각에 대한 요구가 필요하다. 이러한 HTTP의 동작으로 인해 네트워크의 부하는 요구 시점에 집중(bursty)되게 된다.

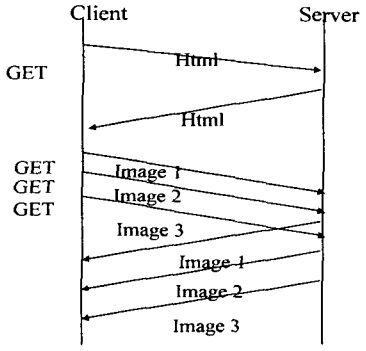

그림 3. 파이프라인 방식을 이용한 P-HTTP

또한 대부분의 사용자들은 빈번하게 사이트를 이동하는 경향이 있으며 이로 인해 웹은 작고 빈번한 요구-응답의 특성을 갖게 된다. 웹의 이러한 작고 빈번한 요구-응답의 특성으로 인해 HTTP가 TCP와의 상호 작용에서 다음과 같은 세 가지의 문제점을 가지게 된다[1].

- 첫째, 빈번한 연결 설정과 해제 과정은 많은 연결을 TIME-WAIT 상태로 두게되므로 서버에게 부담을 가중시킨다.
- 둘째, 짧은 연결 시간은 TCP의 혼잡회피를 위한 slow-start 알고리즘의 성능을 저하시킨다.
- 셋째, TCP 연결을 설정할 때의 three-way handshake는 대기 시간(latency)을 증가시킨다.

위에서 열거된 문제점들의 해결을 위한 연구로는 상시 연결(Persistent connection) HTTP(P-HTTP)[2], Transaction TCP[6], UDP 기반 요구 응답 프로토콜인 Asynchronous Reliable Delivery Protocol(ARDP)[7] 등이 있다. 그림 3은 문서를 가져올 때 파이프라인 방식을 나타낸 것이다.

P-HTTP는 TCP 연결 설정 때의 오버헤드를 경감시키기 위해 여러 개의 HTTP 요구들을 보내는 동안 하나의 TCP 연결을 유지하는 방식이다. P-HTTP는 클라이언트가 서버에 접근할 때 연결을 유지하는 시간이 그다지 짧지 않다면 첫 연결 이후의 연결 설정 시간을 줄여서 이전의 HTTP보다 훨씬 나은 성능을 보일 수 있다. 그러나 단순히 연결을 유지하는 것은 특정 TCP와 HTTP 서버에서 P-HTTP와 TCP와의 상호 작용으로 연결을 설정하고 해제할 때 오히려 성능을 저하시키는 문제점을 발생시켰다[5].

클라이언트는 이전의 요구에 대한 ACK (acknowledgment)를 받은 뒤에만 다음 요구를 보내는 정지-대기(stop-wait)방식으로 서버와 동작하기 때문에 상시 연결에도 불구하고 HTTP는 내부 이미지들(in-lined images)을 가져올 각각에 대해 하나의 RTT(Round Trip Time)가 필요하다. 하지만 다음 이미지를 가져오는 것은 이전 이미지의 수신 여부와는 상관이 없다. P-HTTP에서는 이전 요구 메시지에 대해 서버로부터 응답을 받지 않는 상태임에도 연속적으로 GET 메시지를 보낼 수 있는 파이프라인(pipeline)을 지원하다. 이때 서버는 요구 메시지를 수신한 순서대로 응답 메시지를 전달해야 한다. 하지만 HTTP에 비해 상당한 성능 향상을 거두는 P-HTTP의 파이프라인에도 TCP와의 상호작용으로 인한 문제점이 있다[8].

● 먼저 버퍼에 있는 HTTP 요구들을 배출(flush)하기 위한 메커니즘이 필요하다. 여러 개의 HTTP 요구들은 하나의 같은 TCP 세그먼트(Segment)로 보낼 수 있을 때까지 버퍼에 대기한다. 따라서 전송하는데 필요한 패킷의 수, 그리고 클라이언트와 서버 양측의 패킷처리 시간은 줄어들지만 이것은 요구들이 즉시 보내지지 않는다는 것을 의미한다. 그러므로 이 버퍼의 데이터가 일정한 크기에 이르렀거나 일정한 시간이 흐른 뒤에 데이터가 TCP 세그먼트의 크기보다 작더라도 강제로 배출하는 메커니즘이 필요하다.

