

(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。 Int. Cl. H04L 29/06 (2006.01)	(45) 공고일자 (11) 등록번호 (24) 등록일자	2006년04월07일 10-0569469 2006년04월03일
---	-------------------------------------	--

(21) 출원번호	10-1999-7010508	(65) 공개번호	10-2001-0012553
(22) 출원일자	1999년11월13일	(43) 공개일자	2001년02월15일
번역문 제출일자	1999년11월13일		
(86) 국제출원번호	PCT/US1998/009879	(87) 국제공개번호	WO 1998/52320
국제출원일자	1998년05월14일	국제공개일자	1998년11월19일

(81) 지정국 국내특허 : 알바니아, 아르메니아, 오스트리아, 오스트레일리아, 아제르바이잔, 보스니아 헤르체고비나, 바르바도스, 불가리아, 브라질, 벨라루스, 캐나다, 스위스, 중국, 쿠바, 체코, 독일, 덴마크, 에스토니아, 스페인, 핀란드, 영국, 그루지야, 헝가리, 이스라엘, 아이슬란드, 일본, 케냐, 키르기스스탄, 북한, 대한민국, 카자흐스탄, 세인트루시아, 스리랑카, 리베이라, 레소토, 리투아니아, 룩셈부르크, 라트비아, 몰도바, 마다가스카르, 마케도니아공화국, 몽고, 말라위, 멕시코, 노르웨이, 뉴질랜드, 폴란드, 포르투갈, 루마니아, 러시아, 수단, 스웨덴, 싱가포르, 슬로베니아, 슬로바키아, 타지키스탄, 투르크멘, 터키, 트리니다드토바고, 우크라이나, 우간다, 우즈베키스탄, 감비아, 인도네시아, 기니 비사우, 시에라리온, 베트남, 세르비아 앤 몬테네그로, 짐바브웨, 가나,

AP ARIPO특허 : 케냐, 레소토, 말라위, 수단, 스와질랜드, 우간다, 짐바브웨, 가나, 감비아,

EA 유라시아특허 : 아르메니아, 아제르바이잔, 벨라루스, 키르기스스탄, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크멘,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투갈, 스웨덴, 핀란드, 사이프러스,

OA OAPI특허 : 부르키나파소, 베닌, 중앙아프리카, 콩고, 코트디부아르, 카메룬, 가봉, 기니, 말리, 모리타니, 니제르, 세네갈, 차드, 토고,

(30) 우선권주장	08855902	1997년05월14일	미국(US)
	08855965	1997년05월14일	미국(US)
	08855977	1997년05월14일	미국(US)
	08856051	1997년05월14일	미국(US)

(73) 특허권자 사이트릭스 시스템스, 인크.
미국 플로리다 33309, 포트 라우더데일, 851 더블류. 사이프레스 크릭 로드

(72) 발명자 피터스브라들리,제이.
미국에프엘33067파크랜드7700에스,우드리지드라이브

블룸필드마크,에이.
미국에프엘33062폼파노비치2749에스.이.11스트리트

(74) 대리인 신영무
 최승민

심사관 : 김기완

(54) 서버와 클라이언트노드 사이의 접속을 관리하기 위한시스템 및 방법

요약

서버 클라이언트 시스템의 서버에 통신 제어를 제공하는 접속관리기는 클라이언트노드가 지정된 서버 포트와 기본적인 소 통을 설정하도록 허용한 후 서버에서 실행하는 어플리케이션에 통신 포트 특성에 접속한다. 이 후에, 특정 통신 포트는 클 라이언트노드에 의해 요구된 프로토콜 드라이버를 갖는 통신관리기에 의해 구체화된다. 어플리케이션은 HTML 페이지에 서 디스플레이된다. 일 실시예에서, 동일 데이터는 서버노드에서 실행하는 어플리케이션으로부터 실질적으로 동시에 적어 도 두개의 클라이언트노드로 전송된다. 클라이언트-서버 시스템 내 서버노드는 관련된 클라이언트노드를 위하여 해석언 어로 기록된 어플리케이션을 다운로드시키고 실행시킬 수 있다.

대표도

도 2

색인어

접속관리기, 통신, HTML 페이지, 네트워크, 링크

명세서

기술분야

본 발명은 분산형 클라이언트-서버 환경에서의 어플리케이션 실행에 관한 것 으로, 특히 클라이언트-서버 환경에서 해석 언어로 기록된 어플리케이션의 원격 실행에 관한 것이다.

배경기술

클라이언트 서버 컴퓨터 네트워크는 전형적으로 클라이언트 및 서버가 미리 설정된 몇 세트의 규약에 따라 통신을 설정할 것을 요청한다. 이들 규약들은 통신 프로토콜이라 한다. 이러한 프로토콜은 미리 정의되어 모든 클라이언트노드가 서버노 드와 동일한 통신 프로토콜을 사용할 수 있다. 대안으로, 서버는 각각의 클라이언트에 의해 사용된 통신 프로토콜의 기록 을 유지할 수 있으며 클라이언트가 서버상의 어플리케이션과 통신하기 위해 요청(request)을 보냈을 때 클라이언트와 통 신하기 위해 그 프로토콜을 사용한다.

이러한 통신방법과 관련된 문제는 요청된 모든 기능을 필요로 하지 않는 어플리케이션에 대해 프로토콜이 너무 엄격히 규 정될 수 있거나, 클라이언트가 서버와 통신할 수 있기 이전에 클라이언트 프로토콜이 서버에 미리 알려져야만 한다는 것이 다. 본 발명은 미리 규정된 프로토콜의 엄격함과 서버의 부분에 대한 예비접속 지식의 필요성을 배제하려는 것이다.

새도우잉(shadowing)(하나의 클라이언트노드에 대해 정해진 데이터를 거의 동시에 제2클라이언트노드로 전송하는 것) 및 브로드캐스팅(동일 데이터를 하나 이상의 클라이언트노드에 거의 동시에 전송하는 것)은 일반적으로 서버노드상의 특 정화된 전송 어플리케이션과 클라이언트노드 각각에 대해 특정화된 수신 어플리케이션을 사용하여 수행되었다. 쉐도잉은 데이터 트래픽을 모니터링하는데 유용하며 데이터 보존을 위해 전송되는 정보의 잉여 복사본의 생성과 시스템 보안 목적 에 유용하다. 동일 정보가 "실시간"(REAL-TIME)일 때 및 정보가 그 자체에 시작과 종료를 가지지 않을 때 브로드캐스팅 은 동일 정보를 많은 사용자에게 제공하는 데 유용하다. 예를 들어, 주가 시세 프로그램은 주어진 거래에 따라 다양한 주식 의 현재가를 단순히 전송하며 그 리스트는 주식 리스트가 소멸되면 가장 최근 가격을 반복한다. 따라서, 그것은 어디에서 그 리스트가 시작되는 지를 시세 프로그램에 지정하지 않은 사용자와는 관련이 없다.

일반적으로, 이러한 프로그램은 브로드캐스트 프로그램을 염두에 두고 기록되었으며 전송된 데이터를 수신하기 위해 특정화된 수신프로그램을 필요로 한다. 어플리케이션이 브로드캐스트 프로그램으로 기록되지 않았다면, 상기 어플리케이션에 의해 전송된 데이터는 일반적으로 다수의 클라이언트노드에 브로드캐스팅(broadcast)될 수 없다.

본 발명은 브로드캐스트 기능에 대해 기록되지 않은 프로그램이 네트워크를 통해 데이터를 브로드캐스트하는데 사용될 수 있도록 함으로써 이문제를 극복하려 한다.

하나의 특정화된 클라이언트 서버 네트워크는 지난 수년 동안 폭발적 성장을 보여 온 "인터넷"으로 일반적으로 알려진 컴퓨터의 세계적인 네트워크이다. 이러한 많은 성장은 월드 와이드 웹(WWW)의 대중성의 증가에 의해 이루어 졌다. WWW는 하이퍼 텍스트 마크업 언어(Hyper Text Markup Language; HTML)를 사용하여 기록된 파일의 모음이며, 일반적으로 웹페이지로 불려진다. HTML 파일은 웹 브라우저로 알려진 특정 어플리케이션을 이용하여 액세스되고 디스플레이될 수 있는데, 사용자가 간단한 그래픽 사용자 인터페이스(Graphical User Interface; GUI)를 사용하여 HTML 파일을 액세스할 수 있도록 한다.

HTML 파일을 처리하는(hosting) 서버는 하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol; HTTP)을 이용하여 통신할 수 있다. HTTP는 HTML 페이지 설명 언어를 사용하는 파일(텍스트, 그래픽, 이미지, 음향, 화상 등과 같은 다양한 형태가 될 수 있음)에 사용자 액세스를 제공하는 어플리케이션 프로토콜이다. HTML은 기본 문서 형식을 제공하며 개발자가 다른 서버 및 파일에 통신 "링크"를 지정하게 한다. HTML 컴플라이언트 클라이언트 브라우저(HTML-compliant client browser)의 사용은 유니폼 리소스 로케이터(Uniform Resource Locator) 또는 "URL"을 경유한 링크의 내역(목록)을 포함한다. 이러한 내역에 따라, 클라이언트는 링크에서 확인된 서버로 TCP/IP 요청을 작성하고 그 응답으로 "웹 페이지"를 수신한다. 또한, 조직(Organizations)은 WWW로부터가 아닌 그 조직 내로부터 액세스 가능한 HTML 파일을 제공할 수 있다. 이들 내부 네트워크 및 HTML 파일의 모음은 일반적으로 "인트라넷(Intranet)"으로 불려 진다.

HTML를 사용하여 기록된 파일은 "태그(Tags)"를 포함하는데, 그것은 특정 액션이 취해져야만 할 때 그 파일을 디스플레이하는 브라우저에 지시한다. 예를 들어, 태그는 브라우저에 (1) 그래픽 파일이 문서의 특정 위치(point)에 디스플레이되어야만 한다는 것; (2) 어떤 텍스트가 중심에 있어야만 한다거나, 볼드처리되어야만 한다거나 또는 달리 포맷되어야 한다는 것; (3) 문서의 배경이 음영처리되거나 특별한 패턴을 가져야만 한다는 것; 또는 (4) 다른 HTML이 현재 디스플레이되는 HTML브라우저 대신에 로딩되어야만 한다는 것 등을 지시할 수 있다.

월드 와이드 웹 및 다른 HTML 어플리케이션의 대중성은 광범위한 산업계를 대표하는 회사의 넓은 영역에서 마케팅과 판매 실적에 매력적이다. 다른 회사와의 차별이 점점 어려워지므로 많은 회사는 HTML의 고유의 정적(static) 성질을 극복하기 위해 애써 왔다. 또한, 정보를 공유하는(share) 방법으로 HTML파일을 이용하는 조직은 인트라넷이 많은 사용자에게 단지 정보 이상의 액세스를 제공하기 위한 유용한 방법이란 것을 인식하게 되었다.

그러나, HTML 파일들의 주요 단점은 그들이 본질적으로 정적이란 것이다. 즉, HTML은 "디스플레이 전용(Only)" 언어일 뿐이며, 그것은 HTML 페이지 내에서 어플리케이션 실행을 쉽게 허용하지 않는다. WWW의 대중성과 편재를 레버리지(leverage)로 이용하려는 회사들은 HTML 파일 내에 어플리케이션을 끼워넣는(embedding) 방법을 점진적으로 찾고 있다.

