□특집□

차세대 웹을 위한 통신프로토콜

최 선 완†

◈ 목 차 ◈

1. 서 론

80년대 중반부터 90년대 초까지 인터넷 응용은 FTP, TELNET, EMAIL 등과 같은 고전적인 응용 서비스만을 제공하였고 인터넷 발전이 정체된 시기였다. 이에 따라 인터넷을 대체하기 위한 모델로서 국제표준화기구(ISO)의 OSI 참조모델이 전 세계적으로 각광받는 시기였다. 그러나 1993년 그래픽을 지원하는 웹 브라우저인 모자익(Mosaic)의 출현은 유럽에서 강력하게 추진했던 ISO/OSI 모델의 몰락과 함께 인터넷 역사를 새롭게 쓰는 계기가 되었다. 그러나 이 당시의 웹 서비스는 단순히 HTML로 작성된 웹 서버 정보를 브라우저를 통해서 검색하거나 CGI(Common Gateway Interface) 기반의 폼(form)을 처리하는 것이 대부분이었다. 그럼에도 불구하고 인터넷을 통해서 정보를 얻을 수 있다는 커다란 장점 때문에 93년부터 약 3년 동안은 인터넷 확산의 절정기였다. 물론 최근에도 많은 웹 서버가 구축되고 인터넷 사용자가 늘고 있지만 그 확산 속도는 어느 정도 안정화 단계에 들어가고 있다. 특히 향후 인터넷의 핵심은 XML 기반의 전자상거래가 웹의 대중화를 더욱 앞당길 것이다.

이러한 기술적이고 정치적인 측면에서 웹의 발전은 웹 서버와 웹 클라이언트 사이의 메시지를 전달하는 HTTP(HyperText Transfer Protocol) 또한 HTTP/0.9, HTTP/1.0, HTTP/1.1, 그리고 차세대 HTTP인 HTTP-NG로 발전되고 있다 [1][2].

1.1 HTTP/0.9

원래의 HTTP 버전으로 인터넷에서 적은 부담으로 파일을 불러오는 간단한 프로토콜이었다. 타입 정보, 내용 길이, 날짜, 메타 정보와 같은 복잡한 내용은 포함되지 않았다. 요청시 GET을 제외하고는 거의 사용되지 않았다.

1.2 HTTP/1.0 [3]

HTTP/0.9를 그레픽 브라우저에서 사용하기에는 적당하지 않았다. 즉, 데이터를 적당한 뷰어(외부 뷰어)로 보낼 수 있는 데이터의 타입을 알아야 했고 파일이 전송되면서 진행 상황을 디스플레이 하기 위한 내용 길이도 필요했고, 폼을 잘 사용하기 위한 방법도 필요했다. HTTP/1.0은 MIME (Multipurpose Mail Extension) [4]의 기능을 차용하여 웹을 위한 형태로 변형시켰다. HTTP/1.0은 현재 거의 사용하지 않지만 HTTP/0.9와의 호환성(backward compatibility)을 보장해야 했다. 이를 위해서 매 요청마다 새로운 TCP 연결을 이용한다. 즉, 특정 웹 페이지에 10개의 그림을 포함하고 있으면 10개의 TCP 접속을 설정한다. 웹의 발전 과

† 정회원 · 안양대학교 정보통신공학과 교수

정에서 몇가지 데이터와 타임 스탬프, 새로운 헤더(If-Modified-Since)가 추가되었다. 이것들은 캐싱 프록시 서버를 지원하기 위해서 더해진 것이다.

1.3 HTTP/1.1 [5][6]

HTTP/1.1은 HTTP/1.0의 다음 3가지의 심각한 문제를 해결하기 위해서 1996년부터 제안되고, 인터넷 표준인 RFC 2068 [5]을 거쳐서, 몇 번의 수정 후에 IETF의 드래프트 표준으로 제안되었다 [6].

1.3.1 IP 주소

HTTP/1.0에서는 어떤 회사가 특정 브랜드에 대해 다른 웹 주소를 필요로 할 때 각 사이트마다 새로운 IP 주소를 필요로 한다. 그러나 최근 IP 주소는 생각보다 빠른 속도로 고갈되어 가고 있다. HTTP/1.1에서는 다중 가상 사이트를 허용함으로써 여러개의 IP 주소없이도 별도의 웹 주소를 제공할 수 있다.

