(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[6]
G06F 15/16

(45) 공고일자 2003년10월04일
(11) 등록번호 10-0387207
(24) 등록일자 2003년05월29일

(21) 출원번호	10-1996-0705125	(65) 공개번호	특1997-0701883
(22) 출원일자	1996년09월16일	(43) 공개일자	1997년04월12일
번역문제출일자	1996년09월16일		
(86) 국제출원번호	PCT/US1995/03183	(87) 국제공개번호	WO 1995/25311
(86) 국제출원일자	1995년03월15일	(87) 국제공개일자	1995년09월21일
(81) 지정국	국내특허 : 오스트레일리아 바베이도스 불가리아 브라질 캐나다 헝가리 일본 대한민국 스리랑카 마다가스카르 노르웨이 루마니아 폴란드 체코 몽고 뉴질랜드 슬로바키아 우크라이나 베트남 리투아니아 기네 중국 라트비아 AP ARIPO특허 : 말라위 수단 우간다 EA 유라시아특허 : 벨라루스 카자흐스탄 러시아 타지키스탄 아르메니아 EP 유럽특허 : 오스트리아 스위스 리히텐슈타인 독일 덴마크 스페인 핀랜드 영국 룩셈부르크 모나코 네덜란드 스웨덴 포르투칼 OA OAPI특허 : 니제르		

(30) 우선권주장 08/213,197 1994년03월15일 미국(US)

(73) 특허권자 디지 인터내셔날 인크.
미합중국 미네소타 미네통카 브렌 로드 이스트 11001
(72) 발명자 올손, 진, 에이치.
미합중국, 55419 엠엔, 미네아폴리스, 알드리치 아베뉴 사우스 5131
(74) 대리인 김용인, 심창섭

심사관 : 마정윤

(54) 원격네트워크장치에의한통신시스템및통신방법

명세서

기술분야

<1> 본 발명은 범용 네트워크를 통한 컴퓨터 및 컴퓨터 주변장치와 같은 디지털 장치간의 통신분야에 관한 것이다.

배경기술

<2> 상호의 디지털 컴퓨터간 및 디지털 컴퓨터와 이와 서로 떨어진 컴퓨터 주변 장치간의 통신에 대한 필요성이 오랫동안 절실하였다. 이와 같은 필요성에 부응하기 위해, 아날로그 전화 네트워크를 통한 장거리 연결 구축용 모뎀의 사용이나 컴퓨터와 그 주변기기를 이용하여 데이터 전송 매체를 통한 통신을 가능케 하는 네트워크를 만드는 것을 포함하는 많은 기술들이 개발되었다.

<3> 두 대의 컴퓨터간 통신을 구축하는 데에는 한쌍의 모뎀이 사용되는데, 각각의 모뎀은 아날로그 전화 네트워크에 연결되어 있을 뿐만 아니라 그의 로컬 컴퓨터의 시리얼 포트에 부착되어 있게 된다. 시리얼 포트는 컴퓨터가 모뎀이나 프린터와 같은 주변기기로 정보의 연속적인 비트의 흐름을 통과할 수 있도록 하는 컴퓨터 상의 표준 인터페이스 포트를 말하는 것이다. 컴퓨터나 그 밖의 다른 멀티유저 컴퓨터 시스템을 토대로하는 유닉스(UNIX)상에서, 이와 같은 시리얼 포트는 사용자가 컴퓨터간 통신을 할 수 있도록 하는 단말기와의 통신에 사용된다. 결과적으로, 서로 이격된 단말기간에 전화선을 통하여 멀티유저 컴퓨터 시스템으로 통신을 하는데에 모뎀을 사용하는 것은 비교적 간단한 문제다.

<4> 현대의 멀티유저 컴퓨터는 다수의 등시 사용자들을 지원할 수 있기 때문에, 단말기로 컴퓨터와 통신할 수 있도록 하는 많은 시리얼 포트를 갖추는 것이 바람직하다. 그러나, 멀티유저 컴퓨터는 50대 이상의 단말기가 연결되게 지원할수 있으므로, 컴퓨터에 직접 많은 시리얼 포트를 모두 장착시키기는 것은 논리적으로 어렵다. 결과적으로, 다수의 내부적으로 기준이 되는 시리얼 포트로부터의 연속 트래픽을 하나의 통신 채널로 결합시키는데에 다중화 장치가 이용되어 왔다. 이와 같은 방식에서, 컴퓨터에는 오직 한 개의 통신 케이블만 연결되는 것이 필요하게 된다. 통신 케이블의 타단에는 다른 다중화 장치가 있고, 이장치가 결합된 데이터 트래픽을 개별적인 시리얼 포트를 위한 데이터로 분리시켜서, 이 시리얼 포트로 보내고, 이것에 의하여 그 데이터를 엑세스 할수 있다. 소위, 포트 컨센트레이터라 불리는 이같은 방식의 다중화 장치는 컴퓨터와 떨어져 위치해 있는 단말기들에게 다수의 시리얼 포트를 제공하기 위해 사용될 수 있다. 또한, 메인 통신 케이블 상의 데이터를 모뎀을 사용하여 전화선을 통해 전달하는 시스템에서 다중 전송이 이용될 수 있다. 이것은, 다수의 시리얼 포트가 컴퓨터로부터 극히 멀리 떨어진 곳에서 엑세스 할수 있게 하여준다.