상표명 액티브엑스 오브젝트(ActiveX object)는 실행 어플리케이션을 디스플레이할 수 있는 HTML 파일을 제공하려는 하나의 시도이다. 상표명 액티브엑스 오브젝트는 상표명 액티브엑스 인터페이스를 가진 브라우저에서 사용될 수 있는 데이터 오브젝트이다. 이러한 오브젝트의 분명한 단점은 사용자 브라우저가 상표명 액티브엑스 인터페이스를 가지고 있지 않다면 그것은 실행 어플리케이션을 디스플레이 할 수 없다는 것이다. 모든 HTML 페이지의 주요 목적이 가능한 많은 사용자에게 의해 보여지도록 하는 것이므로 액티브엑스 오브젝트의 유용성을 제한한다.

또한 상표명 자바(JAVA)라 불리는 프로그래밍 언어가 실행 가능한 코드를 HTML 파일에 추가되도록 하는 방법으로 제안되었다. 상표명 자바는 언어이므로 그것은 특정한 브라우저 인터페이스를 필요로하지 않으며 잠재적으로 더 넓은 대중을 가진다. 그러나, 보통 애플릿(applet)이라 불리는 상표명 자바 프로그램은 실행전에 클라이언트로 다운로드된다. 이것은 클라이언트가 애플릿을 다운로드하기 위한 충분한 메모리가 부족하다는 문제가 있으며, 클라이언트가 충분한 메모리를 가지고 있을지라도 클라이언트가 애플릿을 다운로드하기 위해 기다릴 필요가 있다. 더욱이, 상표명 자바는 프로그래밍 언어 그자체이므로 실행 어플리케이션은 그들이 웹 페이지에 끼워넣기될 수 있기 전에 상표명 자바 언어로 재기록되어야만 한다.

발명의 상세한 설명

일 측면에 있어서, 본 발명은 클라이언트노드와 서버노드상에서 실행되는 어플리케이션 프로그램 간의 통신을 관리하는 통신 시스템 및 방법에 관한 것이다.

본 측면의 일실시예에 있어서, 방법은 클라이언트노드와 서버상에 설치된 일반 통신 포트간의 접속을 설정하는 단계를 포함한다. 방법은 엔드포인트(endpoint) 데이터 구조를 생성하는 단계, 클라이언트 스페이스를 엔드포인트 구조와 결합시키는 단계 및 어플리케이션과 결합된 특정 통신 포트용 프로토콜 스택(stack)을 생성하는 단계를 더 포함한다. 접속관리기에 접속이 통지(notification)되고 그 접속은 일반 통신포트로부터 특정 통신 포트의 프로토콜 스택에 전송된다.

본 측면의 다른 실시예에 있어서, 접속을 설정하는 단계는 마스터 네트워크 정보노드에 의한 어플리케이션 요청을 클라이언트노드로부터 수신하는 단계와, 마스터 네트워크 정보노드에 의해 요청된 어플리케이션을 갖는 서버의 서버 어드레스를 클라이언트노드에 제공하는 단계와, 제공된 어드레스에 따른 일반 통신 포트에 접속하기 위해 서버에 의한 요청을 클라이언트노드로부터 수신하는 단계, 및 클라이언트노드와 일반 통신 포트간의 접속을 설정하는 단계를 포함한다.

본 측면의 다른 실시예에 있어서, 통신 시스템은 서버노드 및 클라이언트노드를 포함한다. 서버노드는 일반적인 통신 포트를 구비한다. 클라이언트노드는 클라이언트노드와 서버노드의 일반 통신 포트간의 접속을 설정하는 통신 장치를 구비한다. 서버노드는 또한 엔드포인트 데이터 구조를 포함하는 프로토콜 스택을 포함하며 메모리에 설치된 클라이언트 스페이스를 포함한다. 클라이언트 스페이스는 프로토콜 스택과 관련된다. 통신관리기 및 통지 디바이스(device)는 서버노드 상에 설치된다. 통지 디바이스는 클라이언트노드와 일반 통신 포트간의 접속을 접속관리기에 통지하며 그 응답으로 통신관리기는 일반 통신 포트와 클라이언트노드간의 접속을 프로토콜 스택에 전송한다. 일실시예에 있어서 시스템은 다수의 프로토콜 스택의 각 프로토콜 스택과 통신하는 멀티플렉서(multiplexer)를 더 포함한다.

본 측면의 또 다른 실시예에 있어서, 본 발명은 끼워넣어진(embeded) 클라이언트노드와 통신하기 위한 컴퓨터-판독가능 프로그램 수단을 갖는 제조물품에 관한 것이다. 상기 제조물품은 선정된 포트를 통해 클라이언트노드와의 접속을 설정하기 위한 컴퓨터-판독가능 수단과, 엔드포인트 데이터 구조를 생성하기 위한 컴퓨터-판독가능 수단과, 메모리 스페이스를 엔드포인트 데이터 구조와 결합시키기 위한 컴퓨터-판독가능수단과, 메모리 스페이스 및 상기 결합된 엔드포인트 구조와 결합된 프로토콜 스택을 생성하기 위한 컴퓨터 판독가능 수단과, 선정된 포트와 클라이언트 간의 접속을 접속관리기에 통지하기 위한 컴퓨터-판독가능 수단, 및 선정된 포트와 클라이언트 간의 접속을 결합된 프로토콜 스택에 전송하기 위한 컴퓨터-판독가능 수단을 포함한다.

다른 측면에 있어서, 본 발명은 특정 언어로 재기록된 어플리케이션을 요청하지 않으며, 그리고 특정화된 인터페이스를 지원하는 사용자의 브라우저를 요청하지 않고 디스플레이된 HTML 파일내에 실행 어플리케이션을 디스플레이하는 방법에 관한 것이다. 어플리케이션은 다운로드 타임과 클라이언트측 메모리 제한을 완화시키기 위해 서버상에서 실행한다. 더욱이, 클라이언트는 다수 페이지에 대한 다수의 어플리케이션의 실행을 실시할 수 있으며, 어떤 어플리케이션도 종료하지 않고 HTML 문서사이클을 이동할 수 있다.

본 측면의 다른 실시예에 있어서, HTML 페이지에 실행 어플리케이션을 디스플레이하는 방법은 사용자가 어플리케이션 프로그램의 실행을 시작하기를 원하는 것을 알리는 입력을 사용자로부터 수신하므로써 시작한다. 어플리케이션이 실행될 윈도우 파라미터가 결정되고, HTML 페이지 내에서 어플리케이션 윈도우로의 통신 채널이 생성된다. 서버상에서 실행되는 어플리케이션 프로그램의 출력은 통신 채널을 통해 어플리케이션 윈도우에서 디스플레이된다.

본 측면의 또 다른 실시예에 있어서, HTML 페이지에서 실행 어플리케이션을 디스플레이하기 위한 장치는 파라미터 핸들러 및 네트워크 실행부를 포함한다. 파라미터 핸들러는 HTML 파일에 포함된 어플리케이션 실행 윈도우와 연관되는 파라미터를 수신한다. 네트워크 실행부는 파라미터를 파라미터 핸들러로부터 수신하며, 서버상에서 어플리케이션 프로그램의 실행개시를 야기하며, 파라미터 핸들러로부터 네트워크 실행부에 의해 수신된 파라미터에 따라 어플리케이션 실행 윈도우내에 실행 어플리케이션의 출력을 디스플레이 한다.

본 측면의 또 다른 실시예에 있어서, 제조물품은 HTML 페이지에 내장된 실행 어플리케이션을 디스플레이하기 위한 컴퓨터-판독가능 코드수단을 포함한다. 이 제조물품은 서버상의 어플리케이션 프로그램의 실행이 시작함을 알리는 입력을 클라이언트로부터 수신하기 위한 컴퓨터-판독가능 코드수단을 포함한다. 또한, 그 제조물품은 실행 어플리케이션이 디스플레이될 윈도우의 파라미터를 결정하기 위한 컴퓨터-판독가능 코드수단을 포함한다. 또한, 선정된 파라미터를 사용하여 HTML페이지에 통신채널을 생성하기 위한 컴퓨터-판독가능 코드수단과, 상기 통신 채널을 통해 어플리케이션 윈도우에서 서버상에서 실행되는 어플리케이션의 출력을 디스플레이하기 위한 컴퓨터-판독가능 코드수단을 포함한다.

본 측면의 또 다른 실시예에 있어서, HTML 페이지에 어플리케이션을 끼워넣기 위한 시스템은 서버, 네트워크 실행부, 파라미터 핸들러 및 HTML 파일을 포함한다. 상기 서버는 어플리케이션 프로그램을 저장하고 실행한다. 네트워크 실행부는 커맨드(command)를 특정 어플리케이션의 실행이 시작해야 한다는 것을 지시하는 서버에 전송하고, 네트워크 실행부는 서버상에서 실행되는 어플리케이션으로부터 출력을 수신한다. HTML 파일은 어플리케이션 윈도우를 포함한다. 어플리케이션 윈도우는 윈도우 파라미터를 파라미터 핸들러에 전달하며 네트워크 실행부로부터 어플리케이션 프로그램 출력을 수신한다.

본 발명의 또다른 측면은 동일 데이터를 하나 이상의 클라이언트노드로 거의 동시에 전송하기 위한 시스템 및 방법에 관한 것이다. 일실시예에 있어서, 본 발명은 동일 데이터를 서버노드상에서 실행되는 어플리케이션으로부터 일반화된 수신기 프로그램을 실행하는 적어도 2 이상의 클라이언트노드로 거의 동시에 전송하기 위한 방법에 관한 것이다. 그 방법은 제1 클라이언트노드와 서버노드상의 제1클라이언트 프로토콜 스택간의 접속을 설정하는 단계와, 서버상에서 실행되는 어플리케이션과 제1클라이언트 프로토콜 스택간의 접속을 설정하는 단계와, 제1최소 통신 프로토콜 스택을 제1클라이언트 프로토콜과 결합하는 단계와, 서버노드상에서 실행되는 어플리케이션과 제1최소 통신 프로토콜 스택간의 접속을 설정하는 단계와, 제2클라이언트노드와 서버노드에 대한 제2클라이언트 프로토콜 스택간의 접속을 설정하는 단계와, 제2최소 통신 프로토콜 스택을 제2클라이언트 프로토콜 스택과 결합하는 단계와, 제1최소 프로토콜 스택과 제2최소 프로토콜 스택간의 접속을 제공하는 단계와, 제2최소 프로토콜 스택과 제2클라이언트 프로토콜 스택간의 접속을 제공하는 단계, 및 데이터를 어플리케이션 프로그램으로부터 제1클라이언트 프로토콜 스택과 제1최소 프로토콜 스택에 거의 동시에 전송하는 단계를 포함한다.

본 발명의 측면에 있어서의 다른 실시예는 서버 및 둘 이상의 클라이언트노드를 포함하는 통신 시스템에 관한 것이다. 일 실시예에 있어서 서버노드는 어플리케이션 프로그램과, 어플리케이션과 전기통신하는(electrical communication) 제1클라이언트 프로토콜 스택과, 어플리케이션 프로그램과 전기통신하는 제1최소 프로토콜 스택과, 제1프로토콜 스택과 전기통신하는 제2최소 프로토콜 스택, 및 제2최소 프로토콜 스택과 전기통신하는 제2클라이언트 프로토콜 스택을 포함한다. 또한, 그 시스템은 제1클라이언트 프로토콜 스택과 전기통신하는 제1클라이언트노드와 제2클라이언트 프로토콜 스택과 전기통신하는 제2클라이언트노드를 포함한다. 어플리케이션으로부터의 데이터는 클라이언트 프로토콜 스택과 제1최소 프로토콜 스택으로 거의 동시에 전송된다.