1.3.2 TCP 고려사항

초기의 HTTP/0.9와 HTTP/1.0은 데이터 전송 기능을 담당하는 TCP를 고려하지 않고 설계되었기 때문에 인터넷에 심각한 피해를 발생시키고 있다. HTTP/1.0에서 매 요청마다 새로운 TCP 접속을 설정하는 것은 HTTP 응용이 접속 설정을 위한 TCP 세그멘트를 전송하고 응답 세그멘트를 받는 시간인 RTT (Round Trip Time) 만큼 기다려야 한다. 특히 RTT 값이 상당히 큰 대륙간 통신, 위성 링크, 무선 링크에서는 성능을 저하하는 중요한 요인이다. 한편 TCP는 네트워크의 폭주(congestion)을 방지하기 위해서 슬로우-스타트(slow start) [8]를 사용한다. 슬로우-스타트는 접속을 설정한 후에 1개의 TCP 세그멘트를 보내고, 다음에 거의 두배씩 전송 세그멘트를 전송하는 방법이다. 슬로우-스타트에서는 ACK을 받은 수 만큼 전송 세그멘트를 증가하는데 TCP의 지연 응답(delayed acknowledgement)에 의해서 송신측이 보낸 모든 세그멘트에 대해 수신측이 항상 응답하지는 않으므로 항상 두배씩 증가하는 것은 아니다. 슬로우-스타트에서는 처음에 보내는 데이터의 양이 작기 때문에 대부분의 HTTP 연결이 종료될 때까지 최대 속도에 이르지 못한다. 또한 다음 HTTP 접속도 동일한 과정을 거치기 때문에 성능에 상당한 제한을 받는다.

넷스케이프는 HTTP/1.0의 이러한 문제를 해결하기 위해서 HTTP 프로토콜의 변경없이 브라우저의 응답 시간을 개선하기 위해서 한번에 하나의 TCP 접속을 설정하는 대신에 동시에 여러개의 TCP 접속을 설정하여 동시에 여러개의 아이콘으로 데이터를 디스플레이할 수 있다. 그러나 이 방법은 특정 클라이언트와 서버가 많은 TCP 접속을 독점함으로써 공평성 문제를 야기할 수 있다. 그럼에도 불구하고 다음 3가지 이유로 인해서 동시에 여러개의 TCP 접속 방법을 사용하고 있다.

(1) 다중 TCP 접속을 이용하는 클라이언트는 특정 페이지에 있는 객체의 메타 정보(예, 크기)를 조기에 알 수 있으므로 즉시 페이지를 포매팅할 수 있다.

(2) 특정 서버와 여러 TCP 접속을 동시에 설정하는 클라이언트는 한 개의 TCP 접속을 이용하는 클라이언트 보다 불공정하게 대역폭을 독점하게 된다. 그러나 이 문제를 응용에서 해결할 수는 없고 네트워크에서 공정성 문제를 해결해야 한다.

(3) 낮은 대역폭과 높은 지연시간을 갖는 링크에서 한 개의 TCP 접속을 이용하는 것은 슬로우-스타트에 의해 링크를 효과적으로 사용하지 못하기 때문에 여러개의 TCP 접속이 필요하다.

HTTP/1.1은 TCP 접속을 설정하고 해제함으로써 발생하는 TCP 과부하의 양과 네트워크 트래픽의 양을 줄이기 위해서 지속적 연결(persistent connection)을 지원한다. 즉, 동일한 TCP 연결을 재사용해서 여러 개의 요청을 보내는 파이프라

이닝(pipelining)이 가능하다. 그러나 파이프라이닝상의 요청과 응답은 정확한 순서대로 처리된다. 즉, 두 번째 아이콘을 보려면 첫 번째를 먼저 봐야 하므로 사용자는 다중 접속보다도 느리게 느껴진다.

1.3.3 캐싱

HTTP/1.0에서는 단지 캐싱만을 지원했지 캐시가 클라이언트와 서버사이에서 어떤 규칙으로 동작해야 하는지가 명시되지는 않았다. 그 결과 대부분의 콘텐트 제공자와 사용자는 HTTP/1.0의 캐싱 모델을 신뢰하지 않았다. HTTP/1.1은 서버와 클라이언트가 캐시의 내용이 갱신되는 상황에서 캐시의 능력과 조건을 제어할 수 있도록 잘 정의된 캐싱 모델을 만드는데 주력하였다. 이 캐싱 모델은 다단 캐싱 모델을 지원하고 인터넷 캐싱 프로토콜과 결합하여 캐시를 국가적, 세계적 수준으로 배치할 수 있도록 하였다.

