(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년09월07일
(11) 등록번호 10-1655715
(24) 등록일자 2016년09월01일

(51) 국제특허분류(Int. Cl.)
H04L 29/08 (2006.01) *H04L 29/06* (2006.01)
(52) CPC특허분류
H04L 67/1095 (2013.01)
H04L 65/1069 (2013.01)
(21) 출원번호 10-2015-7032976
(22) 출원일자(국제) 2014년04월21일
심사청구일자 2015년11월18일
(85) 번역문제출일자 2015년11월18일
(65) 공개번호 10-2015-0136141
(43) 공개일자 2015년12월04일
(86) 국제출원번호 PCT/EP2014/001052
(87) 국제공개번호 WO 2014/173521
국제공개일자 2014년10월30일
(30) 우선권주장
1307340.8 2013년04월23일 영국(GB)
(56) 선행기술조사문헌
US6892240 B1*
KR1020090010416 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
구루로직 마이크로시스템스 오이
핀란드 투르쿠 20100 린난카투 34
(72) 발명자
카르카이넨 투오마스 미카엘
핀란드 에프아이-20320 투르쿠 라우타란카투 2 비17
하카라이넨 발테리
핀란드 에프아이-20720 투르쿠 쿨마쿠자 1비 에이에스 2
카레보 오씨
핀란드 에프아이-37800 아카 케툰한타 1
(74) 대리인
김태홍, 김진희

전체 청구항 수 : 총 12 항 심사관 : 문해진

(54) 발명의 명칭 **ＨＴＴＰ를 이용한 양방향 실시간 통신 시스템**

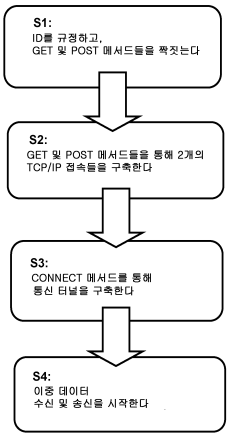
(57) 요 약

HTTP 호환형 통신을 지원하도록 동작 가능한 통신 시스템을 통해 통신 링크를 구축하는 방법(S1~S4)이 제공된다. 상기 방법은,

(a) 상기 시스템을 이용하여, HTTP와 연관된 GET 및 POST 메소드들의 조합을 이용함으로써 시스템의 2개의 노드

(뒷면에 계속)

대 표 도 - 도2



사이에 양방향 실시간 통신 링크를 구축하는 단계와;

(b) HTTP와 연관된 CONNECT 메소드를 이용하여 상기 양방향 통신 링크를 TCP/IP 및/또는 UDP 터널링하는 단계를 포함한다.

HTTP 호환 통신을 지원하도록 동작 가능한 통신 시스템이 또한 제공되고, 상기 통신 시스템은 HTTP와 연관된 GET 및 POST 메소드들의 조합을 이용하여 시스템의 2개의 노드 사이에 양방향 실시간 통신 링크를 구축하도록 동작 가능하며, 상기 양방향 통신 링크는 HTTP와 연관된 CONNECT 메소드를 이용하여 TCP/IP 및/또는 UDP 터널링된다. 상기 통신 시스템은 시스템이 소프트웨어 또는 하드웨어 방화벽에서 및/또는 통신 시스템에서 실행되는 안티바이러스 소프트웨어 애플리케이션에서 추가의 구성이 필요 없는 방식으로 공지의 HTTP 전송 프로토콜을 이용하여, 암호화되거나 암호화되지 않은 양방향 전이중 통신을 제공할 수 있다는 점에서 장점이 있다.

(52) CPC특허분류
H04L 67/02 (2013.01)
H04L 67/06 (2013.01)
H04L 69/16 (2013.01)

명 세 서

청구범위

청구항 1

HTTP 호환형 통신을 지원하도록 동작 가능한 통신 시스템(5)에 있어서, 상기 통신 시스템(5)은 HTTP와 연관된 GET 메소드 및 POST 메소드(method)의 조합을 이용하여 상기 통신 시스템(5)의 2개의 노드들(10A, 10B) 사이에 양방향 실시간 통신 링크(40)를 구축하도록 동작 가능하며, 상기 양방향 실시간 통신 링크(40)를 통한 데이터 교환은 청크(chunked) 방식으로 또는 일련의 다편(multipart) 데이터 블록으로서 구현되며,

상기 양방향 실시간 통신 링크(40)를 통해 전달되는 데이터 청크 및 다편 데이터 블록 중 적어도 하나에 대한 최대 세그먼트 크기(maximum segment size, MSS)가 통신 네트워크에서 가장 약한 링크의 MSS에 기초하여 결정되고;

상기 양방향 실시간 통신 링크(40)를 통한 상기 데이터 교환이 상기 청크 방식으로 또는 상기 일련의 다편 데이터 블록으로서 전송 인코딩(transfer encoding)을 이용하여 구현되는 것을 특징으로 하는 통신 시스템(5).

청구항 2

제1항에 있어서, 상기 양방향 실시간 통신 링크(40)는 HTTP와 연관된 CONNECT 메소드를 이용하여 TCP/IP 터널링 및 UDP 터널링 중 적어도 하나로 터널링되는 것을 특징으로 하는 통신 시스템(5).

청구항 3

제1항에 있어서, 상기 양방향 실시간 통신 링크(40)는 양방향 통신을 제공하기 위한 수신 접속 및 송신 접속을 포함하고, 상기 접속들은 비어있는(empty) 청크 및 비어있는 다편 데이터 블록 중 적어도 하나가 수신될 때까지 개방 상태로 유지되는 것을 특징으로 하는 통신 시스템(5).

청구항 4

제1항에 있어서, 상기 양방향 실시간 통신 링크(40)는 상기 양방향 실시간 통신 링크(40)를 통해 전달된 데이터의 암호화를 이용하도록 동작 가능한 것을 특징으로 하는 통신 시스템(5).

청구항 5

제1항에 있어서, 상기 양방향 실시간 통신 링크(40)는 그래픽 데이터, 이미지 데이터, 비디오 데이터, 오디오 데이터, 텍스트 데이터, 비구조화 데이터 중의 적어도 하나의 통신을 제공하도록 동작 가능한 것을 특징으로 하는 통신 시스템(5).

청구항 6

HTTP 호환형 통신을 지원하도록 동작 가능한 통신 시스템(5)을 통해 양방향 실시간 통신 링크(40)를 구축하는 방법에 있어서,

(a) 상기 통신 시스템(5)을 이용하여, HTTP와 연관된 GET 메소드 및 POST 메소드의 조합을 이용함으로써 상기 시스템(5)의 2개의 노드들(10A, 10B) 사이에 양방향 실시간 통신 링크(40)를 구축하는 단계;

(b) 상기 양방향 실시간 통신 링크(40)를 통해 데이터를 청크 방식으로 또는 일련의 다편 데이터 블록으로서 교환하는 단계; 및

(c) 통신 네트워크에서 가장 약한 링크의 최대 세그먼트 크기(maximum segment size, MSS)에 기초하여 상기 양방향 실시간 통신 링크(40)를 통해 전달된 데이터 청크 및 다편 데이터 블록 중 적어도 하나에 대한 최대 세그먼트 크기(MSS)를 결정하는 단계를 포함하고,

상기 양방향 실시간 통신 링크(40)를 통한 상기 데이터 교환은 상기 청크 방식으로 또는 상기 일련의 다편 데이터 블록으로서 전송 인코딩(transfer encoding)을 이용하여 구현되는 것을 특징으로 하는 양방향 실시간 통신 링크(40)를 구축하는 방법.