또 다른 측면에 있어서, 본 발명은 클라이언트-서버 환경에서 해석언어(interpretive language)를 원격으로 실행하기 위한 방법에 관한 것이다. 클라이언트가 접속된 서버는 상표명 자바 애플릿과 같은 해석언어로 기록된 어플리케이션을 다운로드하고 실행한다. 서버는 클라이언트로부터 입력을 수용하고 스크린 데이터를 클라이언트에 제공한다. 이것은 어플리케이션을 처리 및 실행하는 연산 및 메모리 자원을 소모하는 클라이언트를 필요로 하지 않으며 통상적인 방법으로 어플리케이션을 수행하는 것과 같이 클라이언트에 나타나게 한다. 또한, 서버는 클라이언트보다 더 빨리 어플리케이션을 다운로드할 수 있다. 또한 서버는 클라이언트 노드 제어를 가능하게 하는 입력을 클라이언트로부터 수용하여 다운로드된 어플리케이션에 입력을 제공한다.

본 측면의 일실시예에 있어서, 본 발명은 클라이언트노드에 의해 만들어진 요청에 따라 서버노드에 어플리케이션을 다운로드함으로써 시작하는 해석언어로 기록된 어플리케이션을 원격으로 실행하기 위한 방법에 관한 것이다. 접속은 클라이언트노드와 서버상에 위치한 선정된 통신포트간에 설정되며, 서버는 엔드포인트 데이터 구조를 생성하여 서버에 의해 처리되는 클라이언트 스페이스를 엔드포인트 데이터 구조와 결합시킨다. 서버는 클라이언트 스페이스 및 결합된 엔드포인트 데이터 구조가 결합된 프로토콜 스택을 생성하고, 접속관리에 그 접속을 통지하며, 선정된 통신 포트와 클라이언트노드간의 접속을 결합된 프로토콜 스택에 전송케한다.

본 측면의 또다른 실시예에 있어서, 본 발명은 해석언어로 기록된 어플리케이션을 원격으로 실행하기 위해 그 내부에 끼워넣기된 컴퓨터-판독가능 프로그램 수단을 구비한 제조물품에 관한 것이다. 상기 제조물품은 클라이언트노드에 의해 만들어진 요청에 응답하여 어플리케이션을 서버노드에 다운로드하기 위한 컴퓨터-판독가능 프로그램 수단과, 클라이언트노드와 서버상에 위치한 선정된 통신 포트간의 접속을 설정하기 위한 컴퓨터-판독가능 프로그램 수단과, 엔드포인트 데이터 구조를 생성하기 위한 컴퓨터-판독가능 프로그램 수단과, 서버에 의해 처리된 클라이언트 스페이스를 엔드포인트 데이터 구조와 결합시키기 위한 컴퓨터-판독가능 프로그램 수단과, 클라이언트 스페이스 및 상기 결합된 엔드포인트 데이터 구조와 결합된 프로토콜 스택을 생성하기 위한 컴퓨터-판독가능 프로그램 수단과, 접속관리에 접속을 통지하기 위한 컴퓨터-판독가능 프로그램 수단, 및 선정된 통신 포트와 클라이언트노드간의 접속을 결합된 프로토콜 스택에 전송하기 위한 컴퓨터-판독가능 프로그램 수단을 포함한다.

본 측면의 또 다른 실시예에 있어서, 본 발명은 해석언어로 기록된 어플리케이션을 원격으로 실행하기 위한 시스템에 관한 것이다. 본 시스템은 선정된 통신 포트를 갖는 서버노드 및 클라이언트노드와 서버노드의 예정된 통신 포트 간의 접속을 설정하는 통신 디바이스를 갖는 클라이언트노드를 포함한다. 프로토콜 스택은 서버노드 상에 설치되며 프로토콜 스택은 엔드포인트 데이터 구조를 포함한다. 서버노드상의 메모리에 위치된 클라이언트 스페이스는 프로토콜 스택과 결합되며 해석언어로 기록된 어플리케이션에 대한 실행환경을 제공한다. 본 시스템은 서버노드상에 위치된 통신관리기 및 서버노드상에 위치된 통신 디바이스를 더 포함한다. 통신 디바이스는 접속관리기에 클라이언트노드와 선정된 통신 포트간의 접속을 통지하며 통신관리기가 선정된 통신 포트와 클라이언트노드간의 접속을 프로토콜 스택에 전송한다.

도면의 간단한 설명

본 발명은 첨부된 청구범위와 함께 설명된다. 상술된 본 발명의 장점들 뿐만 아니라 다른 특징들은 첨부된 도면과 관련하여 취해진 하기 설명을 참조함으로써 보다 쉽게 이해될 수 있다.

도 1은 본 발명을 채용하는 통신 시스템의 실시예의 개략적 다이어그램이다.

도 2는 클라이언트와 서버 사이의 통신 동안 발생하는 도 1의 서버의 다양한 구성요소들 사이의 접속을 나타내는 본 발명 실시예의 블록 다이어그램이다.

도 3은 다중 클라이언트노드 접속을 유지하고 관리하는 본 발명 실시예의 블록 다이어그램이다.

도 4는 HTML 페이지에 어플리케이션을 끼워넣기 위한 시스템 실시예의 블록 다이어그램이다.

도 5는 클라이언트노드의 다이어그램 도면이다.

도 6은 동일한 데이터를 어플리케이션로부터 하나 이상의 클라이언트에 전송하는 멀티플렉서의 사용을 나타내는 본 발명 실시예의 블록 다이어그램이다.

도 7은 브로드캐스트성능이 팬아웃(fan out)에 의해 증가되는 본 발명 실시예의 블록 다이어그램이다.

실시예

도 1을 참조하면, 개략적으로, 전형적인 네트워크(20)는 통신 링크(44)에 의해 상호 접속된 적어도 하나의 클라이언트노드(24)와, 적어도 하나의 서버노드(34, 34'), 및 마스터 네트워크 정보노드(40)를 포함한다. 도 1에 예시된 실시예는 통신 링크(44)를 근거리 통신망 링 또는 LAN 링으로 나타냈지만, 다른 통신 접속형태(topology)가 사용될 수 있다. 설명을 목적으로, 서버노드(34)는 클라이언트노드(24)에 의해 요구된 어플리케이션(30)을 갖는 것으로 고려된다. 또한, 설명의 목적상, 마스터 네트워크 정보노드(40)는 별개의 서버노드인 것으로 고려되지만, 사실상 마스터 네트워크 정보노드(40)는 어플리케이션실행 서버노드(34)이다. 임의의 LAN 상에서 다수 노드가 네트워크 정보노드(40)로 작용할 수 있지만, 어느 한 순간에 이러한 노드 중 하나만 시스템(20)용 마스터 네트워크 정보노드(40)로 설계될 수 있으며, 이 노드에는 서버 정보에 대한 클라이언트 요청(들)이 지시된다.

마스터 네트워크 정보노드(40)는 어플리케이션실행 서버노드(34, 34')용 어드레스의 테이블을 보유한다. 부가적으로, 마스터 네트워크 정보노드(40)는 각 어플리케이션실행 서버노드(34, 34')로부터 작동 레벨을 나타내는 메시지를 수신한다. 어플리케이션실행 서버노드(34, 34')의 작동 레벨은 각 어플리케이션실행 서버노드(34)의 어드레스와 함께 테이블에 보유되고 로드 레벨링(load leveling)에 적합한 통신체계(44)에 의해 사용된다.

클라이언트(24)가 어플리케이션실행 서버노드(34) 상에서 실행된 어플리케이션을 갖기를 원할 때, 클라이언트노드(24)는 프로토콜에 의해 미리 정의된 일반적인 통신포트 또는 마스터 네트워크 정보노드(40) 상의 "공지된" 통신포트에 요청을 송신한다. 일 실시예에서, 통신은 데이터그램 서비스의 경로에 의해 발생한다. 마스터 네트워크 정보노드(40)는 서버 어드레스 테이블을 액세스하고, 요청된 어플리케이션과 또한 가장 적은 로드를 갖는 어플리케이션 실행서버 또는 어플리케이션 서버(34)의 주소를 포함하는 메시지를 리턴한다(return). 후속하는 통신들은 클라이언트에 의해 서버노드(34)상에 있는 '공지된' 또는 미리 정의된 일반적인 통신포트에 자동 지정된다. 일 실시예에서, 마스터 네트워크 정보노드(40)에 대한 초기 조회(query)가 이루어지는 프로토콜의 형태는 마스터 네트워크 정보노드(40)에 의해 클라이언트노드(24)에 리턴된 정보의 프로토콜을 결정한다. 따라서, TCP/IP 데이터그램을 사용하는 요청이 이루어지면, 마스터 네트워크 정보노드(40)는 서

버(34)의 TCP/IP 어드레스를 클라이언트노드(24)에 리턴하며, 그 다음 클라이언트노드(24)는 그 프로토콜을 사용하여 서버노드(34)와 접촉할 것이다. 다른 실시예에서, 클라이언트(24)에 의해 어플리케이션 어드레스를 요청하는 데이터그램은 마스터 네트워크 정보노드(40)에 요청을 전송하기 위해 사용된 것과는 다른 프로토콜의 형태의 요청을 포함한다. 예를 들어, 클라이언트(24)는 IPX 프로토콜을 사용하여 마스터 네트워크 통신노드(40)에 요구하고 TCP/IP 프로토콜 어드레스로서 어플리케이션서버의 어드레스를 요구한다.

클라이언트노드(24)[실제적으로 클라이언트노드(24) 상의 클라이언트 프로세스(56)]가 서버노드(34, 34') 상의 어플리케이션과 통신을 원할 때, 클라이언트노드(24)는 원하는 어플리케이션을 갖는 서버(34)의 위치를 결정하기 위해 네트워크 요청을 발송함으로써 시작한다. 이 요청은 네트워크 상의 어느곳에선가 내재하는 마스터 네트워크 정보노드(40)[네트워크 브라우저(40)로서 언급됨]에 의해 수신된다. 도 1에서, 네트워크 브라우저(40)는 어플리케이션을 갖는 서버와는 다른 서버(40) 상에 존재하는 것으로 단순히 도시되어 있지만, 이것은 일반적인 경우가 아니다.

네트워크 마스터 정보노드(40)는 요구된 어플리케이션(30)을 갖는 서버노드(34)의 네트워크 어드레스를 클라이언트노드(24)에 리턴한다. 다음, 클라이언트노드(24)는 네트워크 마스터 정보노드(40)로부터 수신된 정보를 특정 서버(34) 상에서 실행하는 어플리케이션에 대한 요청에 접속하기 위해 사용한다. 상술된 바와 같이, 이러한 접속은 먼저 "공지된" 통신포트로 설정되고, 다음 접속 관리기의 제어하에서 특정 통신포트에 전송된다. 특정 통신포트는 서버노드(34) 상에서 실행하는 어플리케이션과 연관되어 특정 통신포트를 통해 클라이언트노드(34)와 통신한다.