1.4 HTTP 확장 프레임워크 (Extension Framework) [8][9]

한편 웹상에서 동작하는 다양한 형태의 응용들이 증가함에 따라 특정 응용의 필요에 따라 HTTP/1.1을 확장하는 문제가 관심 사항으로 대두되고 있다. 현재 시도는 프로토콜에 새로운 헤더를 추가하고, 상대방 소프트웨어가 헤더를 인식하고 처리하는 방향으로 연구되고 있다. 그 결과 강력한 기반구조를 제공하는 효과적인 방법을 제공하지는 못한다.

HTTP/1.0과 HTTP/1.1의 차이점에 대한 자세한 내용은 [10]을 참고하기 바람.

2. HTTP-NG 문제 제기 및 요구사항

HTTP/1.1은 HTTP/1.0과의 호환성을 제공해야 했으므로 기존의 문제점을 완전히 해결하지는 못했다. 따라서 기존 HTTP와의 호환성을 무시하고 완전히 HTTP를 재설계하기 위한 차세대 HTTP의 필요성이 1995년 Simon Spero에 의해 제안되었다. 시간이 경과하면서 내용 교섭(content negotiation)과 같은 몇가지 기능은 HTTP/1.1에 포함되기도 하였다.

2.1 HTTP-NG 문제 제기 [12]

WWW가 인터넷 확산의 계기가 되었고 HTTP가 그 중심 역할을 담당했지만 HTTP/1.0과 HTTP/1.1은 성능 측면과 모듈화 측면에서 상당한 제약 사항을 포함하고 있다.

모듈화는 단순화를 위한 중요한 수단이다. HTTP/1.x를 자세히 살펴보면 메시지 수송, RMI (Remote Method Invocation), 문서처리에 관련된 메소드(예, 폼 처리, 검색)와 같은 3 계층 기능이 혼합되어 있음을 알 수 있다. 모듈화 부족은 HTTP의 진화를 매우 어렵게 하고 응용들에게 문제를 발생시킨다. 응용은 HTTP 위에서 수행되므로 HTTP 구조에 상당한 영향을 받는다. DCOM, Java RMI, CORBA와 같은 다른 일반 RMI 시스템도 HTTP 위에서 동작하므로 HTTP에 영향을 받는다. 이를 피하기 위해서 어떤 응용은 HTTP의 일부분(subset)을 이용하기도 한다.

성능측면에서 HTTP는 인터넷 트래픽의 대부분을 차지하기 때문에 인터넷 사용자에게 인터넷 자원을 효과적으로 사용할 수 있도록 HTTP가 만들어져야 한다. 특히 현재 인터넷 사용자와 응용에게 전달되는 성능은 매우 낮다. 앞으로 낮은 대역폭과 높은 지연을 갖는 무선 통신을 이용한 인터넷 사용자와 응용이 급속이 증가할 것이므로 이러한 환경에서 HTTP가 효과적으로 설계되어야 한다.

2.2 HTTP-NG 요구사항

2.2.1 단순성 (simplicity)

단순성은 시스템을 이해하고, 구현하고, 유지하는데 중요한 요소이다. 현재의 HTTP는 모든 응용 프로토콜마다 요구되는 각기 다른 기능들을 한 개로 통합시킴으로써 단순화에 실패하였다. 그 결과 프로토콜을 이해하고, 구현하고, 수정하는 것을 매우 어렵게 만들었다. 모듈화는 코어 기반구조를 작게 만들어서 단순성을 제공한다. 이에따라 복잡한 응용도 그 코어 위에서 잘 정의될 수 있다. 즉, 어떤 응용은 많은 기능을 포함할 수 있고 어떤 응용은 제한된 기능을 갖는 최소한의 구현만을 이용할 수 있다. HTTP-NG는 HTTP의 요소들을 계층화하고 모듈화함으로써 단순성과 수행능력, 그리고 유연성을 제공해야 한다.