청구항 7

제6항에 있어서, HTTP와 연관된 CONNECT 메소드를 이용하여 상기 양방향 실시간 통신 링크(40)를 TCP/IP 터널링하거나, UDP 터널링하거나, 또는 이 둘 다를 행하는 단계를 포함한 것을 특징으로 하는 양방향 실시간 통신 링크(40)를 구축하는 방법.

청구항 8

제6항에 있어서, 상기 양방향 실시간 통신 링크(40)는 양방향 통신을 제공하기 위한 수신 접속 및 송신 접속을 포함하고, 상기 접속들은 비어있는 청크 및 비어있는 다편 데이터 블록 중 적어도 하나가 수신될 때까지 개방 상태로 유지되는 것을 특징으로 하는 양방향 실시간 통신 링크(40)를 구축하는 방법.

청구항 9

제6항에 있어서, 상기 양방향 실시간 통신 링크(40)는 상기 양방향 실시간 통신 링크(40)를 통해 전달된 데이터의 암호화를 이용하도록 동작 가능한 것을 특징으로 하는 양방향 실시간 통신 링크(40)를 구축하는 방법.

청구항 10

제6항에 있어서, 상기 양방향 실시간 통신 링크(40)는 그래픽 데이터, 이미지 데이터, 비디오 데이터, 오디오 데이터, 텍스트 데이터, 비구조화 데이터 중의 적어도 하나의 통신을 제공하도록 동작 가능한 것을 특징으로 하는 양방향 실시간 통신 링크(40)를 구축하는 방법.

청구항 11

소프트웨어 프로그램이 저장되어 있는 컴퓨터로 판독가능한 기록 매체에 있어서, 상기 소프트웨어 프로그램은 제6항에 기재된 통신 링크 구축 방법을 구현하기 위해 컴퓨팅 하드웨어 상에서 실행 가능한 것을 특징으로 하는 컴퓨터로 판독가능한 기록 매체.

청구항 12

제11항에 있어서, 상기 소프트웨어 프로그램은 HTTP로 표현되고 HTTP에 따라 동작하는 통신 네트워크의 서버 상에서 실행 가능한 것을 특징으로 하는 컴퓨터로 판독가능한 기록 매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 통신 시스템, 예를 들면, 각종 유형의 디지털 데이터, 예를 들면, 그래픽 데이터, 이미지 데이터, 비디오 데이터, 오디오 데이터 등을 통신하기 위해 실시간 하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol, HTTP)을 이용하는 통신 시스템에 관한 것이다. 또한, 본 발명은 각종 유형의 데이터를 통신하기 위한 전술한 통신 시스템의 동작 방법에 관한 것이다. 더 나아가, 본 발명은 머신 판독가능 데이터 기억 매체에 기록된 소프트웨어 제품에 관한 것이며, 이 소프트웨어 제품은 전술한 방법들을 구현하기 위해 컴퓨팅 하드웨어 상에서 실행 가능하다.

배 경 기 술

[0002] 개관으로, 하이퍼텍스트 전송 프로토콜(HTTP)은 현대 인터넷을 구현하기 위해 널리 사용되고 있다. 이 프로토콜은 분산된 협력적 하이퍼미디어 정보 시스템을 위한 응용 프로토콜이다. 구현시에, HTTP는 네트워크를 규정하는 논리 링크를 이용하여 네트워크를 구축하도록 동작하는 객체들의 다중선형 집합이고, 상기 링크는 종종 노드들 간의 네트워크 관계를 규정하는 "하이퍼링크"라고 불리운다.

[0003] HTTP는 인터넷용으로 구현될 때 예를 들면 클라이언트 서빙 모델에서의 요청 응답 프로토콜로서 기능하도록 동작한다. 이 모델에서, 웹 브라우저는 클라이언트를 구현하기 위해 선택적으로 사용되고, 서버 상에서 실행되는 소프트웨어 애플리케이션은 웹사이트를 호스팅할 수 있다. 동작시에, 소정의 클라이언트는 HTTP 요청 메시지를 서버에 제출하고, 서버는 HTML 파일과 같은 리소스 및 기타 콘텐츠를 제공함으로써 응답하거나 또는 클라이언트 대신에 데이터 프로세싱 기능을 수행하거나, 또는 응답 메시지를 클라이언트에게 반송하기도 한다. 전술한 웹 브라우저는 각종 방법으로, 예를 들면, 사용자 에이전트로서, 웹 크롤러(web crawler)로서, 또는 인터넷에 의해

유도된 데이터 콘텐츠에 액세스하거나, 이를 소모하거나 또는 디스플레이하는, 컴퓨팅 하드웨어 상에서 실행되는 임의의 다른 소프트웨어로서 구현되기 쉽다.

[0004] HTTP는 중간 네트워크 요소들이 클라이언트와 서버들 사이에서 통신할 수 있도록 설계된다. 인터넷의 하이 트래픽 웹사이트는 종종 데이터 및/또는 서비스 전달을 위한 응답 시간을 개선하기 위해 상류 서버 대신에 콘텐츠를 전달하도록 동작하는 웹 캐시 서버를 이용한다. 더욱이, 개인 네트워크 경계에 있는 HTTP 프록시 서버는 전세계적으로 라우팅 가능한 인터넷 어드레스 없이, 즉 외부 서버를 통해 메시지를 중계함으로써 클라이언트의 통신을 촉진하기 위해 유리하게 사용된다.

[0005] HTTP 리소스는 통합 리소스 로케이터(Uniform Resource Locator, URL)라고도 부르는 통합 리소스 식별자(Uniform Resource Identifier, URI)를 이용함으로써 소정의 네트워크에서 식별되고 위치확인된다. 또한, URI 및 하이퍼링크는 서로 상호링크된 하이퍼텍스트 문서들의 웹을 형성할 수 있는 하이퍼텍스트 마크업 언어(Hypertext Markup Language, HTML)로 표현된다.

[0006] HTTP 세션은 일련의 네트워크 요청 응답 트랜잭션(transaction)들에 의해 구현된다. 예를 들면, HTTP 클라이언트는 서버 상의 특정 포트에 대한 전송 제어 프로토콜(Transmission Control Protocol, TCP) 접속을 구축함으로써 요청을 개시한다. HTTP 서버는 클라이언트의 요청 메시지를 청취하고 상태 라인, 예를 들면, "HTTP/1.1 200 OK"를 관련 메시지와 함께 되보냄으로써 응답한다. 이 관련 메시지의 본문(body)은 종종 요청된 리소스이지만, 대안적으로 에러 메시지가 반송될 수도 있다.