<5> 네트워크 기술은 또한 어떤 컴퓨터와 다른 컴퓨터 또는 주변기기간의 원격 통신을 가능케 하기 위하여 사용될수 있다. 비교적 좁은 영역 내에 있는 컴퓨터와 주변기기를 연결하는 네트워크를 근거리 네트워크(Local Area Network) 또는 LANS라고 한다. 범용의 네트워크는 표준 하드웨어와 표준 통신 프로토콜을 사용하고 멀티벤더(mtlti-vendor) 환경에서 작동하는 통신 시스템이다. 범용의 통신용 하드웨어에는 이더넷(Ethernet)과 토큰 링(Token Ring)과 같은 LAN 테크놀로지를 포함한다. 표준 통신 프로토콜에는 TCP/IP와 SPX/IP를 포함한다.

<6> 범용 LAN은, 일반적으로 전송매체를 통하여 컴퓨터와 주변기기들을 접속함으로써 형성되고 표준 통신 프로토콜에 따라 매체를 통해 통신이 행하여진다. 이들 프로토콜들은 데이터가 어떻게 포맷되어야하고 데이터가 특정 장치를 위해 어떻게 명명되어야 하는지에 대하여 필요조건을 명확하게 한다. 이들 프로토콜에 맞지 않는 데이터는 네트워크를 통해 전송할 수 없게된다.

<7> 시스템을 작동시키는 호스트의 응용 프로그램들은 일반적으로 호스트 작동 시스템에 의해 제공한 로컬 통신 장치용 응용 프로그램 인터페이스(API)에 크게 의존한다. 이것은 유닉스의 종래 사용자 인터페이스가 단말기 커맨드 라인 조작이기 때문에 유닉스와 같은 시스템상에서 특히 중요하다.

<8> 범용 네트워크의 대부분이 비접속의 신뢰성이 없는 데이터그램 서비스(Connectionless unreliable datagram service)와 신뢰할 수 있는 쌍방향 바이트스트림 접속의 두 가지 통신 방식을 지원하게 된다. 이 두 가지 서비스의 API는 시리얼 포트 API와는 아주 다르고, 일반적으로, TTY로 알려진 시리얼 장치상에서 실행하기 위해 쓰여진 프로그램들은 정확하게 실행하도록 적합층(adaptation layer)이 필요하다.

<9> 유닉스 환경에서, 일반적으로 이것은 의사-tty(pseudo-tty) 또는 PTTY로 불리는 가상 TTY 장치를 사용하는 통신 프로토콜인 Telnet와 Rlogin으로 행하여진다. PTTY 장치는 하드웨어가 아니라, 대부분의 커맨드 라인 프로그램들을 충분히 실행시킬 수 있도록 유닉스 TTY와 에뮬레이션 할 수 있는 소프트웨어이다.

<10> 이들 PTTY와, 실제 시리얼 포트 하드웨어와 연결되어 이를 통제하는 진짜 TTY를 구별하는 것은 중요하다. PTTY는 슬레이브부와 마스터부를 함께 갖는다. PTTY의 슬레이브부는 TTY 에뮬레이션 장치로서, 시리얼 포트 API를 작동시키도록 하는 프로그램으로 사용된다. 마스터부는 네트워크를 통하여 데이터를 전송할수 있는 Telnet나 Rlogin과 같은 통신 프로그램들과 연결된다.

<11> PTTY의 슬레이브부는, 라인 편집 문자나 텝 확장 그리고 로컬 작동 시스템의 라인 규칙으로 실행되는 그 밖의 기능들과 같은 사용자 옵션을 바꾸라는 명령에 응하지만, 자료처리 속도(baud rate)나 문자의 크기, 입력된 에러에 대한 대응 그리고 BREAK 신호등을 바꾸라는 명령은 무시한다.

<12> PTTY의 마스터부는 판독되고 쓰여지며 그리고 데이터가 슬레이브와 마스터장치간에서 전후로 통과하는 방법을 변경하기 위한 아주 한정된 입출력 제어 세트(IOCTLS)를 받아드린다. PTTY의 마스터부에 쓰여진 데이터는 시리얼 포트에서 수신되었지만, PTTY의 슬레이브부상에서 수신된 시리얼 데이터로써 나타나며, 작동시스템의 라인 규칙에 의하여 입력된 시리얼 데이터로서 처리된다. 그때 데이터는 PTTY의 슬레이브부로부터 판독될 때 수신된 데이터로서 나타난다. 슬레이브의 PTTY에 쓰여진 데이터는 마찬가지로, 유닉스 라인 규칙에 의하여 시리얼 출력 데이터로써 처리되고, 그후 이 출력 데이터는 마스터부에서 판독될수 있다. PTTY의 슬레이브부에서 행해진 IOCTL 조작은 마스터부에서는 볼수 없다. 그래서 PTTY의 마스터부를 엑세스하는 프로그램에서는 이들의 호출이 된것인가를 검출할 수가 없다. 인터페이스는 로컬 조작 시스템의 라인 규칙으로는 실행될수 없는 몇 개의 조작을 지원하는데 충분하지 않는다.

<13> 대부분의 실행에서, 물리적 장치와 PTTY들과의 사이에 1대1의 대응이 없다는 것을 유념하여야 한다. 접속이 Telnet 또는 Rlogin에 의하여 유닉스 시스템으로 이루어질 때, 이들 프로그램은 간단히 세션(Sesson)을 위한 다음의 이용할수 있는 PTTY를 할당한다. 그래서 세션으로부터 세션으로 네트워크의 사용시는 다른 PTTY 장치를 명백한 패턴없이 얻을수 있다. 이것은 의사 TTY 장치에 근거하여 시스템의 안전화 혹은 사용자 옵션을 어렵게 하거나 불가능하게 하여준다. 어느정도까지 이것을 극복하기 이하여 특별한 프로그램이 특별 PTTY의 마스터부에 소속하는 터미널 서버 벤더에 의하여 보통 쓰여진다. 이 PTTY의 슬레이브부는 그 프로그램에 소속될 때까지이며, 다른 사용에 주기적으로 재할당되지 않는다. 이와 같은 프로그램들은 접속을 받아드리고 Rlogin 프로그램을 실행시키는 데에 사용할 수 있지만, 대부분은 다이얼 아웃 모뎀이나 프린터 또는 직접 접속 시리얼 장치들을 시뮬레이션하는 데에 쓰인다. 이사용은, PTTY의 이용을 Telnet나 Rlogin 이외에 다른 사용으로 확장하지만, 진짜 TTY 장치의 제한된 에뮬레이션으로 말미암아 이들의 변화된 작업환경이 필요하고 지속적인 정비와 양립성의 문제를 안고 있다.