● 다음으로 TCP의 구현에서 Nagle 알고리즘을 사용할 때 충돌 문제이다[9]. Nagle 알고리즘은 이미 전송한 패킷이 있을 때 이 패킷에 대한 ACK가 도착하기 전까지는 다른 패킷을 전송하지 않고 버퍼에 대기시켜서 작은 TCP 세그먼트의 전송을 지연함으로써 세그먼트의 수를 줄이는 방법이다. 파이프라인은 연속적으로 HTTP요구 메시지를 전달해야 하지만 Nagle 알고리즘에 의해 즉시 전송되지 못하고 대기해야 하므로 Nagle 알고리즘은 성능 향상을 저하시키는 요인이 된다. 그러므로 파이프라인의 성능향상을 위해서는 HTTP 요구-응답 때에는 클라이언트와 서버 양측에서 이 알고리즘을 사용하지 않고 tcp_nodelay 옵션을 설정, 패킷을 전송할 수 있도록 하여야 한다. 그러나 이것은 상위 프로토콜인 HTTP를 위해서 하위 프로토콜인 TCP의 옵션을 수정해야 한다는 번거로움을 가져온다.

III. 제안기법

제안하는 모델은 그림 4와 그림 6과 같다. 이것은 GETLIST, GETALL 메쏘드를 사용하여 기존의 P-HTTP와 TCP의 상호 작용에서 발생하는 문제점을 개선하였다.