[0007] HTTP는 식별된 리소스와 관련하여 수행되는 바람직한 동작을 표시하기 위해, 편의상 "버브(verb)"라고 부르는 메소드(method)를 규정한다. 리소스는 예를 들면, 하나 이상의 서버 상에 상주하는 실행가능한 객체(executable object)로부터의 출력 또는 데이터 파일이다. HTTP "버브"라고도 알려져 있는 HTTP 메소드의 예시들이 [표 1]에서 제공된다.

표 1

[0008]

"버브"	세부사항
GET	특정 리소스의 표시를 요청한다. "GET"를 이용한 요청은 데이터를 가져오기만 한다.
HEAD	GET로부터 획득가능한 것과 동일하지만 임의의 응답 본문이 없는 응답을 요청한다. "HEAD"는 가끔 메타데이터를 효율적인 방법으로 가져오기 위해 사용된다.
POST	URL에 의해 식별된 소정의 웹 리소스의 새로운 종속어(sub-ordinate)로서 해당 요청에 내포된 엔티티를 소정의 서버가 수용할 것을 요청한다.
PUT	내포된 엔티티가 공급된 URI(URL)와 관련하여 저장될 것을 요청한다. 만일 URI가 이미 존재하는 리소스를 언급하면, 그 리소스는 수정된다.
DELETE	특정 리소스의 삭제를 요청한다.
TRACE	수신된 요청이 소정의 클라이언트에게 반향되게 한다.
OPTIONS	주어진 URL과 연관된 서버에 의해 지원되는 HTTP 메소드를 반송한다.
CONNECT	예컨대, 전술한 비암호화 HTTP 프록시를 통해 TLS 및 SSL 암호화 통신(HTTP)을 촉진하기 위해 요청된 접속을 투명 TCP/IP 터널로 변환한다. 디폴트에 의해, HTTP 접속은 비암호화되는 반면에, HTTPS 접속은 암호화된다.
PATCH	소정의 리소스에 대한 부분 수정의 적용을 요청한다.

[0009] <HTTP 메소드(HTTP "버브")>

[0010] 따라서, 현대 웹 브라우저에 의해 사용되는 주요 전송 프로토콜은 전술한 HTTP이고, 몇 개의 관련 "생태계"(ecosystems), 및 이들이 이용하는 소프트웨어, 특히 브라우저 소프트웨어 애플리케이션은 HTTP를 이용하지 않으면 기능할 수 없다. 전술한 바와 같이, HTTP는 전송되는 요청들([표 1] 참조)에 기초를 두고, 이러한 요청들에 응답하여, 이미지 또는 오디오 스트림/파일과 같은 HTML 페이지 또는 이진 데이터가 상기 요청들의 수신에 응답하여 통상적으로 서빙된다.

[0011] 인터넷의 복잡성을 고려해서, 인터넷 통신 지연, 즉 "레이턴시(latency)"가 동작시에 발생할 수 있다. 이러한 지연은 데이터 교환 상황이 필요할 때, 예를 들면, 지연이 거의 없는 비디오 이미지 및/또는 오디오의 전송 및 수신과 같은 실시간 응답이 요망되는 양방향(전이중(full duplex)) 통신이 요망되는 경우에 문제를 야기할 수 있다. 인터넷을 통한 양방향 통신은 인터넷을 통한 음성 프로토콜(Voice-over-Internet-Protocol, VoIP)로부터

알려져 있고, 또한 예컨대 스카이프(Skype) 소프트웨어 등을 이용하여 동시적으로 제공되는 인터넷 기반 화상 회의로부터 알려져 있다. 상기 "스카이프"(Skype)"는 등록상표이다.

[0012] 특정 유형의 통신 수요를 다루기 위해 웹 사이트 http://tools.ietf.org/html/rfc6455에서 설명되어 있는 바와 같이 "웹소켓(WebSocket)"이라고 알려진 프로토콜을 사용하는 것이 공지되어 있다. 이것에 의해 하기의 통신 속성들이 달성될 수 있다.

[0013] (i) 웹소켓은 HTTP/HTTPS 터널 내에서 사용된다. 이 경우에, 흔히 방화벽이 웹 브라우저에서 동시적으로 사용되기 때문에 방화벽이 포트 80/443에 대하여 이미 개방되어 있다;

[0014] (ii) 웹소켓은 전이중 접속 모드에서 사용되고, 이때 하나의 TCP 접속만이 실시간으로 양방향으로 통신할 수 있다. 즉, 데이터 전달 방향을 변경함으로써 하나의 접속으로 데이터를 송신 및 수신할 수 있다.

[0015] 그러나, 이러한 웹소켓은 포트 종속적일 수 있는데, 이것은 바람직하지 않은 제한을 나타낸다.

발명의 내용

해결하려는 과제

[0016] 본 발명은 HTTP 통신 네트워크를 통한 양방향 데이터 통신을 개선된 방식으로 제공할 수 있는 통신 시스템을 제공하고자 한다.

[0017] 또한, 본 발명은 HTTP 통신 네트워크를 통한 양방향 데이터 통신을 제공하는 통신 시스템의 개선된 동작 방법을 제공하고자 한다.

과제의 해결 수단

[0018] 본 발명의 제1 양태에 따르면, HTTP 호환형 통신을 지원하도록 동작 가능하고, HTTP와 연관된 GET 및 POST 메소드들의 조합을 이용하여 시스템의 2개의 노드 사이에 양방향 실시간 통신 링크를 구축하도록 동작 가능하며, 통신 링크를 통한 데이터 교환이 청크 방식(chunked manner)으로 및/또는 일련의 다편(multipart) 데이터 블록으로서 구현되는 통신 시스템에 있어서, 통신 링크를 통해 통신되는 데이터 청크(chunk) 및/또는 다편 데이터 블록의 최대 세그먼트 크기(maximum segment size, MSS)가 상기 통신 링크를 지원하는 통신 네트워크 능력의 함수로서 최적화되고, 상기 통신 링크를 통한 데이터 교환이 상기 청크 방식으로 및/또는 상기 일련의 다편 데이터 블록으로서 전송 인코딩을 이용하여 구현되는 것을 특징으로 하는 통신 시스템이 제공된다.

[0019] 상기 통신 시스템은 레이턴시가 감소된 실시간 양방향 통신을 제공할 수 있다는 점에서 장점이 있다.

[0020] 선택적으로, CONNECT 메소드가 하기와 같은 3개의 상이한 시나리오 유형에서 사용될 수 있다:

[0021] (i) 접속이 타켓 내로 터널링된다. 이것은 유리하게 디폴트 시나리오이다;

[0022] (ii) 접속이 로컬 호스트를 통해 타겟으로 터널링되고, 이것에 의해 데이터가 로컬 서비스에서의 송신 프로세스로부터 포워딩 프록시 프로세스로 전송되며, 그로부터 데이터가 타겟으로 송신된다. 이 접근법은 안티바이러스 소프트웨어가 데이터를 분석하는 것 및 데이터를 무심코 차단시키는 것 또는 이와 다르게 데이터를 간섭하는 것을 방지할 수 있기 때문에 유리하다;

[0023] (iii) 접속이 포워딩 프록시 서버 내로 터널링되고, 그런 후, 포워딩 프록시 서버는 데이터를 데이터의 타겟으로 재지향시킨다. 이 접근법은 부하 밸런싱 시스템, 즉 클라이언트에 의해 야기되는 네트워크 부하가 타겟에 최적으로 분산되는 시스템에서 사용하기에 유리하다. 예를 들면, 백본 네트워크에서 데이터를 전송하는 것이 직접 접속을 통해서 전송하는 것보다 더 빠르다.