<14> Telnet는 또한 포트 데이터 흐름 제어를 위해 TCP/IP 흐름 제어를 사용하고 명령을 데이터 다음 순서로 보내기 때문에 데이터스트림의 선두 데이터가 처리될 때까지 명령이 처리되지 않게 된다. 그결과로써, Telnet는 사용자가 요구하는 중단과 같은 어떤 명령을 전송하기 위해 TCP/IP의 촉된데이터 특성의 사용을 요구한다. 이와 같은 실행은, 사용자의 중단이 처리되기 전에 사용자가 중단한 선두 데이터를 폐기되는 것을 요구한다.

<15> 그러나, 이러한 실행은 네트워크 자체뿐만 아니라 호스트 컴퓨터에도 커다란 부담을 주게 된다. 예를 들면, 네트워크에 접속된 원격 단말기에 입력할 때 각각의 키스트로크가 처리되어 네트워크를 통해 호스트 컴퓨터로 보내지고 네트워크를 통해 다시 원격 단말기로 되돌아오게 된다. 이더넷(Ethernet) 네트워크 상에서, 전송되는 문자는 64 바이트 최소 크기의 메세지 패킷으로 보내야 하고 이를 승인하는 데에는 64 바이트의 메세지 패킷이 필요하다. 그러므로 입력된 각 문자당 두 개의 64 바이트 패킷이 전송되어야 한다.

<16> 뿐만 아니라, 호스트 컴퓨터가 문자를 받을 때에는, 문자를 받는 데에 PTTY 제어 소프트웨어를 실행시키는 태스크 스위치(task switc)가 필요하고 다시 반향 문자를 보내는 데에 PTTY 제어 소프트웨어를 실행하기 위한 태스크 스위치가 필요하게 된다.

<17> 단말기 서버로 호칭되는 장치는 네트워크의 어떤 컴퓨터로 로그(log)되는 것을 사용자에게 요구

함이 없이, 시리얼 포트를 통하여 사용자에게 리모트로긴(Remote Rogin), TELNET 및 RLOGIN 명령을 이용하는 것을 가능하게 한다. 터미널 서버상에서 시리얼 포트에 접속되는 사용자는, 네트워크의 멀티 유저 컴퓨터에 접속하기 위하여 RLOGIN 혹을 TELNET를 사용할 수 있다.

<18> 터미널 서버의 제 2 사용자는 또한 같은 멀티유저의 컴퓨터에 접속가능하고 이것에 의하여 그것에 할당된 그 자신의 PTTY와 제 2 통신 링크를 확립할 수 있다. TELNET, RLOGIN 및 마찬가지의 수순을 통하여 터미널 서버에 의하여 만들어진 접속은 네트워크가 다른 컴퓨터를 통하여 그의 수순을 사용하게된 접속과 똑같이 불이익이 생긴다.

발명의 상세한 설명

<19> 본발명의 목적은 상술한 종래기술의 불이익들을 극복하기 위한 것으로써, 범용 네트워크의 원격 터미널 서버에 배치된 비동기 포트에 호스트 컴퓨터를 엑세스시키는 통신 시스템을 제공하기 위한 것이다.

<20> 본발명의 다른 목적은 독특한 장치 드라이버 인터페이스를 사용하고 또한 통신 프로토콜을 다중화함으로써 서버가 단일의 접속으로 멀티포트를 위한 데이터 및 제어 명령을 송신하는 것이 가능한 통신 시스템을 제공하기 위한 것이다.

<21> 본발명의 또하나의 다른 목적은 하드웨어에서 구현되어 기존의 호스트 컴퓨터 소프트웨어와의 양립성을 증진시키고 또한 호스트 컴퓨터의 오버헤드를 감소시키는 통신 시스템을 제공하기 위한 것이다.

<22> 본발명의 또하나의 다른 목적은 호스트 컴퓨터에서 실행하는 응용프로그램이 서버의 통신 포트들중 하나의 완전제어를 허용하는 응용프로그래밍 인터페이스를 구비하는 하드웨어 장치를 제공함에 있다.

<23> 이와 같은 목적을 달성하기 위한 본발명의 통신 시스템은 복수의 통신 포트를 가지는 서버; 그리고, 네트워크 접속을 통하여 서버로 호스트 컴퓨터를 통신가능하게 접속하는 드라이버를 가지는 호스트 컴퓨터를 구비하고, 상기 드라이버는 서버의 통신 포트의 각각을 위한 상응하는 로컬 통신 포트를 규정함으로써 서버의 통신포트를 에뮬레이트하며, 또한 상기 드라이버는 서버의 통신포트가 호스트 컴퓨터에 직접 연결된 것처럼, 하드웨어 및 소프트웨어 흐름제어를 포함하여, 호스트 컴퓨터상에서 실행하는 응용프로그램이 서버의 통신 포트들중 하나의 완전 제어 (full controll)를 허용하는 응용프로그래밍 인터페이스(API)를 포함함을 특징으로 한다.