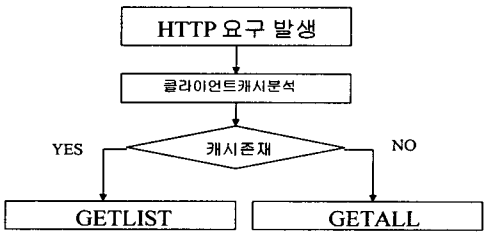

그림 4. 처음 접속일 경우

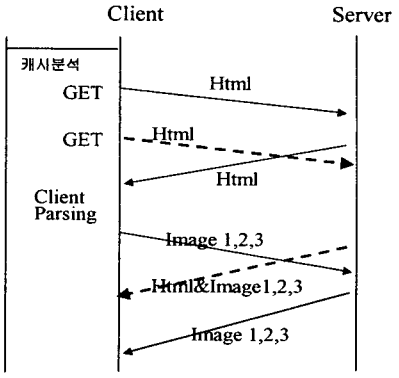

그림 5. 캐시분석을 통한 GETALL/GETLIST

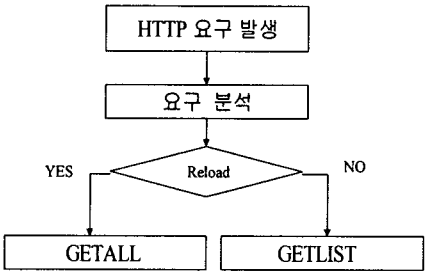

그림 6. Reload일 경우

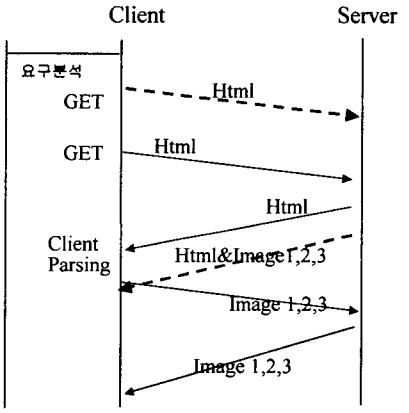

그림 7. Reload인 경우 GETALL/GETLIST

제안한 기법은 먼저 HTTP 요구의 발생에 대해서 클라이언트 측에서 먼저 이 요구를 분석하게 된다. 분석한 요구가

해당 URL에 대한 처음 접속일 경우 그림 3과 같이 GETALL 메쏘드를 선택하고, 아니라면 GETLIST 메쏘드를 선택한다. 마찬가지로 RELOAD 요구라면 GETALL 메쏘드를 선택하고, 아니라면 GETLIST 메쏘드를 선택한다. 처음 접속에 대한 판단은 클라이언트 측의 캐시의 확인을 통해 알 수 있다. 즉 클라이언트 측의 캐시에 요구한 웹문서가 존재하지 않을 경우에는 처음 접속이라 판단하게 된다. 앞에서 설명한 바와 같이 GETALL 메쏘드는 클라이언트 캐시의 효용성을 떨어뜨리게 된다. 따라서 GRTALL 메쏘드는 클라이언트 캐시와 상관없이 문서를 가져오는 경우 예를 들어 서버에 처음 접속해서 문서를 가져오거나 다시 읽기(RELOAD)와 같은 경우에만 유용하다. 또한 GETLIST 메쏘드는 논리적으로 이전의 GET이 끝나기를 기다리지 않고 연속해서 GET을 보내는 것과 같아서 파이프라인이 가지는 효과를 거둘 수 있다. 하지만 캐시 문제를 고려하지 않고 서버쪽 파싱과 클라이언트 쪽 파싱에 걸리는 시간이 동일하다고 보았을 때 GETALL이 GETLIST 보다 더 나은 성능을 보이게 된다. 제안 방법에서 GETLIST와 GETALL을 구현할 수 있는 방법을 살펴보자.

먼저 GETLIST를 구현할 수 있는 방법을 살펴보자. TCP의 MSS(MAXIMUM SEGMENT SIZE)는 이더넷에선 1460bytes, WAN에선 512bytes 또는 536bytes이다. 여러 개의 HTTP 요구가 하나가 됨으로 인해 GETLIST 메쏘드를 포함한 HTTP 헤더의 크기는 보통의 HTTP 헤더 크기인 200-300 bytes보다 커진다. 이 HTTP 헤더를 하나의 같은 TCP 세그먼트로 보낼 수 있을 때까지 버퍼에 대기하지 않고 즉시 TCP 완전(FULL) 세그먼트로 만들어 보낸다. HTTP 헤더를 TCP MSS 크기로 만들어 바로 요구-응답하는 것이다 이렇게 하면 출력버퍼를 배출하기 위한 메커니즘이 필요하지 않게 된다.

파이프라인의 경우 3개의 HTTP 요구가 출력버퍼에 잠시 대기한 후에 배출된다. 그러나 GETLIST의 경우 GETLIST 요구를 즉시 배출하므로 지연을 줄일 수 있어 성능의 효과를 높일 수 있다. GETLIST는 클라이언트나 서버에서 연속된 여러 개의 작을 패킷들을 보내는 게 아니라 하나의 요구-응답으로 보내는 것이다. GETLIST 메시지를 보내고 나서 ACK를 받을 때까지 새로이 메시지를 보낼 필요가 없으므로 GETLIST는 NAGLE 알고리즘에 의해 잠깐 지연하는 문제도 생기지 않는다. 따라서 클라이언트와 서버에서 Nagle 알고리즘을 사용하고도 GETLIST를 사용할 수 있다.

다음으로 GETALL 메쏘드는 GETLIST 메쏘드와 동작방식은 조금 다르지만 이 역시 GETLIST와 같은 방법으로 구현할 수 있다. GETALL을 보내는 클라이언트의 HTTP 헤더 크기는 기존의 GET 메쏘드를 포함한 HTTP 헤더 크기가 다를 바가 없다. 클라이언트는 GETALL 메쏘드를 포함하는 HTTP 헤더가 하나의 TCP 세그먼트 크기보다 훨씬 작아도 버퍼에 대기시키지 않고 바로 TCP 완전 세그먼트로 만들어 서버로 보낸다. GETALL을 보낸 클라이언트는 요청한 문서를 서버에서 파싱하여 내부 이미지들과 함께 모두 보내주기까지 기다릴 수밖에 없다. 따라서 GETALL 또한 Nagle 알고리즘과 문제를 일으키지 않는다. GETALL 요청을 받은 서버는 요구받은 문서를 파싱하고 난 후 GETLIST의 서버와 마찬가지 방식으로 동작한다.