[0024] 선택적으로, 통신 시스템에 있어서, 통신 링크는 양방향 통신을 제공하기 위한 수신 접속 및 송신 접속을 포함하고, 상기 접속들은 비어있는(empty) 청크 및/또는 비어있는 다편 데이터 블록이 수신될 때까지 개방된 상태로 유지된다.

[0025] 선택적으로, 통신 시스템에 있어서, 통신 링크는 통신 링크를 통해 전달된 데이터의 암호화를 이용하도록 동작 가능하다.

[0026] 선택적으로, 통신 시스템에 있어서, 통신 링크는 그래픽 데이터, 이미지 데이터, 비디오 데이터, 오디오 데이터, 비구조화 데이터 중의 적어도 하나의 통신을 제공하도록 동작 가능하다.

[0027] 본 발명의 제2 양태에 따르면, HTTP 호환형 통신을 지원하도록 동작 가능한 통신 시스템을 통해 통신 링크를 구축하는 방법에 있어서,

[0028] (a) 통신 시스템을 이용하여, HTTP와 연관된 GET 및 POST 메소드들의 조합을 이용함으로써 시스템의 2개의 노드 사이에 양방향 실시간 통신 링크를 구축하는 단계와;

[0029] (b) 청크 방식으로 및/또는 일련의 다편 데이터 블록으로서 통신 링크를 통해 데이터를 교환하는 단계와;

(c) 통신 링크를 지원하는 통신 네트워크 능력의 함수로서 통신 링크를 통해 전달되는 데이터 청크 및/또는 다편 데이터 블록의 최대 세그먼트 크기(maximum segment size, MSS)를 최적화하는 단계를 포함하고,

[0030] 상기 통신 링크를 통한 데이터 교환은 상기 청크 방식으로 및/또는 상기 일련의 다편 데이터 블록으로서 전송 인코딩을 이용하여 구현되는 것을 특징으로 하는 통신 링크 구축 방법이 제공된다.

[0031] 선택적으로, 상기 방법에 있어서, 통신 링크는 양방향 통신을 제공하기 위한 수신 접속 및 송신 접속을 포함하고, 상기 접속들은 비어있는 청크 및/또는 비어있는 다편 데이터 블록이 수신될 때까지 개방된 상태로 유지된다.

[0032] 선택적으로, 상기 방법에 있어서, 통신 링크는 통신 링크를 통해 전달된 데이터의 암호화를 이용하도록 동작 가능하다.

[0033] 선택적으로, 상기 방법에 있어서, 통신 링크는 그래픽 데이터, 이미지 데이터, 비디오 데이터, 오디오 데이터, 비구조화 데이터 중의 적어도 하나의 통신을 제공하도록 동작 가능하다.

[0034] 본 발명의 제3 양태에 따르면, 머신 판독가능 데이터 기억 매체 상에 기록된 소프트웨어 제품이 제공되며, 상기 소프트웨어 제품은 본 발명의 제2 양태에 따른 방법을 구현하도록 컴퓨팅 하드웨어 상에서 실행 가능한 것을 특징으로 한다.

[0035] 선택적으로, 상기 소프트웨어 제품은 HTTP로 표현되고 HTTP에 따라 동작하는 통신 네트워크의 서버 상에서 실행 가능하다.

발명의 효과

[0036] 본 발명은 통신 시스템이 소프트웨어 또는 하드웨어 방화벽에서, 및/또는 통신 시스템에서 실행되는 안티바이러스 소프트웨어 애플리케이션에서 추가의 구성이 필요 없는 방식으로 공지의 HTTP 전송 프로토콜을 이용하여, 암호화되거나 암호화되지 않은 양방향 전이중 통신을 제공할 수 있다는 장점이 있다.

[0037] 더욱이, 본 발명은 통신 애플리케이션의 기능 및 신뢰성을 개선하고, 그에 따라서 시스템과 관련된 기술적 유지보수(maintenance) 문제, 예를 들면, 데이터 보안 설정을 단순화하는 장점이 있다.

[0038] 본 발명의 특징들은 첨부된 특허 청구범위에 의해 규정된 발명의 범위로부터 벗어나지 않고 각종 조합으로 결합될 수 있다는 것을 이해할 것이다.

도면의 간단한 설명

[0039] 이제, 본 발명의 실시형태를 하기의 도면을 참조하면서 단지 예로서 설명한다.

도 1은 HTTP를 이용하도록 동작 가능한 통신 네트워크를 보인 도이다.

도 2는 본 발명의 방법의 일련의 단계들을 보인 도이다.

도 3은 본 발명의 방법의 대안적인 일련의 단계들을 보인 도이다.

첨부 도면에 있어서, 밑줄 친 숫자는 밑줄 친 숫자가 그 위에 위치하고 있는 아이템 또는 밑줄 친 숫자가 인접해 있는 아이템을 표시하기 위해 사용된다. 밑줄 없는 숫자는 밑줄 없는 숫자를 아이템에 연결하는 선에 의해 식별되는 아이템에 관한 것이다. 숫자가 밑줄이 없고 관련 화살표를 수반하는 경우, 그 밑줄 없는 숫자는 화살표가 지시하는 전체적인 아이템을 식별하기 위해 사용된다.

발명을 실시하기 위한 구체적인 내용

[0040] 개관으로서, 도 1을 참조하면서, 이하에서는, 이용하는 HTTP의 설명(description)이 RFC2621, RFC2068 및 RFC1945와 같은 표준들을 따르는 방식으로 양방향 실시간 통신을 위한 HTTP와 관련하여, 지연, 즉 "레이턴시"를

감소시킬 수 있는 시스템 및 관련 방법을 설명하며, 이러한 시스템의 일부분이 숫자 5로 일반적으로 표시된다. 보통적으로, HTTP는 제1 노드(10A)와 제2 노드(10B) 사이에서 실시간 양방향 통신이 가능하도록 설계되지 않고, 주어진 클라이언트는 하기와 같은 방식으로 데이터의 실시간 송신 및 실시간 수신을 동시에 할 수 있다.

[0041] (i) 2개의 노드(10A, 10B) 사이에서 사용하는 통신 접속(20)은 암호화 형식의 양방향 통신을 지원하도록 동작 가능하다;

[0042] (ii) 바이러스 보호 소프트웨어(30)는 상기 통신 접속(20)을 통해 송신 및 수신되는 콘텐츠(40)를 간섭하지 않는다;

[0043] (iii) 방화벽(60)은 인터넷 트래픽, 즉 "WWW 트래픽"의 일반적인 장애(blockage)가 예컨대, 보안 금융 거래를 위해 사용되는 뱅킹 접속의 상황에서 차단되지 않은 한 네트워크 트래픽을 금지시킬 수 없다;

[0044] (iv) 네트워크 장치, 예를 들면, 브리지 및 라우터는 통신 접속(20)을 통해 전달되는 데이터를 분석하고 간섭할 수 없다.