<24> 본발명의 또하나의 일면에 의한 본발명의 하드웨어 장치는 네트워크 접속을 매개하여 호스트 컴퓨터에 통신 가능하게 결합된 원격서버의 복수의 통신포트를 에뮬레이터하는 드라이버를 포함하고 상기 드라이버는 서버의 각 통신 포트를 위한 상응하는 로컬 통신포트를 규정하고, 서버의 통신포트가 호스트 컴퓨터에 직접 연결된 것처럼, 하드웨어와 소프트웨어를 포함하여, 호스트 컴퓨터에서 실행하는 응용 프로그램이 서버의 통신포트들중 하나의 완전제어를 허용하는 응용프로그래밍 인터페이스(API)를 포함함을 특징으로 한다.

<25> 본발명은 범용 네트워크의 원격 터미널 서버에 배치된 비동기 포트에 호스트 컴퓨터를 엑세스 시킴으로써 종래 기술의 불이익을 극복하는 통신 시스템으로 된다. 독특한 장치 드라이버 인테페이스를 사용하고 또한 통신 프로토콜을 다중화함으로써 서버는 단일의 접속으로 멀티포트를 위한 데이터 및 제어 명령은 송신하는 것이 가능하다. 이것에 의하여 네트워크상의 요구를 적게한다. 또한 본발명은 로컬 통신 포트의 모든 특성을 가지는 네트워크 상에 진짜의 TTY장치를 제공한다. 본발명은 모뎀이나 프린터와 같은 공용장치에 올바른 엑세스를 하면서 네트워크에 배치된 다수의 포스트 컴퓨터에 서버상의 포트를 엑세스하는 것을 가능하게 한다. 결국, 본발명은 하드웨어에서 구현되어 기존의 호스트 컴퓨터 소프트웨어와의 양립성을 증진시키고, 또한 호스트 컴퓨터의 오버헤드(over head)를 감소시킨다. 본발명은 원격 통신 포트나 그와같은 포트를 합체하는 서버에 엑세스하거나, 혹은 엑세스를 하여주는 것을 희망하는 클라이언트(통상 호스트 컴퓨터)간의 범용 네트워크에 의해 공급되는 신뢰성이 있는 통신 링크상에서 동작한다. 본발명은 범용 네트워크상의 진짜의 TTY 장치가 구비된 종래의 PTTY 구현과는 다르다. PTTY 구현과는 다르게, 본발명의 TTY는 실제 하드웨어와 1대1 대응이며, 시리얼 포트가 호스트 동작시스템에 직접 접속되어 이 사용에 전념하고 있는 것처럼 호스트 동작 시스템으로 그 하드웨어를 제어시킬수 있다.

<26> 이것은 응용 프로그램에 의해서 되는 가정과 저촉되지 않으며, 본 발명은 이런 제한을 완화하고, 네트워크 상에서 다른 호스트와 공유하여 이들 포트의 기능을 넓게 하며, PBX-형 추적 그룹과 동적 포트의 할당을 공용한다. 본 발명은 각 동작 시스템이 진짜의 TTY와 일 대 일 대응 관계를 유지하도록 하나, 본발명의 동작 시스템에 사용되고 있지 않을때 이와 같은 TTY가 다른 목적에 대하여 엑세스 될 수 있도록 하는 것은 임의이다.

<27> 서버상에서 1 혹은 2이상의 포트를 사용하고 싶은 클라이언트는 본발명의 「통신 프로토콜」 에 의하여 준비된 프로토콜을 사용하는 그의 서버에 단일의 접속을 오픈한다.

<28> 클라이언트와 통신 프로토콜을 사용하는 서버 사이에 모든 통신은 아무리 무수한 포트가 클라이언트에 의해서 사용될지라도 그의 단일 접속을 사용하여 진행된다. Telnet 및 Rlogin 과 같은 종래 기술 프로토콜하에서, 서버와 클라이언트 사이의 부가적인 접속이 오픈될 수 있다. 그러나, 이런 종래 기술의 프로토콜 접속의 존재는 본 발명의 통신 프로토콜의 제어하에서 포트에 영향을 주지 않는다. 클라이언트와 서버 사이의 접속이 어떤 이유로 끊겼을 때, 통신 프로토콜 제어 하에서, 모든 포트는 닫히고, 클라이언트에 의해서 요구되는 모든 자원은 복구된다.

<29> 통신 프로토콜 접속의 존재는 서버상의 모든 포트 자원에 클라이언트가 엑세스를 자동으로 허용하지 않는다. 클라이언트가 포트에 엑세스하기 전에, 클라이언트는 계속하여 포트를 오픈해야 하고, 그다음에, 클라이언트가 그포트를 닫힐 때까지 그 클라이언트는 그 포트에 배타적인 엑세스를 허용받는다.

<30> 일단 포트가 오픈되면 클라이언트는 전송속도를 변화하고, 모뎀 설정을 제어하기 위한 요청을 보낼 수 있다. 클라이언트는 데이터를 포트외부로 전송할 수 있으며, 서버는 자동으로 클라이언트에게 포트

에 수신된 모든 데이터를 전송한다.

<31> 통신 프로토콜 하에서, 클라이언트는 서버로부터 상태(status) 갱신을 요청할수 있다.

<32> 갱신은 요청 즉시 이루어질 수 있으며, 클라이언트는 클라이언트의 여러 가지 상태의 상황이 변할때마다 그 변화를 알려주도록 서버에게 요청할 수 있다.

도면의 간단한 설명

<33> 도 1은 본 발명의 서버와 다양한 호스트 컴퓨터를 접속하는 범용 컴퓨터 네트워크의 예시도.