상세하게는, 도 2를 참조하면, 클라이언트노드(24) 상의 클라이언트 프로세스(56)는 요구된 어플리케이션(62)을 포함하는 서버노드(34)의 어드레스를 얻기 위해 네트워크 마스터 정보노드(40)에 요청한다. 네트워크 마스터 정보노드(40)는 서버 어플리케이션(62)을 포함하는 서버노드(34)의 어드레스를 포함하는 메시지(58)를 클라이언트노드(24)로 리턴한다. 일 실시예에서, 접속의 순간에 사용된 프로토콜은 데이터그램 서비스이다.

클라이언트노드(24)는 리턴된 어드레스를 서버(34)와 통신채널(68)을 설정하기위해 사용한다. 클라이언트(24)에 의해 사용된 포트번호는 서버(34)가 클라이언트(24)와의 통신접속을 설정하는 포트로서 네트워크 프로토콜로 정의된 서버(34)에서 "공지된 포트"와 일치한다. 공지된 포트(72)는 엔드포인트 데이터 구조(78)를 주로 포함하는 기본 프로토콜 스택(76)을 갖는다.

엔드포인트 데이터 구조(78)는 프로토콜 스택(76)을 가리키고 이에 의해서 클라이언트 접속은 유일 표현 또는 클라이언트(24)에 대한 "핸들"을 설정한다. 엔드포인트 데이터 구조(78)는 서버(34)와 클라이언트(24)간의 연결이 접속관리기(80)와 서버(34)상의 다양한 어플리케이션간으로 옮겨질 수 있도록 해준다. 일 실시예에서, 엔드포인트 데이터 구조(78)는 클라이언트(24)에 대한 핸들을 포함할 뿐만 아니라 클라이언트 접속에 관계하는 다른 정보를 포함할 수 있다. 예시된 실시예에서, 어플리케이션 서버(34)는 특정 통신체계(예를 들어, LAN 또는 WAN) 상에서 활동을 모니터링하고 "TTY" 통신모드를 지원하기 위해 요구된 필요한 프로토콜 모듈만으로 이 최소 프로토콜 스택(76)을 초기화한다. "TTY" 통신모드는 트랜스포트층 상에 프로토콜추정이 없는 단순한 ASCII 스트림이다. 즉, 전송된 데이터의 압축, 암호화, 신뢰성, 프레임링(framing) 또는 전송데이터의 표시에 관한 어떠한 프로토콜층도 없다. 따라서, 서버(34) 상에서 실행하는 어플리케이션(62)을 원하는 클라이언트노드(24)는 TTY 통신모드를 지원하는 데 요구되는 최소 프로토콜 세트를 이용하여 공지된 통신포트(72)에 대한 접속을 설정한다.

서버노드(34) 상에 실행하는 접속관리기(80)는 공지된 통신포트(72)를 "리스닝"(listening)하여 접속요청(68)을 수신한다. 접속요청(68)이 클라이언트노드(24)로부터 수신되면, 접속관리기(80)는 이를 통지한다(84). 접속관리기(80)는 프로토콜이 상기 통지(84)에 따라 어떠한 프로토콜이 사용될지를 인식한다.

이 정보와 함께 접속관리기(80)는 신규의 최소 프로토콜 통신스택(104)를 생성하고, 실행환경(96)을 개시하고, 신규 최소 프로토콜 스택(104)을 실행환경(96)으로 설정(bind)한다. 일 실시예에서, 서버(34)는 이미 시작하였지만, 통신포트와 관련되지 않은 다수의 실행환경(96)을 포함한다. 이 실시예에서, 실행환경의 예비접속 개시는 접속 요청가 클라이언트(24)로부터 수신되었을 때 각각의 실행환경(96)이 개시되는 경우보다 빠른 응답시간을 허용한다. 실행환경(96)이 개시될 때, 클라이언트(24)에 의해 요구된 서버 어플리케이션(62)도 개시된다. 다른 실시예에서, 클라이언트(24)가 어플리케이션을 특정하지 않으면, 디폴트 어플리케이션이 개시되거나 또는 어플리케이션이 없는 단순한 실행환경(96)이 개시된다.

다음, 접속관리기(80)는 유일한 클라이언트 식별자 또는 핸들을 포함하는 클라이언트 접속을 공지된 포트(76)에서 신규 최소 프로토콜 스택(104)으로 이동시킨다. 최소 프로토콜 스택을 사용하는 접속관리기(80)는 이용할 수 있는 서비스를 표

시하는 TTY 데이터 스트림을 전송한다. 따라서, 클라이언트 접속을 검출하기 위한 방법은 접속이 최초로 설정된 포트에 독립적이다. 클라이언트노드(24)가 지정된 주기(예를 들어, 5초) 내에 서비스 유효 메시지로 응답하지 않으면, "서비스 유효" 메시지의 재송신이 서버(34)에 의해 수행된다.

클라이언트(24)가 메시지를 수신하면, 클라이언트(24)는 "서비스 유효" 메시지가 검출된 것을 나타내는 TTY 스트링(string)을 전송한다. 클라이언트(24)는 서버(34)가 응답하는 것을 대기하고 응답이 지정된 시간(예를 들어, 5초) 내에 없으면 클라이언트(24)는 메시지를 재전송한다. 다음, 접속관리기(80)는 클라이언트의 디폴트 통신 파라미터가 필요한지를 클라이언트(24)에게 질의한다(90). 이 질의(90)는 클라이언트(24)에 다시 전달되고 클라이언트(24)가 무슨 프로토콜을 접속하여 사용하고자 하는 지에 관한 상세한 내용으로 응답해야 함을 지시한다.

응답시, 클라이언트(24)는 프로토콜 패킷(92)의 세트를 전송하고, 이 세트의 각 패킷은 서버(34)로부터 요청되어지는 필수 또는 옵션의 프로토콜 모듈을 지정하는데 이용된다. 일실시예에서, 세트에서의 패킷의 수는 가변적이어서 하나의 패킷이 요청된 각 프로토콜에 대해 전송된다. 다른 실시예에서, 전송되어지는 패킷의 개수는 제1패킷의 헤더에 포함된다. 제3실시예에서, 전송되고자 하는 패킷의 잔류개수는 각 패킷의 헤더에 포함되고 각 패킷이 연속하여 전송됨에 따라 감소된다. 따라서, 클라이언트(24)는 예를 들어 암호화 및 데이터 압축이 사용될 수 있다는 것을 나타냄으로써 질의(90)에 응답할 수 있다. 이 경우에, 두개의 프로토콜 패킷은 클라이언트(24)로부터 서버(34)에 전송될 것이고, 일실시예에서, 제1패킷의 헤더는 패킷의 수를 2로 표시할 것이다.

일단 질의(90)에 대한 응답이 수신되면, 접속관리기(80)는 클라이언트노드(24)에 의해 요청된 프로토콜에 대응하는 프로토콜 드라이버(120, 120', 120'')를 이용하여 프로토콜 스택을 형성한다. 일실시예에서, 접속관리기(80)는 요청된 클라이언트 프로토콜에 대응하는 각각의 요구된 프로토콜 드라이버(120, 120', 120'')(예를 들어, 암호화가 클라이언트에게 요구되면 암호화 드라이버)를 프로토콜 스택 "컨테이너"에 배치시키고 이들을 서로 링크한다. 이 동적 프로세스는 서버(34)가 특정 클라이언트노드(24)용 이전 프로토콜 스택 설명을 구비할 것을 요구하지 않고 클라이언트노드(24)를 프로토콜 스택의 내용을 동적으로 지정하도록 한다. 이 방법을 사용하면, 각각의 클라이언트(24)가 관련된 통신 채널에 대해 광범위하게 상이한 요구사항을 구비하여도, 다수 클라이언트(24)는 단일 서버에 의해 처리될 수 있다. 예시된 실시예에서, 각 클라이언트(24, 24', 24'')는 개별적인 통신 프로토콜 스택(104, 104', 104'')과 관련된다. 이러한 동적으로 확장가능한 프로토콜 스택들은 하기에 더욱 상세히 설명되고 1995년 10월 11일에 출원된 미국 특허출원 제08/540,891호는 본 명세서에 참조 문헌으로 포함된다.

상술된 실시예에서, "컨테이너"(112)는 상표명 NT 디바이스 드라이버와 같은 사용자 레벨 또는 커널 레벨 디바이스 드라이버이다. 이 컨테이너드라이버는 클라이언트노드(24)의 프로토콜 요구사항에 대응하는 내적 프로토콜 모듈 또는 "드라이버"(일반적으로 120)에 적합한 보조 서포트를 제공한다. 이 보조 서포트는 예를 들어, 하나의 프로토콜 드라이버가 데이터를 다음 드라이버로 전송하는 데 협력하는 조력자 경로(helper routine)의 형태이다. 대안으로, 다른 실시예에서, 각 프로토콜 드라이버는 그 자체가 완전한 사용자-레벨 또는 커널-레벨이다.

이제, 도 3에 도시된 실시예를 참조하면, 통신관리기(60)는 두개의 주요 소프트웨어 모듈: ICASRV.EXE(90) 및 ICAAPI.DLL(94)을 포함한다. 예시된 실시예에서, ICASRV.EXE(90)은 클라이언트/서버 인터페이스의 서버측이다.

ICASRV.EXE(90)은 모든 통신 상태를 관리하고, 일실시예에서, WINDOWS NTTM 서비스로서 구현된다. 접속관리기(60)의 제2부분은 ICAAPL.DLL(94)이다. ICAAPL.DLL(94)는 클라이언트와 접속을 설정하고, 사용되는 프로토콜을 설정하고, 프로토콜 스택의 완성을 ICASRV.EXE(90)에 통지한다. 일실시예에서, 제3모듈CDMODEM.DLL(96)은 ICAAPL.DLL(94')에 링크된다. CDMODEM.DLL(96)은 ICAAPI.DLL(94')가 모뎀 디바이스와 통신하는데 사용하는 모듈이다.

상술된 접속 방법론은 웹브라우저 프로그램을 실행하는 클라이언트(24)용으로 사용될 수 있다. 이 설명의 목적을 위하여, 웹브라우저 프로그램을 실행하는 사용자는 "윈도우 사용자"(viewing user)로 언급될 것이다. 용어 "서버" 또는 "서버노드"는 HTML파일을 처리하는 머신 또는 실행될 수 있는 어플리케이션이라 칭하도록 사용될 것이다. 예를 들어, 윈도우 사용자는 클라이언트노드에서 웹브라우저를 실행하고 HTTP 프로토콜을 통해 파일 서버에 요청한다. 서버는 파일 데이터를 HTTP 프로토콜을 통해 클라이언트에게 전송함으로써 응답한다. 클라이언트상에서 동작하는 웹브라우저는 전송된 데이터를 수신하여 HTML 페이지로서 데이터를 윈도우 사용자에게 표시한다.

도 4를 참조하면, 서버(34')에 위치되고 본 발명의 실시예에 따라 제작된 HTML 파일(64)은 일반적 끼워넣은 윈도우태그(66)를 포함한다. 일반적 끼워넣은 윈도우태그(66)는 일반적 끼워넣은 윈도우(66')가 HTML 파일(64)에 의해 설명된 HTML 페이지(64') 내의 특정 위치에 표시되어야 하는 HTML 파일(64)을 표시하는 브라우저(60)를 지시하는 소정의 데

이타 구조이다. 일반적 끼워넣은 윈도우태그(66)는 윈도우 높이, 윈도우 폭, 윈도우 경계 스타일, 윈도우 내 배경 색상 및 형태, 어떤 어플리케이션이 윈도우에 표시될지, 얼마나 자주 출력표시가 갱신되어야하는지와 같은 추가 정보 또는 어플리케이션 출력의 디스플레이를 향상시키는 데 사용가능한 임의의 다른 추가정보를 포함할 수 있다.