[0045] 본 발명의 실시형태들은 하기의 특징을 이용함으로써 상기 (i) 내지 (iv)의 기능들을 다룰 수 있다.

[0046] (a) 2개의 서로 다른 유형의 GET 및 POST 메소드들(전술한 [표 1] 참조)가 사용되고, 여기에서 GET 메소드는 통신 접속(20)을 통한 수신 접속을 구축하고, POST 메소드는 통신 접속(20)을 통한 송신 접속을 구축한다;

[0047] (b) 상기 둘 다의 접속들은 현대 HTTP에서 사용하는 CONNECT 메소드를 이용하여 터널링된다;

[0048] (c) 뒤에서 더 자세히 설명하는 것처럼, "청크형" 또는 다편 전송 인코딩의 형태가 사용된다.

[0049] 종래에, HTTP는 인터넷 세션을 위해 사용되고, 이때 GET 및 POST 메소드들은 상호 독립적인 방식으로 사용된다. 예를 들면, GET 메소드는 웹 브라우저 클라이언트에 대한 호스트로서 기능하도록 동작하는 웹 서버로부터의 HTML 콘텐츠를 요청하기 위해 사용되고, 이때 GET 메소드를 위한 접속은 모든 응답 데이터가 호스트로부터 클라이언트에게 전달될 때까지 개방된 상태로 유지된다. 더욱이, 접속 절차는 데이터가 클라이언트로부터 호스트로 전달되는 것을 제외하고, POST 메소드([표 1] 참조)와 동일한 접속 절차가 사용된다.

[0050] 이하에서 설명하는 실시형태에 있어서, 통신은 주어진 소켓이 반이중(half duplex) 방식으로 사용되는 방식으로 실행되고, 이것은 공지된 접근법으로부터의 실시형태, 예컨대 전술한 웹소켓과 구별된다. 실시형태에 있어서, 데이터의 송신 및/또는 수신은 네트워크 인터페이스 카드들이 수신과 송신 간에 카드 자신들의 입력/출력(I/O) 상태를 전환할 필요가 없기 때문에 전이중 접속에서보다 더 효율적이다. 종래의 기술에서 사용하는 이러한 전환은 시스템 리소스를 소비하고, 이에 대응하여 잠재적 통신 속도를 감소시킨다.

[0051] 이하에서 설명하는 실시형태에 있어서, 소켓은 수신 모드 또는 송신 모드에서 HTTP GET 및 POST 메소드들의 초기화 후에만 사용된다. 그 결과, 사용되는 네트워크 어댑터는 반이중 상태에서만 동작할 필요가 있고, 이것은 HTTP GET 및/또는 POST 메소드 헤더들을 협의한 후 접속의 종료가 발생할 때까지 접속이 송신 모드 또는 수신 모드에서만 동작하기 때문에 네트워크 기반구조 및 장치 리소스를 절약한다. 더욱이, 예를 들면 방화벽 및 라우터, 즉 허브와 스위치가 스위칭 부하를 더 적게 수신하고, 따라서 하나의 전이중 접속만을 사용하는 공지된 현대의 전이중 통신 접근법만큼 빨리 파괴되지 않는 다른 장점들이 또한 발생한다. 따라서, 이하에서 설명하는 실시형태들은 예컨대 전술한 웹소켓보다 훨씬 더 큰 리소스 효율성이 있다.

[0052] 전술한 공지된 웹소켓은 식별되지 않은 접속 유형에 속하는 경우에 방화벽에 의해 쉽게 분석되고 그에 따라서 접속해제될 수 있으며, 이것에 의해, 관련 접속이 터널링되는지 여부와 관계없이 이 웹소켓의 사용을 금지 또는 제한시킨다. 이하에서 설명하는 실시형태에 있어서, GET 또는 POST 접속은 HTTP 프로토콜에 따라 기능하고, 따라서 방화벽은 이들 메소드들을 이용하는 통신을 제한시키거나 또는 금지시키지 못한다.

[0053] 이하에서 설명하는 실시형태에 있어서, TCP보다 실질적으로 3배 더 빠른 것으로 추정되는 UDP 프로토콜이 유리하게 사용된다. 선택적으로, 실시형태들은 피어 투 피어(peer-to-peer, P2P) 접속을 이용할 수 있고, 이것은 통신이 응용 레벨에서 달성될 수 있게 한다.

[0054] 여기에서 설명하는 실시형태는, 공지의 HTTP 구현이 GET 메소드와 POST 메소드 사이에 어떠한 링크도 없음에 반하여, 여기에서 설명하는 실시형태는 실시간 전이중 데이터 통신을 제공하기 위해 신규의 방식으로 함께 합병된 GET 메소드와 POST 메소드를 이용한다는 점에서, 공지의 HTTP 구현과 구별된다. 전술한 전이중 데이터 통신은 하나의 수신 접속과 하나의 송신 접속을 이용하여 구현된다. 하나의 수신 접속 또는 하나의 송신 접속은 하나의 반이중 접속 모드 또는 하나의 전이중 접속 모드를 사용할 수 있다.

[0055] 비록 실시형태들을 전송 제어 프로토콜(TCP)에 기초하여 이하에서 설명하지만, 대안으로서 사용자 데이터그램 프로토콜(UDP)이 사용될 수 있다는 것을 이해할 것이다. 비록 UDP 및 TCP 양자가 기저부 인터넷 프로토콜(Internet Protocol, IP)에 의존하고 UDP 데이터그램 및 TCP 세그먼트 양자가 IP 패킷으로 송신되지만, UDP는 UDP가 네트워크 어드레스 변환(network address translation, NAT) 횡단 기술을 이용하여 근거리 통신망(LAN) 내에서뿐만 아니라 외부 인터넷에서도 애플리케이션들 간 P2P 통신을 달성할 수 있게 하는 무접속 프로토콜이라는 점에서 구별된다. 이러한 접근법을 이용함으로써, 시스템(5)의 서버를 통해 데이터를 전송할 필요성이 회피되고, 그 결과 상당한 통신 네트워크 능력이 절약된다. 시스템(5)에서 UDP를 이용하는 것에 의한 추가의 장점은, UDP가 제어형 프로토콜이 아니기 때문에, UDP가 네트워크 통신 능력의 사용에 있어서 TCP보다 실질적으로 3배 더 효율적이라는 점이다. 더욱이, 예컨대 시스템(5)을 구현하기 위해 사용하는 IPv4 및 IPv6 통신 네트워크 양자에서 바이트 단위로 측정한 MSS 능력은 더 큰데, 그 이유는 UDP 헤더가 대응하는 TCP 헤더보다 더 작기 때문이다.

[0056] 비록, 이하에서는 GET 접속과 POST 접속 양자를 위한 TCP의 사용을 설명하지만, 선택적으로, 이들 접속 중의 하나만이 TCP를 사용하고 이들 접속 중의 다른 하나는 UDP를 사용하는 예도 예상된다. 더욱이, GET 접속과 POST 접속 양자가 UDP를 사용할 수 있는 것도 또한 예상된다.