<34> 도 2는 서버의 중요 부품을 포함하여 본 발명의 서버에 대해 범용 네트워크를 부착한 클라이언트 컴퓨터의 예시도.

<35> 도 3은 연속 공간내의 위치로 향해 지정되는 단일의 시퀀스 넘버를 가지는 순환 연속공간의 예시도.

<36> 도 4는 연속 공간내의 위치로 향해 지정되는 2개의 시퀀스 넘버를 가지는 순환 연속 공간의 예시도.

<37> 도 5는 서버를 통한 클라이언트로부터 포트로 전송되는 데이터 흐름을 서버가 제어하는 처리도.

<38> 도 6은 클라이언트가 서버를 통한 클라이언트로부터 포트로 전송되는 데이터의 흐름을 제어하는 처리도.

<39> 도 7은 서버를 통한 포트로부터 클라이언트로 전송되는 데이터의 흐름을 서버가 제어하는 처리도.

<40> 도 8은 클라이언트 드라이버의 중요 부품을 포함하여 본 발명의 서버에 대해 범용 네트워크를 부착하는 클라이언트 컴퓨터의 예시도.

실시예

<41> 도 1은 특정장소에서 사용되는 신규한 PC의 네트워크(16)와 디지털 이퀍먼트사의 백스(VAX) 컴퓨터 (14)와 유닉스(UNIX) 컴퓨터(12)를 포함한 다양한 컴퓨터를 상호 접속하는 범용 네트워크(10)를 도시한 것이다.

<42> 네트위크(10)는 에러 프리형 바이트스트림의 데이터 전송용에 설치된 프로토콜이라면, TCP/IP, SPX/IPX 및 X.25와 같은 다양한 표준 네트워크 통신 프로토콜중 어느 것을 이용할 수 있다.

<43> 통신 프로토콜은 이더넷(Ethernet) 또는 토큰 링(Token Ring)과 같은 특정한 논리위상 수학상에서 그 기능을 수행한다.

<44> 도 1에서 도시된 컴퓨터와 네트워크를 접속하는 통상 네트워크는 이더넷의 상부에서 동작하는 TCP/IP 프로토콜이 사용된다.

<45> 프로토콜의 TCP 부분은 바이트스트림 데이터를 취하고 그것을 패킷(PACKET)으로 변환한다. IP부분은 패킷을 취하여 데이터그램을 형성하고, 반면에 이더넷(Ethernet)은 데이터그램을 취하여 프레임을 형성한다. 비록 본 발명 그 자체가 이용된 네트워크 프로토콜에 영향이 없지만 전형적인 구성은 본 발명을 설명할 목적으로 이용될 것이다.

<46> 본 발명의 서버(20)는 네트워크(10)의 여러 개 포트(도시생략)를 접속하도록 작동한다.

<47> 통신라인(22)은 서버(20)상의 포트에 접속될 수 있다. 비록 단지 5개 통신 라인(22)이 제 1도에 도시되었을지라도, 서버(20)는 네트워크(10)에 접속될 수 있는 16개 이상 포트를 가진다.

<48> 도 2는 범용 네트워크(10)를 통하여 접속되는 서버(20)와 하나 이상의 클라이언트(18) 사이의 주 통신 소자를 도시한 것이다.

<49> 클라이언트(18)는 유닉스컴퓨터(12) 또는 백스컴퓨터(14) 또는 심지어 근거리 네트워크(16)와 같은 호스트 컴퓨터가 될 수 있다.

<50> 서버(20)는 서버의 네트워크 인터페이스(30)를 통해 네트워크(10)에 접속된다. 네트워크 인터페이스(30)는 서버(20)가 네트워크(10)의 어떤 다른 장치와 동일한 방법으로 네트워크를 통하여 통신하도록 필요한 산업용의 표준 하드웨어 및 소프트웨어를 제어를 제공하여준다. 모든 산업용의 표준 네트워크 인터페이스와 마찬가지로 네트워크 인터페이스(30)는 네트워크(10)로부터 데이터 프레임을 수신하여서 데이터가 서버(20)에 어드레스 되는지를 결정하고 만일 어드레스 된다면, 그의 네트워크를 제거하여 프레임, 데이터그램 및 패킷을 포맷팅하고 재조립함으로써 바이트스트림에 이용할수 있는 형식의 데이터를 만든다. 즉, 본 발명의 네트워크 인터페이스(30)는 전송, 네트워크, 링크의 기능과 국제표준기구 오픈시스템 상호 연결(ISO/OSI)모델의 물리적 계층을 포함한다.

<51> 네트워크 인터페이스(30)에 의해서 제공되는 바이트스트림 데이터는 본 발명의 통신 프로토콜의 실행을 상호 조정하는 중앙처리장치(CPU)로 공급된다. 통신 프로토콜의 실행에 필요한 프로그램 모듈은 서버 메모리(34)의 프로그램 메모리부(36)내에 저장된다.

<52> CPU(32)가 네트워크 인터페이스(30)로부터 수신된 데이터를 분석한 후, 미처리 데이터는 서버(20)의 적당한 포트(40)로 통과된다.

<53> 서버(20)의 포트(40)는 비동기, 동기 또는 평행 포트일 수 있다. 포트(40)에 부착되는 것은 터미널, 프린터 또는 모뎀과 같은 다양한 장치일 수 있다.