2.2.2 분산 확장성 (distributed extensibility)

분산 작업, 프린팅, RPC와 같은 다양한 범위의 응용들이 분산환경에서 HTTP 확장을 위해 제안되었다. 그러나 HTTP/1.x의 비구조화된 확장 모델 때문에 제안된 여러 방법들이 동일한 메시지를 이용할 수 있는지 미지수이다. 그 결과 기존 웹 기반구조로는 많은 웹 응용을 수용할 수 없게 되고 웹 응용의 분열과 상호운용성 부족을 발생시켰다. 확장성의 의미는 확장된 응용이 인터넷 전체에서 적용되는 것을 요구하지 않는다. 대신에 양측이 특정 확장 기능을 구현하였거나, 새로운 기능을 확장한 한 측에서 다른 측에게 그 기능을 이해하기를 요구하거나 거절할 수 있도록 하거나, 확장 기능을 협상하도록 함으로써 해결할 수 있다.

2.2.3 범위성 (scalability)

HTTP는 모든 인터넷과 인트라넷상의 클라이언트, 서버, 프록시, 캐시, 게이트웨이, 터널, 기타 시스템에서 효과적으로 동작해야 한다. HTTP는 인터넷 대역폭의 대부분을 소비하는 단일 프로토콜이므로 보다 효과적인 프로토콜로 대체되어야 한다. 그러나 성능 측정 결과 [13], 시스템 성능에 영향을 미치는 부분이 웹 모델에서 한 부분에 국한되지 않고 하위 수송계층부터 상위 사용자 인터페이스까지 모든 계층에 의해 영향을 받음을 알았다.

2.2.4 네트워크 효율성 (network efficiency)

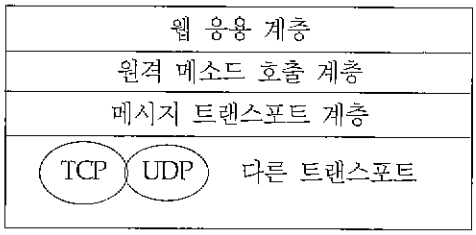
인터넷의 대역폭과 지연시간이 급속히 개선되고 있지만 무선 PDA, 이동 컴퓨터, 위성 링크에서는 여전히 대역폭과 지연시간은 제한될 것이다. 그러나 지연시간과 대역폭은 다른 요소이다. 즉, 위성 인터넷은 높은 대역폭을 제공할 수 있지만 나쁜 지연시간을 갖는다. 반면에 셀룰러 모뎀과 무선 시스템은 지연시간도 크고 대역폭도 낮다. 현재 대부분의 웹 사용자는 집에서 저속 모뎀을 사용하므로 이에 대한 효과적인 기능이 제공되어야 한다.

HTTP/1.1은 지속적 연결과 파이프라이닝을 제공하여 네트워크의 성능을 높이려고 하였지만 단지 HTTP/1.0에서 발생된 TCP 과부하를 줄이도록 설계되었기 때문에 HTTP가 안고있는 프로토콜 과부하를 제거하지는 못한다. 예를 들면 HTTP/1.1에서 메시지의 길이를 찾는 방법으로 5가지 다른 방법을 제공하는데 ([6]의 4.4장 참고) 그 중에서 어떤 방법을 사용하였는지를 결정하는데 상당한 시간을 소요한다.

머신이 읽을 수 있는 메시지와 사람이 읽을 수 있는 메시지 둘다 비록 ASCII 문자열로 인코딩되지만 컴퓨터에서 처리할 때는 다르다. HTTP에서 MIME 기반의 헤더 인코딩은 HTTP를 사람이 읽을 수 있는 프로토콜로 설계하는 문제를 발생시켰다. 그 결과 장황한(verbose) 메시지로 구성되었고 매우 복잡한 파서를 필요로 했다. 전형적인 HTTP 요청 메시지는 약 250 바이트로 구성되고 차후 HTTP 요청 메시지의 90%가 중복된 내용을 포함하고 있다. 이것은 낮은 대역폭에서 데이터 교환을 느리게 하는 요인이다. 만일 HTTP가 저속의 접속에서 상당한 성능 향상을 달성하지 못하면 상호호환성을 보장하지 않더라도 보다 경량 프로토콜을 선택해야 한다.