[0057] 송신단 또는 수신단의 데이터가 발명의 범위로부터 벗어나지 않고 회로 교환형 데이터로부터 IP 기반형 데이터로 및 이에 대응하여 IP 기반형 데이터로부터 회로 교환형 데이터로 또한 변경될 수 있다는 것을 이해할 것이다.

[0058] 제1의 예시적인 실시형태에 있어서, 일련의 단계들이 도 2에 도시된 것처럼 다음과 같이 수행된다.

[0059] 단계 1(S1): 데이터 접속을 위한 클라이언트가 고유 스트림 식별자(ID)를 생성하고, 상기 ID는 GET 및 POST 메소드들을 함께 짝짓기 위해 사용되며, 그래서 데이터 접속을 구현하기 위해 사용되는 서버가 상기 GET 및 POST 메소드들의 쌍이 동일 클라이언트에 속한다는 것을 인식한다. 사용되는 ID에 대해서는 뒤에서 더 자세히 설명한다. 그러나, 고유 스트림 식별자(ID)가 송신 접속과 수신 접속을 결합하기 위해 사용될 때 상기 GET 및 POST 메소드들은 본 발명을 제한하지 않는다는 것을 이해할 것이다. 비록 스트림 ID의 주요 목적이 서버에서 클라이언트의 송신 접속과 수신 접속을 함께 묶기 위한 것이지만, 이것은 클라이언트를 인증하고 식별하기 위해 또한 동시에 사용될 수 있다. 이것은 서버가 위험성 있고, 에러 있으며/있거나 식별되지 않은 접속들을 이 접속들의 프로세싱이 계속되기 전에 폐기시킬 수 있다는 것을 의미한다. 이러한 기능은 서버를 보호하고, 식별되지 않은 접속 요청 및 불필요한 컴퓨팅에 의해 야기되는 서버 부하를 감소/제거할 수 있게 한다. 다시 말해서, 이것은 시스템이 리소스를 보존할 수 있게 하고, 이것은 특히 부하 밸런싱 시스템에서 에너지를 절약하고 서버 설비에서 필요로 하는 서버의 수를 감소시키는 장점을 제공한다.

[0060] 단계 2(S2): 그 후, 클라이언트는 예를 들면, 자신의 디폴트 포트 "80"에서 서버에 대한 2개의 TCP/IP 접속들을 구축하고, 그 다음에 클라이언트는 CONNECT 메소드와 연관된 헤더를 송신한다. 동작시에, CONNECT 메소드는 예를 들면, 일반적으로, 전술한 바와 같이 비암호화 프록시를 통한 TLS 및 SSL 암호화 통신(HTTP)을 촉진하기 위해 요청된 데이터 접속을 투명 TCP/IP 터널로 변환한다.

[0061] 단계 1 및 단계 2를 구현할 때, 예를 들면 SSL 1.0, SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2 또는 유사한 유형의 암호화와 같은 각종 형태의 암호화가 선택적으로 사용된다. 그러나, 전술한 터널은 상이한 "생태계" 간의 안전한 통신을 보장하기 위해 투명한 것이 유리하다. 더욱이, 악성 공격 또는 간섭에 대하여 보호되는 하드웨어를 사용하는 것이 또한 유리하다. 본 발명의 실시형태를 구현하기 위해 사용되는 이러한 투명 터널 접속은 해커, 적대적 소프트웨어, 안티바이러스 소프트웨어, 방화벽 소프트웨어, 또는 데이터 트래픽을 모니터링하고 분석하도록 동작하는 다른 장치 및/또는 소프트웨어가 터널 접속을 통해 전달되는 데이터를 간섭하는 것을 금지할 수 있다.

[0062] 단계 3(S3): 통신 터널용으로 사용되는 수신 또는 송신 접속에 따라서, GET 메소드 또는 POST 메소드의 헤더는 계속하여 송신 및 수신된다. 상기 헤더는 통신 터널에 의해 제공된 주어진 통신 세션에 대한 필요한 정보를 포함한다. 더욱이, 상기 헤더는 비록 헤더가,

[0063] (i) 결합/링크 접속을 위한 스트림 ID 종류의 정보; 및

[0064] (ii) 청크형 또는 다편 형식으로서의 전송 인코딩

[0065] 등의 파라미터를 포함하지만 종래 형식의 데이터 구조를 이용하는 것이 유리하다.

[0066] 헤더 내에 포함된 정보는 데이터의 전송 및 수신이 개별 데이터 블록으로서 발생하는 것을 보장한다. 유리하게, 데이터의 최대 세그먼트 크기(MSS)는 청크형 또는 다편 헤더용으로 사용되는 바이트의 양을 고려하여 통신 터널을 지원하는 네트워크의 능력에 대하여 최적화되고, 그래서 데이터를 전송 및 수신할 때 바이트가 손실되지 않고, 이것에 의해 신뢰성 있고 안전한 데이터 교환에 제공된다.

[0067] 이러한 네트워크 최적화는, 예를 들면, 접속된 클라이언트 장치를 서버에 결합하는 네트워크로부터 최대 전송 유닛(Maximum Transfer Unit, MTU) 값을 요청함으로써 구현된다. 이것에 의해 통신 네트워크에서 가장 약한 통신 링크를 식별할 수 있고, 그 다음에 가장 약한 링크에 의해 수용될 수 있는 레이트로 상기 가장 약한 링크와 연관된 클라이언트 장치에 송신용의 최대 세그먼트 크기(MSS)를 설정할 수 있다. MSS 값은 서버에 의해 시스템의 다른 클라이언트 장치에 선택적으로 전달된다. 이러한 네트워크 최적화는 하기의 단계를 포함한 방법을 이용하여 유리하게 구현된다.

[0068] 단계 A: 서버를 클라이언트 장치에 결합하는 가장 약한 링크를 시스템이 결정한다. 예를 들면, 주어진 데이터 링크의 MTU 값은 1500 바이트이다. 이 MTU 값이 TCP 헤더 바이트의 수, 즉 40 바이트만큼 감산되면, 1460 바이트가 이용가능할 수 있다. 이 1460 바이트는 MSS에 대응한다.

[0069] 단계 B: 가장 약한 식별된 링크의 MSS를 이용하여 주어진 세션의 MSS를 시스템이 결정한다.

[0070] 단계 C: 선택적으로, 시스템 내의 혼잡 제어를 방지하기 위해 시스템에서 사용되는 네이글(Nagle) 알고리즘이 디스에이블된다. 즉, 시스템의 소켓에서 TCP_NODELAY 옵션을 설정함으로써 달성되고, 이것은 네이블 알고리즘을 디스에이블한다. 네이글 알고리즘의 이러한 디스에이블은 대응하는 데이터 패킷이 전송되기 전의 송신 큐(queue)에 소정량의 바이트의 데이터가 추가되기 전에 네이글 알고리즘이 대기하기 때문에 바람직하다. 네이글 알고리즘이 디스에이블된 때, 시스템은 전술한 바와 같이 시스템에 의해서만 결정된 크기의 데이터 패킷을 전송할 수 있다.