<54> 포트(40)를 통해 전송하기 위하여 클라이언트(18)로부터 전달되는 데이터는 포트(40)가 데이터를 전송할 준비가 될 때까지 출력 또는 송신 버퍼(42)내에 저장된다. 포트(40)를 통해 서버(20)내에 전송되는 데이터는 유사하게 조종된다. 클라이언트(18)에게 전송하기 위하여 포트(40)로부터 전송되는 데이터는 그것이 네트워크를 통해 클라이언트(18)에게 전송될 준비가 될 때까지 입력 버퍼 또는 수신 버퍼(44)내에 저장된다. 본 발명의 통신 프로토콜은 CPU(32)에 의해서 실행되고, 바이트스트림의 전송을 네트워크(10)에 필요한 프레임, 데이터그램 및 패킷으로 포맷팅하는 단계는 네트워크 인터페이스(30)에 의해서 조종된다.

<55> 서버 메모리(34)는 또한 통신 프로토콜을 충분히 실행하기 위하여 필요한 상태 정보를 포함하는 서버 상태변수(38)를 포함한다. 본 발명의 양호한 실시 예에서 저장되는 서버 상태 변수(38)의 형태는 표 1에서 도시된다.

<56> 이와 같은 변수 형태는 다양한 데이터 구조를 나타내며, 서버(20)의 작동을 제어하고 클라이언트(18)및 서버(20)사이에 통신을 원활하게 하도록 여러 방법으로 이용된다. 이와 같은 변수의 목적을 확실하게 하기 위하여, 하기에서 상세히 서술 될 것이다.

<57> 클라이언트 접속 리스트는 통신 프로토콜하에서 오픈된 서버(20)로의 현재 접속되는 클라이언트 리스트를 보존하고 있다. 양호한 실시 예에서 서버(20)는 8개 이상 클라이언트 접속을 동시에 지원할 수 있다.

<58> 메시지 빌드 인터벌은 통신 프로토콜에 따라서 구성되어서, 클라이언트(18)에 전송되는 정보가 필요한 가를 결정하기 위하여 얼마나 자주 서버(20)가 그 상태를 내부에 폴링하는지를 결정하기 위한 폴링 인터벌을 제어한다. 양호한 실시 예에서 이런 폴링은 초당 약 50 번 발생한다. 이결과, 풀링은 최악의 경우 20 밀리초루에, 행하여지며, 평균적으로는 10밀리초후에 행하여진다. 대부분 슬라이딩 식인 윈도우 프로토콜에 충분하다.

<59> 메시지 빌드 인터벌에서만 정보를 전송함으로써 서버(20)는 인터벌 중에 정보를 축적할 수 있다. 그럼으로써, 네트워크(10)와 클라이언트(18) 모두의 요구에 의해 제한되어서 한번에 한 캐릭터의 데이터를 전송하는 경우 보다 더 큰 패킷으로 데이터가 전송된다.

<60> 포트 오픈 상태 변수는 클라이언트(18)가 이미 포트(40)에 연결되는 오픈된 접속을 하고 있는가, 혹은 클라이언트가 포트(40)의 오픈을 기다리고 있는가를 추적하는 데이터 구조이다. 즉, 각 오픈 포트(40)에 대하여, 이와 같은 데이터 구조는 클라이언트가 현재 포트를 유지하는가, 혹은 클라이언트가 오픈 대기 리스트에 있는가를 추적하는 데이터 구조이다. 또한, 포트를 오픈하는 것에 관한 상세한 내용이 하기에서 서술될 것이다.

<61> 포트 전송률 변수는 포트(40)의 전송률이 포트 전송률 변수에 의해서 1,843,200로 분할되는 방식으로 각 포트의 전송률을 저장한다. 이런 표시는 50-115K 전송률의 범위로 모든 표준 전송률을 나타내기 위해서 단지 16 비트를 필요로 한다.

<62> 처리 플래그 변수는 캐릭터 사이즈, 하드웨어 상태, 및 각 포트에 대한 입력/출력 캐릭터 처리를 제어한다. 서버 상태 변수에 저장되는 4개 세트의 처리 플래그가 있으며 이들은 CFLAGS, IFLAGS, OFLAGS 및 XFLAGS이다.

<63> CFLAGS, IFLAGS 및 OFLAGS는 대부분 표준 UNIX 동작 시스템의 터미오 인터페이스내에 제공되는 표준 플래그이며, 본 발명에서도 산업분야에서 표준인 것으로써 사용된다.(UNIX 프로그래머 매뉴얼, 터미오).

<64> CFLAGS는 캐릭터 사이즈, 스톱 비트, 및 패리티를 포함하여 포트(40)의 UART 하드웨어 설정을 제어하는 설정을 가지고 있다. 서버(20)에서의 실행으로 이해되는 실제의 CFLAGS 설정은 CBAUD, CSIZE, CSTOPB, CREAD, PARENB, PARODD, 및 HUPCL을 포함한다. 유사하게, IFLAGS는 포트(40)로 부터 수신되는 서버(20)에서 데이터의 처리를 제어하는 설정을 가진다. IFLAGS설정은 브레이크 조건, 에러 조종 및 소프트웨어 플로우 제어 설정의 해석에 관한 제어를 포함한다. 서버(20)에서 관해 실행되는 IFLAGS는 IGNBRK, IGNPAR, PARMRK, INPCK, ISTRIP, IXON, IXOFF, IXANY, DOSMODE등이다.

<65> OFLAGS는 포트를 통하여 출력되고자하는 데이터를 처리하는 서버(20)를 제어한다. OFLAGS 설정은 CR 및 NL의 설정과 탭 지연과 같은 것을 제어하며, OPOST, OLCUC, ONLCR, OCRNL, ONOCR, ONLRET, OFILL, OFDEL, NLDLY, CRDLY, TABDLY, BSDLY, VTDLY, FFDLY를 포함하도록 터미오에 의해서 정하여진다.