2.2.5 수송계층의 유연성 (transport flexibility)

3. HTTP-NG 구조



(그림 1) HTTP-NG 계층 구조

2장의 요구사항을 만족하기 위해서 W3C의 HTTP-NG 프로젝트 [11]에서는 HTTP를 (그림 1)과 같이 "메시지 트랜스포트", "원격 메소드 호출 (RMI, Remote Method Invocation)", 그리고 "웹 응용" 계층으로 나누었다.

3.1 메시지 트랜스포트 계층

최하위 계층이다. 중간 계층인 원격 메소드 호출 계층에서 사용하는 메시지를 수송한다. "메시지 트랜스포트"는 기존의 TCP, UDP와 같은 인터넷 트랜스포트 프로토콜 위에서 동작할 수 있다. HTTP-NG에서는 다양한 메시지 트랜스포트 서브스택과 서비스를 이용할 수 있으므로 향후 진화에 따른 유연성을 제공한다.

웹과 다른 응용은 아래와 같은 서비스를 필요로 한다.

- 다이얼-업 라인과 무선 통신망에서 RTT를 줄이기 위해서 메시지의 파이프라이닝
- 이미 캐시된 응답이 out-of-order로 도착할 때 캐시되지 않은 응답을 기다리지 않고 캐시로부터 빠른 응답뿐만 아니라 페이지의 빠른 디스플레이를 지원하는 메시지의 다중화
- 메시지 길이를 결정할 때 파싱 부하를 줄이기 위한 효과적인 레코드 마킹
- 많은 응용에서 필요로 하는 callback 함수

이들 서비스는 서로가 구별되는 다른 서비스이다. HTTP-NG에서는 이들 서비스들을 함께 묶어서 "WebMUX"라 불리는 단일 필터를 제공한다 [15]

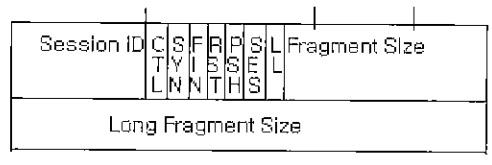
3.1.1 WebMUX [15]

WebMUX는 상위계층 응용 프로토콜로부터 하위 트랜스포트를 분리한 세션 관리 프로토콜이다. WebMUX는 TCP와 같은 신뢰성있는 스트림 지향 프로토콜 위에서 데이터 스트림을 멀티프랙싱함으로써 응용에게 경량 통신 채널을 제공한다. WebMUX의 목표는 다음과 같다.

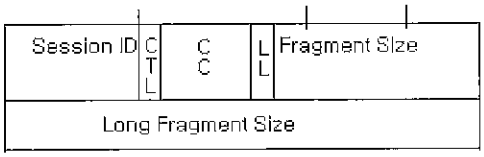
- 서버와의 협상(negotiation) 또는 왕복 시간이 소요되지 않는 비확인(unconfirmed) 서비스
- 단순한 설계
- 고성능
- 크레디트 기반의 흐름제어 (교착상태 제거)
- 동일 TCP 접속상에 여러 프로토콜의 다중화
- 한 방향으로의 접속 설정 (또는 세션 개시자로의 callback)
- WebMUX 프로토콜 위에 완전한 소켓 인터페이스 구축
- 데이터 스트림에서 alignment 기능을 제공하여 데이터 마샬링과 관련된 프로토콜이 쉽게 이용하도록 함.

(1) WebMUX 헤더

WebMUX 헤더는 (그림 2-a)와 (그림 2-b)와 같이 항상 big-endian 바이트 순서이며 데이터 헤더와 제어 헤더로 구성된다.



(그림 2-a) 데이터 헤더



(그림 2-b) 제어 헤더

CTL: Control. PSH: PUSH
SES: Session LL. Long Length
CC: Control Code

WebMUX 헤더는 항상 32 비트 단위로 구성된다. Fragment Size는 프라그멘트 크기로 WebMUX 헤더와 패딩 바이트를 제외한 값이다. Long Length 비트가 세팅된 WebMUX 헤더는 WebMUX 헤더 다음에 Fragment Size 필드 값으로 32 비트의 Long Fragment Size 필드를 이용해야 한다. WebMUX는 어떤 문자열에 대해 정수값을 할당하여 작고 쉬운 표기법을 이용하는데 이 정수 값을 atom이라 한다. 현재 atom은 Protocol ID에 사용되며, Protocol ID의 문자열은 URI 값이다. Atom은 *InternAtom* 제어 메시지를 이용하여 정의된다.