HTML 파일에 끼워넣을 수 있는 총칭 끼워넣은 윈도우태그의 몇몇의 예는 하기와 같다.

상표명 ActiveX tag

```
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
```

```
data="/ica/direct.ica"CODEBASE="/cab/wfica.cab"
```

```
width=436height=295>
```

```
<param name="start"value="Auto">
```

```
<param name="Border"value="On">
```

```
</object>
```

상표명 Netscape Plugin tag

```
<embed src="http://www.citrix.com/ica/direct.ica"
```

```
pluginspage="http://www.citrix.com/plugin.html"
```

```
height=295 width=436 Start=Auto Border=On>
```

```
<embed>
```

상표명 JAVA tag

```
<applet code=JICA.class width=436 height=295>
```

```
<param name=Address value="128.4.1.64">
```

```
<param name=InitialProgram value=Microsoft Word 7.0>
```

```
<param name=Start value=Auto>
```

```
<param name=Border value=On>
```

```
</applet>
```

상기 각 경우에서, 태그는 295 픽셀의 높이와 436 픽셀의 폭을 갖는 윈도우는 어플리케이션 출력을 수신하도록 묘사되어야 한다는 것을 나타낸다. 각 태그는 또한 어플리케이션이 자동적으로 실행을 개시하여야 하고 어플리케이션 출력이 표시되는 윈도우가 경계선으로 묘사되어야 한다는 것을 명시한다. 상표명 액티브엑스와 상표명 넷스케이프 플러그인 태그는 "/ica." 디렉토리에 위치된 "direct.ica" 파일에 명시된 원격 어플리케이션 파라미터를 갖는다. 상표명 자바 태그는 원격 어플리케이션 파라미터를 직접 명시한다. 상기 예에서, 어플리케이션을 처리하는 서버의 어드레스 뿐만 아니라 실행되고자 하는 어플리케이션의 명(name)이 명시된다.

브라우저 어플리케이션(60)은 요청을 특정 URL(Uniform Resource Locator) 어드레스에 송신함으로써 HTML 파일(64)에 액세스한다. HTML 파일(64)을 처리하는 서버(34')는 HTML 파일(64) 데이터를 브라우저 어플리케이션(60)에 전송하

여, 텍스트를 디스플레이하고 HTML 파일(64)에 포함된 소정의 태그를 번역한다. 브라우저 어플리케이션(60)은 HTML 파일(64) 데이터를 HTML 페이지(64')로 디스플레이한다. 일반적 끼워넣은 윈도우 태그(66)가, 상술된 태그들 중 하나와 같이, HTML 파일(64) 내에 존재하면, 브라우저(60)는 표시된 HTML 페이지(64') 내에 블랭크 윈도우(66')를 묘사한다.

요구된 어플리케이션(62')의 실행은 HTML 페이지(64')의 디스플레이상에 즉시 시작할 수 있거나 또는 실행은 일부 신호(예를 들면, 어플리케이션(62')의 실행을 표시하는 특정 사용자 입력을 개시할 수 있는 신호)를 기다린다(wait). 일단, 어플리케이션(62')의 실행이 시작되면, 브라우저 어플리케이션(60)은 어플리케이션 윈도우(66')와 관련된 파라미터 핸들러(40)를 인스턴스화한다(instantiate). 파라미터 핸들러(40) 인스턴스(instance)는 브라우저 어플리케이션(60)의 자(child) 프로세스로서, 브라우저 어플리케이션(60)의 동위(peer) 프로세스로서, 또는 브라우저 어플리케이션(60)와 관련된 동적으로 링크된 라이브러리("DLL")로서 스핀(spawn)될 수 있다.

브라우저 어플리케이션(60)은 일반적 내장 윈도우 태그(66)에 의해 제공된 어플리케이션 윈도우(66')와 관련된 임의의 특정 파라미터를 파라미터 핸들러(40) 인스턴스로 전달한다. 부가적으로, 브라우저 어플리케이션(60)은 어플리케이션 윈도우(66')용 핸들러를 파라미터 핸들러(40) 인스턴스에 전달할 수 있거나 또는 파라미터 핸들러(40) 인스턴스는 브라우저 어플리케이션(60)을 어플리케이션 윈도우(66')용 핸들러를 탐색하도록 질의할 수 있다. 또한 파라미터 핸들러(40) 인스턴스는 네트워크 실행(50)을 야기한다. 네트워크 실행(50)은 파라미터 핸들러(40) 인스턴스의 자 프로세스로서 또는 파라미터 핸들러(40) 인스턴스의 동위 프로세스로서 스핀될 수 있다.

파라미터 핸들러(40) 인스턴스는 임의의 특정 어플리케이션 윈도우 파라미터(66')를 네트워크 실행부(50)로 포워드한다. 파라미터 핸들러(40) 인스턴스에 의해 명시되지 않은 파라미터 또는 일반적 내장 윈도우 태그(66)는 디폴트값으로 설정될 수 있다. 네트워크 실행부(50)는 하드-코드된 소정의 파라미터 디폴트를 가질 수 있거나 또는 네트워크 실행부(50)는 파라미터 디폴트를 포함하는 파일에 액세스할 수 있다.

네트워크 실행부(50)는 자체 어플리케이션 출력 윈도우(66')를 생성한다. 네트워크 실행부(50)는 디스플레이된 윈도우(66')의 자(child)로서 그 어플리케이션 출력 윈도우(66')를 생성하고 브라우저 어플리케이션(60)에 의해 묘사된 페어런트(parent)윈도우(66')를 통해 직접 어플리케이션 출력 윈도우(66')를 표시한다. 네트워크 실행부(50)에 의해 묘사된 어플리케이션 출력윈도우(66')가 브라우저 어플리케이션(60)에 의해 묘사된 어플리케이션 출력윈도우(66')의 자이므로, 어플리케이션 출력 윈도우(66")는 위치정보를 포함하는 그것의 페어런트의 다양한 특성을 이어받는다. 따라서, 어플리케이션 출력 윈도우(66")는 윈도우 사용자가 브라우저 어플리케이션(60)의 스크린을 스크롤하거나(scroll) 어플리케이션 윈도우(66')를 변환시키는 다른 행동을 실행할 때 어플리케이션 윈도우를 수행할 수 있다.

또한, 네트워크 실행부(50)는 서버(60)와 통신 채널을 설정하고 상술된 접속방법을 사용하여 서버(34")에 의해 요구된 어플리케이션(62')의 실행을 호출한다. 상기 설명에서 클라이언트로서 작용하는 네트워크 실행부(50)는 파라미터 핸들러(40) 인스턴스로부터 수신된 소정의 파라미터를 소정의 필요한 디폴트값과 함께 서버로 전달한다. 파라미터가 서버로 전달되지 않으면, 서버는 디폴트값을 갖지 않은 필요한 파라미터(예를 들어 "user id")가 있으면 파라미터를 요청할 수 있거나 또는 파라미터용 디폴트값, 예를 들어 실행 우선순위를 제공할 수 있다. 서버(34")는 요구된 어플리케이션 프로그램(62')의 실행을 개시하고 출력을 네트워크 실행부(50)로 향하게 한다. 네트워크 실행부(50)는 어플리케이션 프로그램(62')으로부터 데이터를 수신하고 출력 데이터를 그 어플리케이션 출력 윈도우(66")에 표시한다. 어플리케이션 출력 윈도우(66")는 브라우저 어플리케이션(60)에 의해 묘사된 어플리케이션 윈도우(66')의 상부에 묘사되므로, 어플리케이션 출력 데이터는 HTML 페이지(66')에 표시된다. 상술된 바와 같이, 네트워크 실행부(50)에 의해 묘사된 어플리케이션 출력 데이터는 브라우저 어플리케이션(60)에 의해 묘사된 어플리케이션 윈도우(66')의 자이다. 이것은 HTML 페이지(64')가 스크롤될 때 어플리케이션 출력 윈도우(66")가 스크롤하도록 허용한다.

또한 어플리케이션 출력 윈도우(66")는 입력을 윈도우 사용자로부터 수신한다. 비가공(raw) 입력데이터, 예를 들어 마우스 클릭은 네트워크 실행부(50)에 의해 어플리케이션 출력 윈도우(66")로 수신된다. 네트워크 실행부(50)는 비가공 입력 데이터를 서버(34") 상에서 실행하는 어플리케이션(62')으로 포워드시킨다. 이 방식에서, 윈도우 사용자는 HTML 페이지(64')를 통해 어플리케이션(62')과 상호작용할 수 있다.

도 5를 참조하면, 윈도우 사용자는 사용자 컴퓨터(14)의 스크린(18) 상에 어플리케이션 윈도우(66')를 갖는 HTML 페이지(64')를 디스플레이하도록 소위 "브라우저" 프로그램을 사용한다. 윈도우 사용자는 어플리케이션 프로그램(62')의 실행을 호출할 수 있다. 전형적으로, 이것은 "포인트-앤드-클릭" 인터페이스를 사용하는 사용자, 즉 윈도우 사용자가 윈도우 사용자 컴퓨터(14)의 스크린(18) 상에 디스플레이되는 커서(12)를 조작하도록 마우스(16)를 사용함으로써 수행된다. 일단, 커서(16)가 HTML 페이지(64')의 특정 부분 위에 있으면, 윈도우 사용자는 마우스(16) 상의 버튼(15)을 "클릭"함으로써 신호

를 보낼 수 있다. 대안으로, 윈도우 사용자는 또한, "리턴" 키와 같은 관련된 키보드(17) 상의 키를 누름으로써 신호를 보낼 수 있다. 다른 실시예에서, 윈도우 사용자는 마우스(16)를 전혀 사용하지 않지만, 대신에 터치패드, 트랙볼, 감압 태블릿 및 펜 또는 커서(12)를 조작하기 위한 다른 입력기구를 사용할 수 있다.

다른 실시예에서, 어플리케이션 윈도우(66') 또는 HTML 페이지(64')의 다른 부분은 "핫 존(hot zone)"으로 정의될 수 있다. 윈도우 사용자가 커서(12)를 "핫 존"으로 이동할 때, 서버(34') 상의 어플리케이션(62')의 실행이 개시된다.

일단 윈도우 사용자가 어플리케이션(62')의 실행이 시작되어야 한다는 것을 나타내면, 브라우저 어플리케이션(60)은 파라미터 핸들러(40)를 인스턴스화하고 일반 내장 윈도우 태그(66)에 의해 어플리케이션 윈도우(66')와 관련된 인스턴스 파라미터를 전달한다. 파라미터 핸들러(40) 인스턴스(instance)는 네트워크 실행부(50)를 스핀하고 어플리케이션 윈도우(66')의 파라미터를 네트워크 실행부(50)에 전달한다. 네트워크 실행부(50)는 어떤 어플리케이션(62')이 호출되어야 하는지와 어플리케이션(62')이 어떤 서버(34') 상에 존재하는지를 결정한다. 일반적으로, 이 정보는 브라우저 어플리케이션(60)으로부터 일반 윈도우 태그(66) 형태로 얻는 파라미터 핸들러(40) 인스턴스에 의해 전달되지만, 네트워크 실행부(50)는 가능하다면 요구되는 어플리케이션(62')을 어떤 서버들이 처리하는지를 결정하기 위하여 마스터 네트워크 정보노드(40) 또는 다른 다양한 서버에 질의할 필요가 있을 수도 있다. 그리고, 네트워크 실행부(50)는, 어플리케이션의 실행을 개시하고, 상술한 바와 같이, 어플리케이션 윈도우(66')에 어플리케이션 프로그램(62')의 출력을 디스플레이한다.