[0071] 단계 4(S4): HTTP 요청 헤더가 송신되고 대응하는 성공적 응답이 서버로부터 수신되었으면, 그 다음에 이중 데이터 수신 및 송신이 시작된다. 이것에 의해 서버와의 2개의 접속들, 즉 수신 접속과 송신 접속이 성공적으로 이루어지고, 이 접속들은 비어있는 데이터 청크 또는 비어있는 다편 데이터 블록이 수신될 때까지 개방 상태로 유지된다.

[0072] 다음에, 2개의 예시적인 실시형태를 HTTP 코드에 의해 설명한다.

[0073] 예시 1: 클라이언트와 서버 간의 단순 터널화 수신 접속을 생성하도록 실행될 때 동작 가능한 HTTP 코드가 제공되고, 이때 IP 어드레스가 192.168.0.101인 피어는 IP 어드레스가 192.168.0.100인 호스트에 접속한다. HTTP 코드에서 "GET" 및 "CONNECT" 메소드들 둘 다의 사용은 하기와 같이 특정되는 청크형 전송 코딩과 함께 발견될 것이다.

<connect>

<send> CONNECT 192.168.0.100:80 HTTP/1.0 \r\n

<send> Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n

<send> \r\n

<send> GET /readstream? streamid=12345&param1=value1&param2=value2 HTTP/1.1 \r\n

<send> Host: 192.168.0.100 \r\n

<send> Transfer-Coding: chunked \r\n

<send> User-Agent : Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n

<send> \r\n

<recv> HTTP/1.1 200 OK \r\n

<recv> 5AD\r\n

<recv> 1453 bytes of data... \r\n

[0074]

<recv> 5AD\r\n

<recv> 1453 bytes of data... \r\n

...

<recv> 5AD\r\n

<recv> 1453 bytes of data... \r\n

<recv> 0 \r\n

[0075] <disconnect from 192.168.0.100>

[0076] 예시 2:

[0077] 클라이언트와 서버 간의 단순 터널화 송신 접속을 생성하도록 실행될 때 동작 가능한 HTTP 코드가 제공되고, 이 때 IP 어드레스가 192.168.0.101인 피어는 대응하는 IP 어드레스 192.168.0.100을 가진 호스트와 접속된다. HTTP 코드에서 "POST" 및 "CONNECT" 메소드들 둘 다의 사용은 하기와 같이 특정되는 청크형 전송 코딩과 함께 발견될 것이다.

<connect to 192.168.0.100>

<send> CONNECT 192.168.0.100:80 HTTP/1.0 \r\n

<send> Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n

<send> \r\n

<send> POST /writestream?streamid=12345&param1=value1&param2=value2 HTTP/1.1 \r\n

<send> Host: 192.168.0.100 \r\n

<send> Transfer-Coding: chunked \r\n

<send> User-Agent : Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n

<send> \r\n

<send> 5AD\r\n

<send> 1453 bytes of data... \r\n

<send> 5AD\r\n

<send> 1453 bytes of data... \r\n

...

<send> 5AD\r\n

<send> 1453 bytes of data... \r\n

<send> 0 \r\n

<recv> HTTP/1.1 200 OK \r\n

[0078]

[0079] 상기 2개의 예시 1 및 예시 2에 있어서, MSS는 1460 바이트인 것으로 가정하고, 그래서 실제로 최적화 청크에 대한 데이터 크기는 1453 바이트이다. 최적화 청크 크기는 [수학식 1]에서 주어진 공식을 이용하여 시스템에서 계산된다.

[0080] [수학식 1]

[0081] MSS = (청크 헤더의 시작부) - (청크 헤더의 종단부)

[0082] 청크 헤더의 시작부는 일반적으로 캐리지 리턴(Carriage Return, CR) 및 라인 피드(Line Feed, Lf)인, 예를 들면 16진수 표시법으로 실제 청크 데이터의 길이 및 하나 이상 선 문자의 종단의 길이로 구성된다. 청크의 종단은 청크를 완성하는 선 문자의 종단과 유사하다.

[0083] 다음에, 도 2를 참조하면, 단계 3(S3), 즉 CONNECT 메소드를 이용하여 접속 터널을 구축하는 단계는 도 3에 제공된 것처럼 선택적으로 생략되는 점을 이해할 것이다. 접속 터널은 터널에 대한 요구가 없을 때 생략된다. 따라서, 통신이 이용되지 않을 때는 단계 1, 단계 2 및 단계 4만이 사용된다. 또한, 도 2와 관련하여, 접속 터널은 GET 접속 또는 POST 접속에 대해서만, 즉 복수의 노드들 간에 비대칭 터널 통신 배열로 구성될 수 있는 것으로 이해되고, 선택적으로 통신 터널은 GET 접속 또는 POST 접속을 위해서만 사용된다.

[0084] 예시 3: MSS 최적화는 대응하는 http 청크 헤더가 프로세싱 지점에서 이미 제거되었고 데이터 블록의 페이로드가 100%이기 때문에 주어진 데이터 청크에 의해 제공된 주어진 페이로드에만 의존한다. 이제, 이러한 MSS 최적화는 원칙적으로 다음과 같은 개념에 기초를 둔다. 최대 송신 유닛(MTU)은 개별적 송신 버스트이고, 그래서 계층이 통과할 수 있는 최대 프로토콜 데이터 유닛은 예를 들면 1500 바이트 이상이고, MSS(최대 세그먼트 크기)는 MTU에서 프로토콜 헤더를 뺀 것과 동일한 데이터 크기를 갖는다. 본 발명에 따른 기술의 실시형태에 있어서, MSS는 정확히 해당 네트워크의 가장 약한 링크가 송신할 수 있는 바이트 단위의 데이터 양을 실어나른다. 그러

므로, 본 발명에 따른 기술을 사용할 경우 더 작은 패킷으로의 데이터 분할은 발생하지 않고, 이 때문에 데이터 송신의 속도 및 신뢰성이 증가하고, 따라서 예를 들면 와이파이(WiFi) 네트워크에서 충돌 및 패킷 손실이 더 적다.

[0085] MSS 최적화의 예는 다음과 같다.

[0086]

클라이언트 1	클라이언트 1과 클라이언트 2 사이의 운영자들	클라이언트 2
(네트워크 1500 바이트의 MTU)	(가장 약한 네트워크 600 바이트의 MTU)	(네트워크 1300 바이트의 MTU)

[0087] 접속 생성의 시작:

[0088] ICMP-핑(ping)이 네트워크를 테스트하기 위해 전송되고, 클라이언트 1과 클라이언트 2 사이의 통신은 MTU>600인 경우 금지되는 것이 탐지된다. 그러므로, MTU는 600 바이트로 설정되고, 이것은 40 바이트의 TCP 헤더가 생략된 후에, 즉 고려된 후에 MSS가 560 바이트임을 의미한다. UDP 프로토콜의 헤더는 더 작은 것으로 인식되고, 그래서 만일 UDP가 사용되면 페이로드가 그에 대응하여 더 커질 것이다.