가. Protocol ID

세션에서 사용되는 프로토콜을 식별할 때 사용한다. IANA (Internet Assigned Number Authority) 포트 번호 또는 atom이다. Protocol ID 값은 다음과 같다.

* 0-0xFFFF: IANA에 등록된 TCP 프로토콜
* 0x10000-0x1FFFF: IANA에 등록된 UDP 프로토콜
* 0x20000-0x2FFFF: 하부접속마다 정의된 MUX atom
* 0x30000-0x3FFFF: 서버가 할당한 Protocol ID. 본 프로토콜 범위가 아님.

나. Session ID

각 세션은 세션 식별자를 할당한다. 0과 1 이하의 세션 식별자는 현재 사용하지 않는다. TCP 접속 개시자가 할당하는 Session ID는 짝수이고 수신측은 홀수를 할당한다. 프록시는 Session ID를 알지 못하면 변경없이 전달한다. Session ID는 세션의 양방향이 모두 종료될 때 해제되거나 새로운 세션에 의해 재사용될 수 있다.

다. 세션 설정

송신측은 SYN 비트를 세팅하고 사용하지 않는 Session ID를 할당한 첫 번째 메시지를 전송함으로써 새로운 세션을 설정한다. 이때 Fragment Size 필드는 그 세션의 Protocol ID 값으로 해석한다. 수신측은 SYN 메시지를 보냄으로써 반대 방향에 대한 세션을 생성하거나, FIN 메시지를 보내어 반대 방향 세션을 사용하지 않음을 알리거나, RST 메시지를 보내 에러를 알린다.

라. 정상적인 세션 종료

FIN 메시지를 보냄으로써 세션은 종료된다. WebMUX의 양방향 접속의 해제는 각 방향에서 FIN 메시지를 별도로 보냄으로써 종료된다.

마. 비정상적인 세션 종료

RST 메시지를 보냄으로써 세션은 비정상적으로 종료된다. 그 세션에 대해 남아있던 데이터는 버린다. RST 비트를 세팅한 프라그멘트의 페이로드는 에러 메시지의 URI를 포함하는 널 종료 문자열 (null terminated string)과 비정상 종료 원인에 대한 널 종료 UTF-8 문자열을 포함한다.

바. 메시지 경계

메시지 경계는 PUSH 비트가 세팅된 메시지를 보냄으로써 표시한다. 그 경계는 이 메시지의 마지막 옥텟과 다음 메시지의 첫 번째 옥텟 사이에 세팅된다.

사. 흐름 제어

간단한 크레디트 스킴을 이용하여 흐름 제어를 수행한다. 즉, 전송 프라그멘트는 그 세션의 크레디트를 초과해서는 안된다. 초기 크레디트 값은 16 KByte 이다. *AddCredit* 제어 메시지를 이용한다.

아. 종단점 (end point)

WebMUX의 목적중의 하나가 추가적인 TCP 접속없이 TCP 접속을 시작한 프로세스의 객체들(objects)에게 callback을 허용하는 것이다. *DefineEndPoint* 제어 메시지는 TCP 접속을 통해 도달할 수 있는 특정 URIs를 광고한다. MUX의 프로토콜 ID는 특정 종단점(end point)에서 채널의 상대적인 값이다. 즉, <Endpoint ID><Protocol ID> 쌍을 이용하여 MUX 채널을 완전하게 식별할 수 있다. Endpoint ID는 http://foo.com/bar/"와 같이 어떤 종단점에 대한 URI 이름이다. 어떤 Endpoint는 여러 개의 Endpoint ID를 가질 수 있다. 접속 개시자는 안전성을 보장하기 위햇 종단점 정보를 숨길 수 있다.

자. 제어 메시지

WebMUX 제어 메시지의 Control 비트는 항상 세팅된다. 제어 메시지는 세션이 개방되지 않은 경우를 포함해서 모든 세션에 보낼 수 있다. Control Code는 SYN, FIN, RST, PUSH 비트를 다시 사용한다. 제어 헤더의 Control Code는 다음과 같은 제어 메시지 유형을 결정한다.