네트워크 실행부(50)는 윈도우 사용자가 어플리케이션(62')의 실행이, 예를 들어 어플리케이션 윈도우(66')를 단음으로써, 정지되어야 함을 지시하거나 또는 다른 HTML 페이지가 디스플레이되어야 하는 것을 지시하는 태크를 클릭할 때까지 어플리케이션 출력 윈도우(66')에 어플리케이션 출력을 직접 디스플레이한다. 이것이 발생할 때, 어플리케이션(62')의 실행이 끝날 수 있다. 그러나, 접속을 "캐시"(cache)하는 것이 바람직하다. 효과적으로, 제1파라미터 핸들러(40)에는 즉시 종료되지 않는다. 그러나, 어플리케이션(62')은 제1파라미터 핸들러(40)가 더 이상 "집중"(focus)되지 않기 때문에, 낮은 우선순위 레벨, 즉 "백그라운드" 모드에서 실행을 계속한다.

일반적으로, 파라미터 핸들러(40) 소스 코드에 인스턴스들을 등록하기 위해 전역적으로 액세스가능한 데이터 구조를 제공함으로써 접속 캐싱을 수행하는 것이 바람직하다. 예를 들어, 파라미터 핸들러(40)에는 전역적으로 액세스가능한 링크된 리스트 데이터 구조, 데이터 배열, 데이터 테이블 또는 다른 데이터 구조가 제공될 수 있다. 데이터 구조는 전체적으로 이용가능하기 때문에, 파라미터 핸들러(40)의 각 인스턴스는 데이터 구조를 관독하고 기록할 수 있다. 데이터구조가 전역적으로 사용가능하기 때문에, 파라미터 핸들러(40)의 각 인스턴스는 데이터구조를 읽고 쓸 수 있다. 이것은 파라미터 핸들러(40)의 각 인스턴스가 그 존재를 알리는 데이터 구조에 기록함으로써 모든 다른 인스턴스와 함께 "등록"하는 것을 허용한다.

어떠한 접속 정보도 저장되지 않는 다른 실시예의 경우에, 어느 순간에 캐시될 수 있는 접속의 수에 대한 선정된 제한이 설정될 수 있다. 이 실시예에서, 인스턴스들의 등록이 캐시된 접속의 과잉수를 야기시키면, "캐시된" 접속들 중 하나는 제거되어, 예를 들면 그 접속과 관련된 파라미터 핸들러(40)의 인스턴스에 그것이 종료되어야 함이 통지된다. 종료 전, 파라미터 핸들러(40)는 연관 네트워크 실행부(50)에 종료되어야함을 통지한다. 네트워크 실행부(50)는 어플리케이션 프로그램(62')을 처리하는 서버와의 세션(session)을 닫은 후 종료한다.

기타 정보가 저장되는 실시예에서, 추가 정보는 캐시된 접속을 더욱 효과적으로 관리하도록 사용될 수 있다. 예를 들어, 사용자가 미리정해진 시간, 예를 들면 10분 내에 HTML 페이지(64')를 능동적으로 고찰하지 않으면, 파라미터 핸들러(40) 인스턴스들은 종료하도록 지시되어, 처리 서버와의 세션은 종료되고, 파라미터 핸들러(40)는 레지스트리 내의 엔트리를 제거한다.

캐시된 접속 정보는 임의의 공지된 캐시 관리 계획을 사용하여 관리될 수 있다. 접속 엔트리는 "선입선출(first in, first out)" 이론에 근거하여 폐기되어, 예를 들면, 오래된 엔트리는 새로운 엔트리가 첨가될 때마다 폐기된다. 대안으로, 캐시된 접속 정보 엔트리는 "최근 저사용(least recently used)" 이론에 근거하여 폐기되어, 사용자에게 의해 최소량이 사용된 접속에 관련된 정보를 폐기한다. 랜덤 교체와 같은, 다른 캐시 관리 기술도 사용될 수 있다.

윈도우 사용자가 캐시된 접속을 갖는 이전 HTML 페이지(64')로 복귀하면, HTML 페이지(64')와 관련된 네트워크 실행부(50)는 전경(foreground)으로 복귀되어, "초점(focus)"을 다시 얻고, 관련된 어플리케이션의 진행이 보통 우선순위 레벨에서 시작한다. 필요하다면, 네트워크 실행부(50)는 어플리케이션(62')과의 접속을 재설정한다. 출력 데이터가 캐시된 접속에 대한 네트워크 실행부(50)에 의해 저장되지 않아도, 접속이 어플리케이션 윈도우(66')에 적합하게 재설정되자마자 어플리케이션(62')에 대한 접속은 재설정되고 어플리케이션(10)은 어플리케이션 윈도우(66')에 직접 재기록한다.

유사하게, 상술된 접속 방법이 해석언어로 기록된 어플리케이션의 원격 실행을 제공하도록 사용될 수 있다. 다시 도 2를 참조하면, 클라이언트노드(24)는 클라이언트노드(24)를 위하여 어플리케이션(62)을 실행하는 서버노드(34)에 접속된다. 이 예에서, 서버 어플리케이션(62)은 클라이언트노드(24)가 해석언어로 기록된 어플리케이션을 요청하도록 허용하는 어플리케이션이다. 예를 들어, 어플리케이션(62)은 클라이언트노드(24)가 URL 어드레스를 사용하는 상표명 자바 어플리케이션을 다운로드하도록 허용하는 웹브라우저일 수 있다. 언급된 바와 같이, 어플리케이션이 다운로드되는 노드와 서버노드(34)는 컴퓨터 네트워크에 의해 상호접속된 분리 머신이다. 그러나, 몇몇 실시예에서, 이들 머신은 동일한 하나일 수 있다.

클라이언트노드(24)가 클라이언트 메모리와 프로세서 사용의 양쪽면에서 보면 금지될 수 있는 다운로드된 어플리케이션을 저장하고 실행하도록 요구하는 것을 방지하기 위하여, 클라이언트노드(24)가 관련된 서버노드(34) 상의 실행환경(96)은 다운로드된 어플리케이션에 적합한 실행환경을 제공한다. 상기 실행환경은 다운로드된 어플리케이션의 바이트 스트림을 해석하여 어플리케이션을 나타내는 일련의 명령어를 생성한다. 어플리케이션이 자바 해석언어로 기록되어 있으면, 실행환경은 대체로 "가상 자바 머신"으로 칭해진다.

몇몇 실시예에서, 실행환경은 컴파일러를 포함한다. 이들 컴파일러는 어플리케이션의 바이트 스트림을 "고유(native)" 코드로 변환시킨다. 예를 들어, 컴파일러는 자바 어플리케이션 언어로 기록된 어플리케이션의 바이트 스트림을 80486 기계어로 변환시킨다. 해석언어 바이트 스트림을 고유 코드로 변환하는 것은 각 바이트가 실행시간에 해석되어 실행되어야 하는 경우보다 어플리케이션을 더 빠르게 실행하도록 허용한다. 그러나, 몇몇 컴파일러는 어플리케이션이 실행하는 동안 바이트 스트림을 컴파일할 수 있다. 이들 컴파일러는 때때로 "실시간"(just in time) 컴파일러로서 언급되고, 보통 컴파일된 코드의 정적 스트림을 생성하기 위하여 현재-실행 지시를 실행하기 전에 미리 정해진 수의 바이트를 고찰한다.

다운로드된 어플리케이션은 서버노드(24)에 의해 해석되고 실행되며, 어플리케이션의 출력은, 도 2와 관련해서 설명된 바와 같이, 클라이언트노드로 전송된다. 또한, 서버노드(34)는 클라이언트노드(24)로부터의 입력을 수신한다. 이것은 클라이언트노드(24)가 다운로드된 어플리케이션을 조절하거나 또는 입력을 어플리케이션에 제공하도록 허용한다. 서버노드(34)는 다운로드된 어플리케이션을 해석하고 실행시키도록 개별 실행환경을 설정한다. 이들 실시예에서, 다운로드된 어플리케이션과 관련된 실행환경은 그 출력을 믹스(mux; 121)로 향하게 할 것이다.

도 6을 참조하면, 어떤 클라이언트(24, 24', 24''), 또는 사실상 어플리케이션(63)을 갖는 서버(34)에 부착된 모든 클라이언트(일반적으로 24)는 다른 서버(34', 34'')일 수 있다는 것을 인식해야 한다. 이 방식에서, 어플리케이션(63)에 의해 전송된 데이터는 클라이언트노드(24)로 보내지기 전에 다른 서버로 보내진다. 이 방식에서, 어플리케이션(63)에 의해 전송된 데이터는 이 네트워크가 팽아울 때 증가된 다수의 클라이언트노드에 전송된다.

각 클라이언트(24)가 서버(34)와의 접속을 종료할 때, 각 클라이언트 프로토콜 스택(일반적으로 104)과 이에 관련된 최소 스택(일반적으로 107)은 파괴된다. 유사하게, 제1클라이언트 프로토콜 스택(104)과 관련된 최소 프로토콜 스택(일반적으로 106)도 파괴된다. 마지막 최소 스택(107)과 제2(및 부수)클라이언트 프로토콜 스택(104)이 종료될 때, 구성은 실행환경(96)과 관련된 제1클라이언트 통신프로토콜 스택(104)만을 갖는 초기의 것이다. 제2 및 이어지는 클라이언트 프로토콜 스택(104)이 모두 종료될 때까지, 더우기 제1클라이언트(24)가 더이상 존재하지 않을 지라도 제1클라이언트 프로토콜 스택(104)은 파괴되지 않는다.

도 2에 도시된 바와 같이, 각 실행환경(96)은 멀티플렉서(121, 121', 121'')를 통해 각 프로토콜 스택(104)과 통신한다. 이제 본 발명에 따른 도 6을 참조하면, 예를 들어 서버(34)로부터 데이터 전송을 새도우하거나 모니터하기 위하여 또는 주시시세 어플리케이션과 같이 동일 데이터가 다수의 클라이언트(일반적으로 24)에 거의 동시에 브로드캐스팅되거나 연쇄적으로 동시에 전송되는 특정 브로드캐스트 어플리케이션으로부터 데이터를 브로드캐스팅하기 위하여, 하나 이상의 클라이언트가 제1클라이언트(24)에 전송되려는 데이터를 수신하는 것이 가능하다.

이러한 경우에, 제1클라이언트(24)는 전술한 바와 같이 특정화된 어플리케이션(63)을 실행하고 그 데이터를 클라이언트(24)에 전송하도록 야기한다. 제2클라이언트(24')가 브로드캐스트 어플리케이션(63)에 대한 액세스를 요청할 때, 접속관리기(80)는 제1클라이언트(24)와 관련하여 이미 설명된 바와 같이, 제2클라이언트(24')에 적합한 프로토콜 스택(104')을 구축하기 시작한다. 그러나, 어플리케이션(63)은 브로드캐스트 어플리케이션이기 때문에, 접속관리기(80)는 추가 실행환경(96)을 개시할 필요가 없고 대신 브로드캐스트 어플리케이션(63)으로부터 제2클라이언트(24') 및 임의의 추가 클라이언트(24'')에 데이터를 전송하는 데 필요한 스텝을 취한다는 것을 인식한다.