[0089] 그런 후, 클라이언트 1은 6개 부분으로 분할된 3000 바이트 패킷을 클라이언트 2에 송신한다. 이러한 분할은 단순하고, 하기 공식에 따라 유리하게 구현된다: 전체 바이트 양이 네트워크의 최소 MTU로 나누어지고 시작 및 종료 청크 헤더가 차감된다. 즉 3000/(560-(5+2))=5.42개 패킷들이고, 이것은 다른 데이터가 송신을 위해 큐잉(queue)되지 않는 한 가장 가까운 정수의 패킷으로 반올림된다.

[0090] 패킷 1: 560 바이트가 송신되고, 그 중 페이로드는 553 바이트이다.

[0091] 패킷 2: 560 바이트가 송신되고, 그 중 페이로드는 553 바이트이다.

[0092] 패킷 3: 560 바이트가 송신되고, 그 중 페이로드는 553 바이트이다.

[0093] 패킷 4: 560 바이트가 송신되고, 그 중 페이로드는 553 바이트이다.

[0094] 패킷 5: 560 바이트가 송신되고, 그 중 페이로드는 553 바이트이다.

[0095] 패킷 6: 560 바이트가 송신되고, 그 중 페이로드는 235 바이트이다.

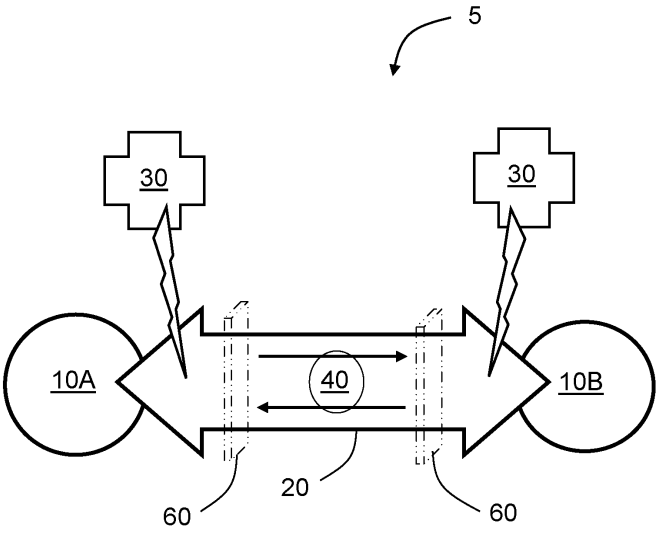
[0096] 만일 패킷이 분할 없이, 즉 하나의 3000 바이트 패킷으로 직접 송신되었으면, 패킷은 네트워크를 따라 운영자의 장치에 의해 분할, 즉 단편화되었을 것이고, 이것은 시간이 걸리고 잠재적으로 문제를 야기할 수 있으며, 아마도 손실된 패킷을 재전송할 필요가 있을 것이고, 이 모든 것은 수신자의 불안정한 네트워크에 의해 야기되는 랙(lag)에 기인하여 새로운 패킷의 송신 전에 송신자를 대기하게 한다.

[0097] 전술한 실시형태들은 첨부된 특허 청구범위에서 규정하는 본 발명의 범위로부터 벗어나지 않고 수정이 가능하다.

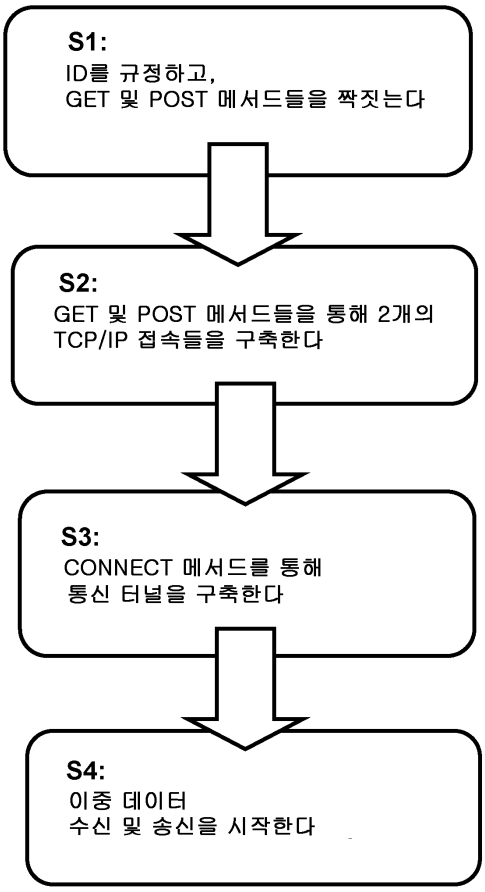
[0098] 본 발명을 설명하고 청구하기 위해 사용된 "포함하는", "구비하는", "통합하는", "구성된", "가진", "...인" 등의 용어는 비배타적 방식으로, 즉 명시적으로 설명되지 않은 아이템, 컴포넌트 또는 요소도 또한 존재할 수 있다는 의미로 해석되어야 한다. 단수형의 표현은 복수형에도 관계가 있는 것으로 또한 해석하여야 한다. 첨부된 특허 청구범위에서 괄호 내에 표시된 숫자들은 청구범위를 이해하는 데 도움을 주기 위한 것이고, 어떻게든 특허 청구범위에 의해 청구되는 주제를 제한하는 것으로 해석되어서는 안된다.

도면

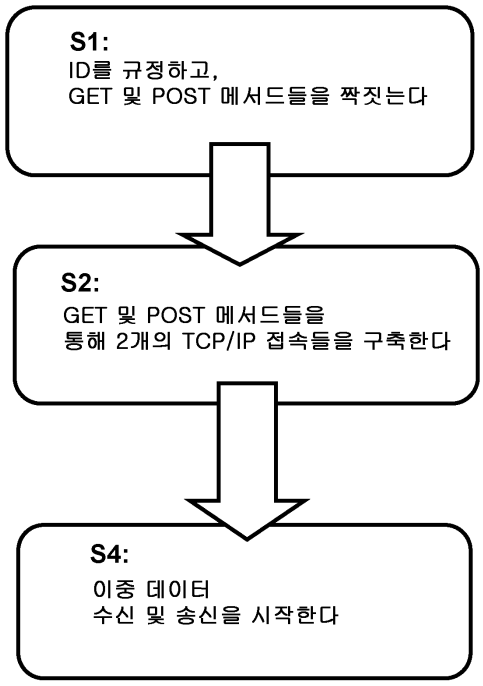
도면1


5
30
30
10A
40
10B
20
60
60

도면2


S1:
ID를 규정하고,
GET 및 POST 메서드들을 짝짓는다
S2:
GET 및 POST 메서드들을 통해 2개의
TCP/IP 접속들을 구축한다
S3:
CONNECT 메서드를 통해
통신 터널을 구축한다
S4:
이중 데이터
수신 및 송신을 시작한다

도면3


S1:
ID를 규정하고,
GET 및 POST 메서드들을 짝짓는다
S2:
GET 및 POST 메서드들을
통해 2개의 TCP/IP 접속들을 구축한다
S4:
이중 데이터
수신 및 송신을 시작한다