InternAtom (0) : 양방향

Session ID 필드는 정의될 atom 값으로 사용한다. Fragment Size 필드는 UTF-8 인코딩 문자열의 길이이다. Fragment 필드는 사용하고자 하는 문자열을 포함한다.

DefineEndPoint (1) : 양방향

Session ID 필드는 무시한다. Fragement Size 필드는 이 TCP 접속상에서 실제로 가용한 endpoint를 명명하기 위한 Protocol ID로 이용한다. 즉, 이 메시지는 callback을 위해 TCP 접속을 이용하거나 또는 그 TCP 접속의 상대방 프로세스에 도달될 수 있는 프로토콜 Endpoint를 광고하는데 사용한다.

SetMSS (2) : 양방향

프라그멘트 크기를 제한한다. Session ID는 '0'이다. Fragement Size 필드는 최대 프라그멘트 크기를 저장한다. 이 값이 '0'이면 프라그멘트 크기가 제한되지 않았음을 의미한다.

AddCredit (3) : 수신측->송신측

Session ID 필드는 특정 세션을 가리킨다. Fragement Size는 허용된 크레디트 값이다. '0은 무제한 크레디트를 의미한다.

SetDefaultCredit (4) : 수신측->송신측

Session ID는 '0'이다. Fragment Size 필드는 초기 디폴트 크레디트 값을 지정한다.

NoOP (5) : 양방향

어떤 기능도 수행하지 않는다.

3.2 원격 메소드 호출 (Remote Method Invocation) 계층 [16]

중간 계층이다. 클라이언트가 서버에 있는 자원에 대해 어떤 연산을 기동하여 서버의 서비스를 이용하는 일반적인 요청/응답 메시지 계층이다. 본 계층 프로토콜은 기존의 CORBA, DCOM, Java RMI의 기능을 한가지 형태로 통합할 수 없지만 이들 방법들이 기술적으로 또는 정치적인 문제 때문에 웹에서 단일 모델로 선택될 수 없으므로 단일화된 원격 메소드 호출 방법을 지원할 수 있다. 본 모델은 응용 계층에서 프록시를 지원하는 경우에 hop-by-hop 연산을 가정한다.

3.2.1 이슈

- 바이트 순서 : 모든 값은 인터넷에서 사용하는 big-endian을 적용한다.
- 정렬(alignment)과 패딩 : 각 값의 마샬(marshal) 형식은 32 비트 경계를 갖고, 그 뒤에 패딩이 붙는다.
- 마샬링 양식 : 마샬링은 XDR 사양에서 명시된 XDR 형식을 따른다.

3.2.2 메시지

InitializeConnection 메시지는 호출자(caller)가 올바른 서버와 접속이 되었는지를 검증할 때 사용

한다. *DefaultCharSet* 메시지는 양측이 디폴트 문자를 지정할 때 사용한다. *Request* 메시지는 원격 서버상에서 연산을 수행시킨다. *Reply* 메시지는 서버가 클라이언트에게 *Request*에 대한 연산 결과를 전달하는데 사용한다. *TerminateConnection* 메시지는 한쪽이 정상적인 접속 종료시에 사용한다. 한편 이 프로토콜은 확장성과 수선성(tailorbility)를 제공하기 위해서 "확장 헤더" 기능을 제공한다.

또한 원격 호출과 마샬링/언마샬링을 위한 boolean, enumeration, numeric, string, sequence, record, array, union, pickle, reference, object 유형을 제공하고 접속 예외(connection exception)을 위한 예외 코드 값을 제공한다.

3.3 웹 응용

하위 두 계층을 제외하고 남은 기능은 메소드(GET, HEAD, PUT, 내용 교섭, 캐싱, 접근 제어 등)를 포함한 HTTP/1.1의 연산과 응용 계층 서비스이다. 응용 계층은 응용에 따라 다양하게 정의되는 부분이다. HTTP-NG 구조에서는 여러 응용이 공존하며 기존 응용에 관계없이 새로운 응용을 추가할 수 있는 기능을 제공한다. HTTP-NG는 기존 웹 응용에 대한 성능 향상이 주 목적은 아니지만 필수적으로 기존 기능을 보다 진보된 기술로 발전시키고 HTTP/1.1이 내재하고 있는 제약사항을 최소화할 필요가 있다.