먼저, 접속관리기(80)는 제1클라이언트(24)의 통신프로토콜 스택(104)과 관련되는 제1최소 통신프로토콜 스택(106)을 형성한다. 접속관리기(80)는 다음에 제2클라이언트(24')의 통신프로토콜 스택(104')과 관련되는 제2최소 통신프로토콜 스택(106')을 형성한다. 각 추가의 클라이언트(24")가 브로드캐스트 어플리케이션(63)으로 액세스를 요청함에 따라, 다른 최소 프로토콜 스택(106')은 제1클라이언트 프로토콜 스택(104)과 관련되어 생성되고 및 다른 최소 프로토콜 스택(107') 및 클라이언트 프로토콜 스택(104")은 각 신규 클라이언트(24")에 적합하게 생성된다. 제1클라이언트 프로토콜 스택(104) 및 제1클라이언트 프로토콜 스택(104)에 관련된 모든 최소 프로토콜 스택(106, 106')과, 클라이언트 프로토콜 스택(104', 104")의 각 쌍 및 각 추가의 클라이언트(24', 24")에 관련된 최소 프로토콜 스택(107, 107')은 멀티플렉서(121)를 통해 통신하게 된다.

멀티플렉서(121)가 단지 하나의 클라이언트(24)에 데이터를 향하게 하거나 또는 이로부터 데이터를 수신할 때, 멀티플렉서(121)는 단순 통과디바이스(pass-through device)로서 작용한다. 그러나, 단일 어플리케이션(63)으로부터 데이터를 수신하거나 또는 단일 어플리케이션(63)에 데이터를 전송하는 하나 이상의 클라이언트(24, 24', 24")가 있으면, 각 멀티플렉서(일반적으로 121)는 두개의 추가 구성을 취한다. 하나의 구성에서, 멀티플렉서(121')는 제1클라이언트 프로토콜 스택(104)과 이에 관련된 각각의 최소 통신프로토콜 스택(106, 106')에 어플리케이션 데이터를 전송하고 이로부터 데이터를 수신하도록 구성된다. 다른 구성에서, 멀티플렉서(121")는 최소 프로토콜 스택(107, 107')에 의해 수신된 데이터를 이와 관련된 클라이언트 프로토콜 스택(104', 104")에 각각 전송하도록 구성된다. 이 실시예에서, 멀티플렉서(121)는 각 클라이언트 프로토콜 스택(104, 104', 104")로부터 입력데이터를 직접 수신할 수 있다.

접속관리기(80)는 제1클라이언트(24)와 관련된 최소 프로토콜 스택(106, 106')과 제2클라이언트(24') 및 부속클라이언트(24")의 최소 프로토콜 스택(107, 107') 각각에 접속시키고, 멀티플렉서(121)가 출력을 어플리케이션(63)으로부터 제1클라이언트(24)의 통신프로토콜 스택(104) 및 이와 관련된 최소 프로토콜 스택(106, 106')에 전송하도록 지시한다. 멀티플렉서(121)는 또한 각 제2 및 후속의 클라이언트 최소 프로토콜 스택(107, 107')을 이와 관련된 클라이언트 프로토콜 스택(104, 104')에 개별적으로 접속시키도록 접속관리기(80)에 의해 지시된다. 그러므로, 제1클라이언트 프로토콜 스택(104)을 통해 제1클라이언트(24)로 전송된 데이터는 각각의 관련된 프로토콜 스택(104', 104")과 각각의 관련된 최소 프로토콜 스택(107, 107')을 통해 제1클라이언트(24), 제2클라이언트(24') 및 후속 클라이언트(24")와 관련된 최소 프로토콜 스택(106, 106')으로 전송된다. 일 실시예에서, 프로토콜 스택 컨테이너는 트렉수와 주어진 어플리케이션(63)에 관련된 프로토콜의 형태를 유지하는 데이터 구조를 포함한다.

도 7을 참조하면, 상술된 바와 같이, 하나의 서버(34)의 "클라이언트"가 다른 서버(34', 34") (단순화를 위해 단지 두개만 도시됨)인 것이 가능하다. 그리고, 제2서버(34', 34")는 데이터를 클라이언트(일반적으로 24) 또는 추가 서버에 전송한다. 본 실시예에서, 서버 프로토콜 스택(일반적으로 104)의 출력은 제2서버(34', 34")의 프로토콜 스택(107')에 접속된다. 그리고, 상술된 바와 같이, 데이터는 프로토콜 스택과 클라이언트(일반적으로 24)에 대한 출력 간에 전송된다. 본 방식에서, 데이터는 하나의 서버에 의해서 지지될 수 있는 것보다 많은 클라이언트에 팬아웃되고 분배될 수 있다.

본 발명은 특정의 바람직한 실시예를 참조하여 특별히 도시되어 설명되었지만, 본 발명은 당업자에 의해서 첨부된 청구범위에 의해 한정된 본 발명의 요지 및 사상으로부터 벗어남이 없이 형태 및 설명에서의 다양한 변경이 이루어질 수 있다는 것을 이해되어진다.

산업상 이용 가능성

삭제

(57) 청구의 범위

청구항 1.

삭제

청구항 2.

삭제

청구항 3.

삭제

청구항 4.

삭제

청구항 5.

삭제

청구항 6.

삭제

청구항 7.

삭제

청구항 8.

삭제

청구항 9.

삭제

청구항 10.

삭제

청구항 11.

삭제

청구항 12.

삭제

청구항 13.

HTML 페이지 내에 실행 어플리케이션을 디스플레이하기 위한 방법에 있어서,

(a) 어플리케이션의 실행을 알리기 위하여 클라이언트로부터 입력을 수신하는 단계와,

(b) 상기 실행 어플리케이션이 디스플레이될 윈도우의 파라미터를 결정하는 단계와,

(c) 상기 (b)단계에서 결정된 파라미터를 이용하여 상기 HTML 페이지에 대한 통신 채널을 형성하는 단계와,

(d) 서버상에서 실행되는 상기 어플리케이션의 출력을 상기 통신 채널을 통해 어플리케이션 윈도우에서 디스플레이하는 단계

를 포함하는 방법.

청구항 14.

제13항에 있어서, 상기 (a)단계는,

(a-a) 어플리케이션의 실행을 알리기 위한 적어도 하나의 HTML 태그를 포함하는 HTML 페이지를 클라이언트에 디스플레이하는 단계와;

(a-b) 사용자에게 의한 적어도 하나의 HTML 태그의 선택에 응답하여 서버상의 상기 어플리케이션의 실행을 야기하는 단계를 포함하는 방법.

청구항 15.

제13항에 있어서, 상기 (b)단계는 상기 실행 어플리케이션이 디스플레이될 윈도우의 파라미터를 결정하기 위하여 상기 HTML 페이지가 저장된 파일을 액세스하는 단계를 포함하는 방법.

청구항 16.

제15항에 있어서, 상기 (b)단계는 어플리케이션의 실행과 연관된 파라미터를 결정하기 위하여 HTML 페이지가 저장된 파일을 액세스하는 단계를 더 포함하는 방법.

청구항 17.

제13항에 있어서, 상기 (c)단계는 상기 실행 어플리케이션과 상기 HTML 페이지내의 어플리케이션 윈도우 간의 데이터 파이프(pipe)를 형성하는 단계를 포함하는 방법.

청구항 18.

HTML 페이지내에 실행 어플리케이션을 디스플레이하기 위한 장치에 있어서,

HTML 페이지내의 어플리케이션 실행 윈도우와 연관된 파라미터를 수신하는 파라미터 핸들러와,

상기 파라미터 핸들러로부터 파라미터를 수신하고, 어플리케이션이 서버상에서 실행을 시작하도록 야기하고, 상기 파라미터 핸들러로부터 수신된 상기 파라미터에 응답하여 상기 어플리케이션 실행 윈도우내에 상기 어플리케이션의 실행을 디스플레이하는 네트워크 실행부

를 포함하는 장치.

청구항 19.

제18항에 있어서, 상기 HTML 페이지내의 어플리케이션 실행 윈도우와 연관된 파라미터를 결정하기 위해 HTML 페이지를 저장하는 파일을 액세스하는 파라미터 핸들러를 더 포함하는 장치.

청구항 20.

제18항에 있어서, 상기 파라미터 핸들러로부터 상기 네트워크 실행부에 의하여 수신된 상기 파라미터 중 하나는 요청된 어플리케이션을 처리하는(host) 서버의 식별자인 장치.

청구항 21.

제18항에 있어서, 상기 네트워크 실행부는 요청된 상기 어플리케이션을 처리하는 서버와 통신채널을 설정하는 장치.

청구항 22.

HTML 페이지내에 실행 어플리케이션을 디스플레이하기 위한 컴퓨터 판독가능 코드 수단을 포함하는 컴퓨터 판독가능 기록매체에 있어서,

(a) HTML 페이지내에서 어플리케이션의 실행을 알리기 위하여 클라이언트로부터 입력을 수신하기 위한 컴퓨터 판독가능 코드 수단과,

(b) 상기 실행 어플리케이션이 디스플레이될 윈도우의 파라미터를 결정하기 위한 컴퓨터 판독가능 코드 수단과,

(c) 상기 (b)단계에서 결정된 파라미터를 이용하여 상기 HTML 페이지에 대한 통신 채널을 형성하기 위한 컴퓨터 판독가능 코드수단과,

(d) 서버상에서 실행되는 상기 어플리케이션의 출력을 상기 통신 채널을 통해 어플리케이션 윈도우에서 디스플레이하기 위한 컴퓨터 판독가능 코드수단

을 포함하는 컴퓨터 판독가능 기록매체.

청구항 23.

HTML 페이지내에 실행 어플리케이션을 디스플레이하기 위한 방법에 있어서,

(a) 제1 HTML 페이지내에서 어플리케이션의 실행을 알리기 위하여 클라이언트로부터 입력을 수신는 단계와,

(b) 상기 실행 어플리케이션이 디스플레이될 윈도우의 파라미터를 결정하는 단계와,

(c) 상기 (b)단계에서 결정된 파라미터를 이용하여 제1 HTML 페이지에 대한 통신 채널을 형성하는 단계와,

(d) 서버상에서 실행되는 상기 어플리케이션의 출력을 상기 통신 채널을 통해 어플리케이션 윈도우에서 디스플레이하는 단계와,

(e) 제2 HTML 페이지의 디스플레이를 알리기 위하여 클라이언트로부터 입력을 수신하는 단계와,

(f) 상기 제1 HTML 페이지와 연관된 상기 (b)단계에서 결정된 파라미터를 저장하는 단계

를 포함하는 방법.

청구항 24.

제23항에 있어서,

(g) 상기 제1 HTML 페이지를 재디스플레이(redisplay)하기 위하여 사용자로부터 입력을 수신하는 단계와,

(h) 상기 제1 HTML 페이지와 연관된 상기 (f)단계에서 저장된 파라미터를 검색(retrieving)하는 단계와,

(i) 검색된 연관 파라미터에 응답하여, 상기 실행 어플리케이션을 디스플레이하는 윈도우를 포함하는 상기 제1 HTML 페이지를 재디스플레이하는 단계

를 더 포함하는 방법.