<66> 바람직한 실시예에서, 더 이상 현대 통신 장비로 이용되지 않는 여러 가지 플래그 설정은 지지되지 않는다. 이런 플래그 설정은 OFILL, OFDEL, NLDLY, CRDLY, BSDLY, VTDLY 및 FFDLY, OFLAGS로부터 모든 것을 포함한다. 이들 각각의 설정에 대하여서는 양호한 실시 예에 대해 약간의 수정만을 기재하였지만 본 발명은 이들 기재된 설정에 한정되는 것은 아니다.

<67> XFLAGS는 약간의 추가 기능을 포함하여 터미오 LFLAGS로부터 취한 플래그의 설정이다. XFLAGS 설정은 표 2에서 서술된다.

<68> XFLAGS 변수(UNIX라인 규칙의 XCASE 또는 LFLAG 변수는 제외)는 인터페이스를 다수의 호스트 동작 시스템으로 확장이나 또는 가치부가기능을 제공하도록하는 UNIX기능의 설정에 대한 확장이다.

<69> 이들 XFLAGS 설정은 CFLAGS, IFLAGS 및 OFLAGS 설정과 상호작용 한다. 상호 작용의 상세한 내용은 다음과 같다.

<70> 만일 XFLAGS의 XPAR설정이 리세트되거나 또는 기초가 되는 하드웨어가 마크/스페이스 패리티를 지지하지 않는다면, CFLAGS의 PARODD는 설정이 되어 있는 경우 패리티를 선택하고 리세트되어 있는 경우에 짝수 패리티를 선택한다. 그렇지 않고 PARODD가 설정된다면 스페이스 패리티를, 리세트 된다면 마크 패리티를 선택한다.

<71> XFLAGS의 PARMRK와 XFLAGS의 XMODEM이 설정되면, 한 개 이상의 모뎀신호의 변화는 입력 스트림에서 FF 80 MM으로 인코드되며, 여기서 MM은 갱신된 모뎀신호 값이다.

<72> XFLAGS의 XCASE이 설정되면, 어떤 비-알파베틱 프린팅 캐릭터는 UNIX 사양의 출력에 상당하는 2개의 문자로 변환된다.

<73> XFLAGS의 XTOSS는 흐름 제어중에 출력이 재개되는 캐릭터가 무시되는지 어떤지를 제어한다. XFLAGS의 IXANY가 리세트될때, 출력은 XON 또는 XXON의 일치되는 캐랙턱가 수신될 때만 재스타트된다.

<74> 그러나, XFLAGS의 IXANY가 설정될때, 어떤 캐릭터를 수신하여도 출력을 재개한다. 그럼으로써, 만일 XTOSS가 설정된다면, 출력을 재개하는 캐릭터는 폐기된다. 만일 XTOSS가 리세트된다면, 출력을 재개하는 어떤 넌-플로우 제어 캐릭터도 데이터 스트림내에 위치된다.

<75> XFLAGS의 XIXON이 설정될 때, 흐름 제어 캐릭터의 출력의 특별한 설정이 가능하게 된다. XXOFF의 수신은 XXON이 그것을 재개하는 동안에 출력을 멈춘다.

<76> 흐름 제어 캐릭터 변수는 흐름 제어 캐릭터가 무엇이 될 것인지를 결정한다. 2개의 흐름 제어 캐릭터의 출력설정이 지지되며 그리고 하나의 흐름 제어 캐릭터의 입력설정이 지지된다. 또한, 흐름 제어에 스케이프 캐릭터(escape charactor)는 해석없이 흐름 제어 캐릭터의 입력을 허용하게 지지된다. 흐름 제어 캐릭터는 1) 표준 소프트웨어 흐름 제어 스타트 캐릭터 (XON), 2) 표준 소프트웨어 흐름 제어 스톱 캐릭터 (XOFF), 3)여분 소프트웨어 흐름 제어 스타트 캐릭터(XXON), 4)여분 소프트웨어 흐름 제어 스톱 캐릭터(XXOFF) 및 5) 흐름 제어 이스케이프 캐릭터(LNEXT)를 포함한다.

<77> LNEXT 캐릭터의 수신 후, 수신된 다음 캐릭터는 흐름 제어 캐릭터로써 인식 되지 않으며, 두개 캐릭터는 입력 데이터 스트림내에 위치된다.

<78> 서버 상태 변수 (38)로 되돌아가서, 클라이언트(18)으로의 전 모뎀 제어 기능의 준비는 8개 모뎀 제어 변수를 통해 제공된다. 이들 변수 각각은 모뎀의 동작 상태를 나타내기 위해서 마스크를 사용한다. 7개 모뎀 변수는 표3에서 다음과 같이 서술될 것이다.

<79> 표3에서 도시된 모든 변수는 표4에서 도시된 비트 할당을 사용하여 모뎀신호의 설정을 나타낸다.

<80> 다른 모뎀 제어 변수의 사용은 다음과 같다. MOUT모뎀 제어 변수는 클라이언트(18)에 의해서 최종으로 요청되는 RTS및 DTR값은 포함된다. 이들 변수는 서버(20)로의 요구를 함으로써 변경된다. MFLOW에서 RTS/DTR 비트가 설정될 때, MOUT 내에서 설정된 대응 값의 세트는 일시로 무시되고, 그들 신호는 서버가 데이터를 수신할 준비가 되었는지를 나타낸다. 수신이 멈추었을 때, 이런 신호는 로우로 구동된다. 수신이 재스타트될 때 신호는 하이로 구동된다.

<81> MFLOW는 "IN-BAND"출력 데이터를 금지하기 위해 사용된다. 데이터, 명령 및 데이터 포트의 내외에서 데이터의 정상 플로우와 동일 시컨스내에서 취급되는 응답은 IN-BAND데이터라 불린다. 데이너, 명령 및 정상 데이터에 선두에서 전송되고, 수신될 때 직접 처리되는 응답은 아웃오프(OUT-OF) 데이터라 불린다.