4. 결론 및 향후 방향

HTTP-NG의 목적은 웹에서 새로운 응용을 개발하고 수용하는데 있다. 또한 웹에서 기존 응용을 HTTP-NG로 옮기는데 있다. 물론 현재 웹 응용의 통합 부분인 내용 교섭과 캐싱과 같이 HTTP/1.1 프로토콜 자체에서 제공하는 서비스도 포함될 것이다. 그러나 HTTP/1.1은 빙산의 일각이다. HTTP는 인터넷과 인트라넷에서 확장되고 진보되어야 한다. 그러나 HTTP/1.x의 확장과 이에 따른 HTTP-NG와의 상호운용성 문제는 중요한 이슈가 될 것이다. 특히, RMI 계층의 인터페이스 사양이 미비함에 따라 암호화 정책을 어렵게하고 웹에서의 수용을 더디게할 것이다.

이러한 상황을 변화시키기 위해서는 응용 서비스 제공자와 관련 종사자들이 기존 응용 뿐만 아니라 새로운 응용에 HTTP-NG를 수용해야 한다. 특히 기반구조(infrastructure) 수준 변화를 위해서는 다음과 같은 기술이 고려되어야 할 것이다.

- HTTP/1.1 Upgrade 헤더 필드
- HTTP Extension Framework
- 프로토콜 변환 게이트웨이
- HTTP-NG 서버를 위한 디렉토리 서비스
- HTTP-NG 서버를 위한 DHCP
- 특정 서버가 HTTP-NG 기능을 포함하고 있음을 지시하는 새로운 DNS 기록
- 새로운 URI 스킴

참고문헌

[1] W3C Home Page, http://www.w3c.org

[2] 한명우 역, 유닉스 웹 서버, 도서출판 대림, 1997.

[3] T. Berners-Lee, R. Fielding, H. Frystyk, "HyperText Transfer Protocol -- HTTP/1.0," RFC 1945, May 1996.

[4] N. Borenstein and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," RFC 1521, Sep. 1993.

[5] R. Fielding and et. al, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2068, Jan. 1997.

[6] R. Fielding and et. al, "Hypertext Transfer Protocol -- HTTP/1.1," Internet Draft, draft-ietf-

http-v11-spec-rev-06, Nov. 1998.

[7] V. Jacobson. "Congestion Avoidance and Control," SIGCOMM '88, ACM. August. 1988.

[8] HTTP Extension Framework, http://www.w3c.org/Protocols/HTTP/ietf-http-ext/

[9] H. F. Nielsen, P. Leach, and S. Lawrence, "HTTP Extension Framework," draft-frystyk-http-extensions-03, Mar. 1999.

[10] B. Kristhnamurthy, J. C. Mogul, and D. M. Kristol, "Key Differences between HTTP/1.0 and HTTP/1.1," AT&T Work Project No. 3116-17-1001, Dec. 1998.

[11] IETF HTTP-NG Working Group Home Page, http://www.w3c.org/Protocols/HTTP-NG/

[12] H. F. Nielsen, M. Spreitzer, B. Janssen, and J. Gettys, "HTTP-NG Overview," draft-frystyk-httpng-overview-00.txt, Nov. 1998.

[13] H. F. Nielsen, and et. al, "Network Performance Effects of HTTP/1.1. CSS1, and PNG," ACM SIGCOMM '97, Sep. 1997.

[14] B. Jansen, H. Frystyk, and M. Spreitzer, "HTTP-ng Architectural Model," draft-frystkyk-httpng-arch-00.txt, Aug. 1998.

[15] J. Gettys and H. F. Nielsen, "The WebMux Protocol," draft-gettys-webmux-00.txt, Aug. 1998.

[16] B. Janssen, "w3ng: Binary Wire Protocol for HTTP-ng," draft-janssen-httpng-wire-00.txt, Aug. 1998.

최 선 완

1984년 홍익대학교 전자계산학과 (이학사)
1986년 한국과학기술원 전산학 (석사)
1996년 한국과학기술원 전산학 (박사)
1986년-1996년 한국전자통신연구소 선임연구원
1996년-현재 안양대학교 정보통신공학과 교수
관심분야 : 인터넷 멀티미디어 통신, 위성/무선 인터넷