청구항 25.

HTML 파일 내에 어플리케이션을 끼워넣기(embedding) 위한 시스템에 있어서,

어플리케이션 프로그램을 저장하고 실행하는 서버와,

어플리케이션 프로그램의 실행을 개시하기 위하여 커맨드(command)를 상기 서버에 전송하고, 상기 서버상에서 실행되는 어플리케이션 프로그램으로부터 출력을 수신하여, 어플리케이션 프로그램 출력을 전송하는 네트워크 실행부와,

파라미터를 수신하고, 상기 수신된 파라미터를 상기 네트워크 실행부에 전달(pass)하는 파라미터 핸들러와,

상기 파라미터 핸들러에 윈도우 파라미터를 전달하는 어플리케이션 윈도우를 포함하고 상기 네트워크 실행부로부터 어플리케이션 출력을 수신하는 HTML 파일

을 포함하는 시스템.

청구항 26.

제25항에 있어서, 상기 HTML 파일내에 끼워넣어질 때, 어플리케이션 윈도우가 디스플레이되도록 야기시키며, 상기 파라미터 핸들러에 윈도우 파라미터를 전달하며, 상기 네트워크 실행부로부터 어플리케이션 프로그램 출력을 수신하는 HTML 태그(tag)를 더 포함하는 시스템.

청구항 27.

삭제

청구항 28.

삭제

청구항 29.

삭제

청구항 30.

삭제

청구항 31.

삭제

청구항 32.

삭제

청구항 33.

삭제

청구항 34.

삭제

청구항 35.

삭제

청구항 36.
삭제

청구항 37.
삭제

청구항 38.
삭제

청구항 39.
삭제

청구항 40.
삭제

청구항 41.
삭제

청구항 42.
삭제

청구항 43.
삭제

청구항 44.
삭제

청구항 45.
삭제

청구항 46.
삭제

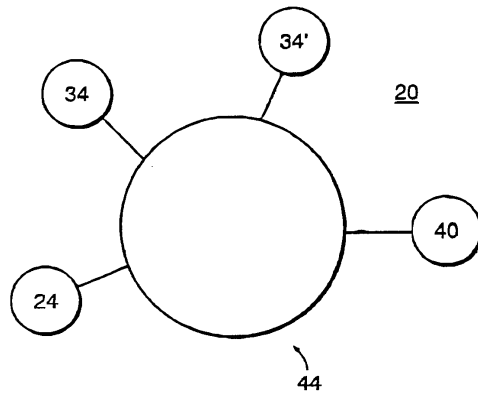
청구항 47.
삭제

청구항 48.
삭제

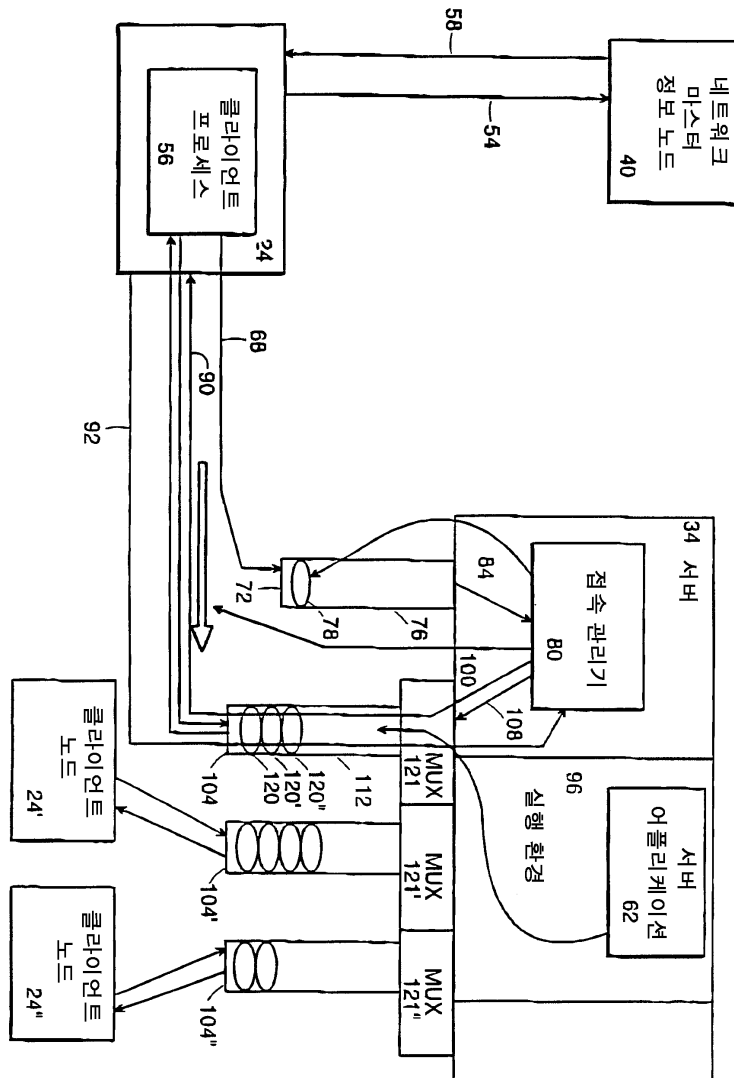
청구항 49.
삭제

도면

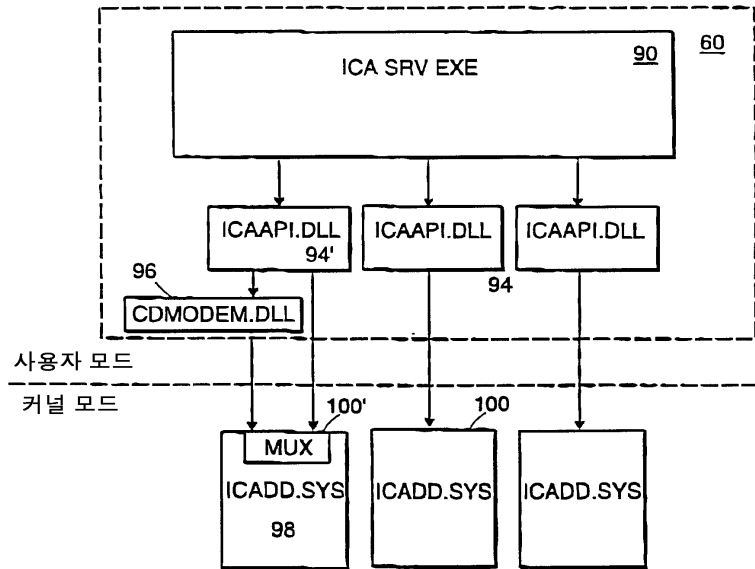
도면1



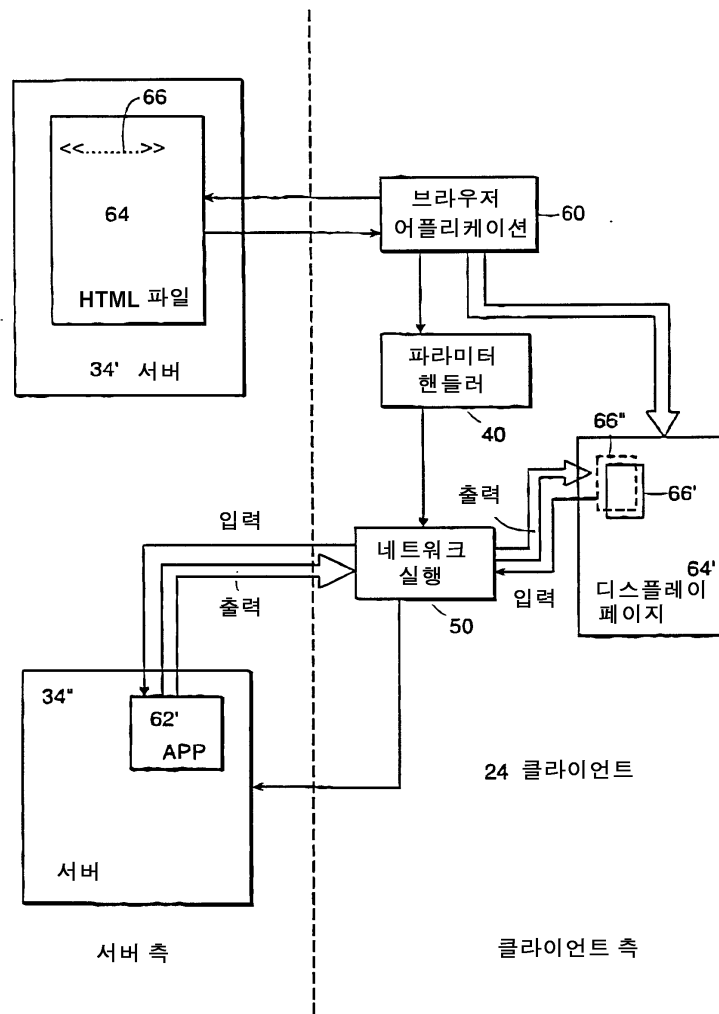
도면2



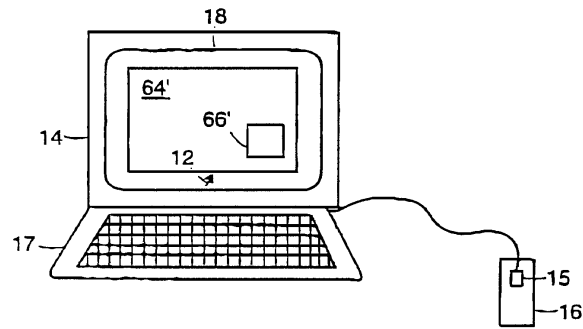
도면3



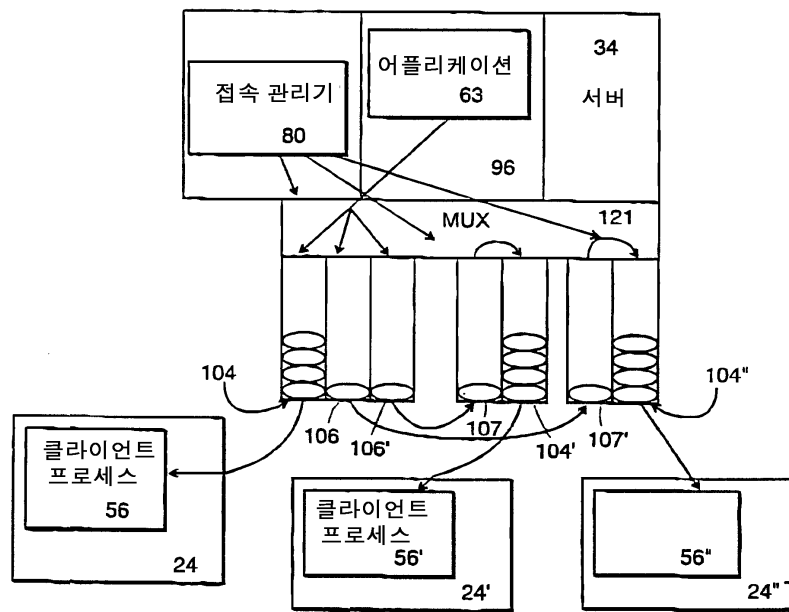
도면4



도면5



도면6



도면7