IV. 결론

웹 사용의 폭발적인 증가는 네트워크 부하를 급격히 증가시켰다. 이러한 네트워크 부하의 증가로 인해 웹의 프로토콜인 HTTP(Hypertext Transfer Protocol)에 대한 개선이 필요하게 되었으며 이에 대한 많은 연구가 진행되었다. 이 중에서 HTTP의 성능을 개선한 P-HTTP의 파이프라인 구조는 웹의 TCP 기반의 동작에서 P-HTTP와 TCP의 상호작용에서 발생하는 문제점으로 인해 얘기치 않은 성능의 저하를 보인다. 이것을 개선하기 위해서는 HTTP의 구조적 설계의 수정만으로 HTTP와 TCP의 상호작용에서 발생하는 문제점들을 모두 해결할 수 없다. 본 논문에서는 P-HTTP와 TCP의 상호 작용으로 인해 발생하는 문제점을 분석하고 이를 개선한 모델을 제시하였다. 본 논문에서 제시한 모델은 Padmanabhan과 Mogul이 제안한 GETALL과 GETLIST를 이용하여 파이프라인의 장점을 가지면서도 TCP와의 상호작용을 해결할 수 있다. 제안한 모델은 GETALL이 가지는 장점을 유지하면서 클라이언트 측의 캐시의 효용성을 높일 수 있으며 기존의 P-HTTP의 장점을 활용할 수 있다. 또한 구현과정을 통해 TCP와 P-HTTP가 가지게 되는 상호작용으로 발생하는 문제점을 해결할 수 있는 방안을 제시하였다. 제안한 모델을 활용한다면 기존의 HTTP의 동작 속도를 개선할 수 있을 것이다.

참 고 문 헌

[1] Touch, J. Heidemann, K. Obraczka, "Analysis of HTTP Performance, " USC/Information Sciences

Institute, June, 1996.

[2] Padmanabhan, J. Mogul, "Improving HTTP Latency," Computer Networks and ISDN Systems, Vol. 28, pp.25~35, Dec 1995. Slightly Revised Version in Proceedings of the 2nd International WWW Conference '94; Mosaic and the Web Oct., 1994.

[3] Fielding R. J. Gettys, J.C Mogul, H. Frystyk, T. Berners-Lee, "RFC 2068- Hypertext Transfer Protocol -- P-HTTP," UC Irvine, Digital Equipment Corporation, MIT.

[4] H. Frystyk Nielsen, Dan Connolly, "PEP - an Extension Mechanism for HTTP"

[5] Heidemann, "Performance Interactions Between P-HTTP and TCP Implementation," ACM Computer Communication Review, 272, pp.65~73, April 1997.

[6] R. Braden, "T/TCP-TCP extensions for transactions functional specification," RFC 1644, Internet Request For Comments, July 1994.

[7] B. C. Neuman, The Virtual System Model: A Scalable Approach to Organizing Large Systems Ph.d. dissertation, University of Washington, 1992.

[8] Henrik Frystyk Nielsen, etc., "Network Performance Effects of P-HTTP CSS1, and PNG", W3C

[9] Nagle, J. "Congestion Control in IP/TCP Internetworks," RFC896, Ford Aerospace and Communications Corporation, January 1984.

[10] Berners-Lee Tim R. Fielding, H. Frystyk. "Informational RFC 1945 - Hypertext Transfer Protocol - HTTP/1.0" MIT/LCS, UC Irvine, May 1996.

고일석(Il-Seok Ko) 종신회원

1989년 : 경북대학교 전자계산 전공 (공학사)
1996년 : 경북대학교 컴퓨터공학과 (공학석사)
미)USIU College of Business Administration(MBA)
성균관대학교 경영대학원 앤소프전략스쿨 수료
연세대학교 컴퓨터산업시스템공학과 박사수료
현재 : 충북과학대학 전자상거래과 교수

최우진(Woo-Jin Choi) 정회원

상지대학교경영학과(경영학박사)
동국대학교경영학과(경영학석사)
동국대학교공업경영과(경영학사)
전)문경대학 사무자동과 조교수
현재 : 동원대학 e-비지니스과 부교수

나윤지(Yun-Ji Na) 정회원

1994년 : 경북대학교생명공학부 (이학사)
2001년 : 충북대학교 컴퓨터공학 (공학석사)
2001년 : 충북대학교 컴퓨터공학 (박사과정)
1999년 ~ 2000년 : 뉴욕공과대학(NYIT)대학원 Communication ART 전공(석사과정 2학기 수료)
현재 : 대전보건대학 컴퓨터정보처리과 초빙교수

류승렬(Seung-Ryul Ryu) 종신회원

1994년 2월 : 충북대학교 무역학과 (경영학학사)
2002년 8월 : 충북대학교 컴퓨터공학 (공학석사)
1999년 12월 ~ 현재 : (주)아심정보기술 CEO
2000년 3월 ~ 현재 : toolbook.net 운영자
2000년 3월 ~ 현재 : 주성대학 겸임교수