<82> IN-BAND 출력은 CTS/DRS/DCD/RI중 어느것이 MFLOW에서 설정되고, 대응하는 하드웨어 입력 신호중 적어도 하나가 로우일 때 금지된다.

<83> 흐름 제어 캐릭터 및 즉시 전송 캐릭터는 소프트웨어 흐름 제어 또는 MFLOW에 의해서 지배되지 않는다. 그것들은 MCTRL에 의해서 제공되는 CTS/DSR/DCR/RI 흐름 제어의 제약조건들에 따라서 전송된다. 서버(20)는 MFLOW에서 세트되지않은 MCTRL에서 무엇인가의 비트를 자동으로 리세트한다.

<84> MSTAT는 입력과 출력 모뎀 신호중 현재 상태를 포함한다.

<85> MLAST는 클라이언트(18)에게 보고된 마지막 모뎀 상태를 포함한다.

<86> MTRAN는 모뎀 신호가 클라이언트에게 마지막으로 보고되고 나서부터 변화된 모뎀 신호를 포함한다. MSTAT가 MLAST를 같게하는 것이 가능하고 MTRAN에서 설정된 비트를 가지고 있다. 이런 조건은 MSTAT에서 몇개의 신호가 적어도 하나의 천이를 이루고, 클라이언트에게 마지막으로 보고된 값으로 복귀하는 것을 나타낸다.

<87> 6개 모뎀 신호중 어느것인가가 MINT에서 설정될때, 이들 모뎀 신호에 대응하는 변화가 클라이언트에게 보고된다.

<88> 본 발명의 서버(20)는 포트(40)에서 발생되는 수신 캐릭터 에러의수를 일어 버리지 않도록 서버 상태 변수(38)내에 포트 통계 카운터마다 유지하고 있다. 이들의 카운터는 라인 에러 카운터로써 언급된다. 다른 카운터는 UART 오버런 에러의 수, 버퍼 오버플로우 에러의 수, 프레이밍 에러의 수, 패리티에러의 수, 수신된 브레이크의 수의 카운트 각각을 추적하기 위해 사용된다. 이들변수의 각각은 포트의 각 범주내에서 발생되는 라인 에러의 수를 놓치지 않는 16개 비트 순환 카운터이다.

<89> 부가적인 라인 에러 카운터 변수는 밀리 초로 보고시간을 추적한다. 이 변수가 제로일때, 라인 에러는 자동으로 클라이언트에게 보고되지 않는다. 그렇지 않으면, 서버는 각각의 특정한 시간간격에서 카운터를 검사하고, 마지막 보고 이래로 어떤 것이 변화된다면, 클라이언트에게 완전한 보고한다.

<90> 전송 브레이크 및 즉시전송변수(send immediate variable)는 브레이크 또는 다른 캐릭터를 포트가 서버(20)에게, 즉시 전송하는것을 클라이언트(18)가 요청하도록 허용한다. 즉시전송변수는 클라이언트가 전송하기를 원하는 캐릭터를 포함하는 하나의 캐릭터 버퍼이다. 전송 브레이크 변수는 요청된 브레이크의 밀리 초의 길이를 포함한다. 2개의 특정한 경우는 전송 브레이크 변수의 내용에 존재한다. FFFF값이 전송 브레이크로 전송될때, 이것은 무한의 브레이크 시간을 나타낸다. 전송 브레이크로 전송되는 0000은 진행중의 어떤 브레이크도 취소한다. 서버에 전송되는 어떤 다른 값은 전송 브레이크 시간에 단순히 추가되어서 브레이크는 두개의 요청의 결합된 존속기간 동안에 진행한다.

<91> 본 발명이 포트 흐름 제어를 위해 TCP/IP 흐름 제어를 이용하지않으므로, 클라이언트 또는 서버에 의해서 수신된 데이터는 항상 즉시 처리될수 있다.

<92> 그 결과로써, IN-BAND 데이터의 흐름을 파괴하지 않고 OUT-OF-BAND 명령 및 데이터를 전송하는 것이 용이하다.

<93> 서버(20)는 서버 상태 변수 (38)에서 상태 상황의 조건 리스트를 유지하고 있다. 이들 상태 상황은 사건 보고 변수에 저장된다. 그리고, 클라이언트(18)에게 일반적인 관심이 되는 상황을 나타낸다. 클라이언트는 이들을 등록할 수 있거나, 또는 이들 상황이 변화될 때마다 서버가 자동으로 적당한 사건 정보를 전송하는 것을 요청할수 있다. 사건 보고 변수는 서버에게 이런 사건 정보의 전송을 측적하는 것을 허용한다. 표5는 여러 개 사건 보고 변수를 나타낸것이다.

<94> EINT는 사건 정보가 서버에 의해서 클라이언트에게 전송되게 하는 기설정된 조건을 만들도록 클라이언트에 의해서 설정될 수 있다. 사건 정보는 (ETRAN & EINT)가 넌-제로(non-zero)일 때마다 클라이언트에게 전송된다.

<95> ESTAT가 ELAST와 같고, ETRAN내에 비트세트를 가지는 것은 가능하다는 것을 유의해라. 이런 조건은 ESTAT의 어떤조건들이 적어도 하나의 전이를 만들고 클라이언트에게 그 조건들이 보고될 수 있기 전에 ELAST값으로 복귀되는 것을 나타낸다.

<96> 이런 변수에서 기록된 사건은 표6에서 사건이 어떻게 나타내는지를 보이는 마스크와 함께 나타낸다. 표6에서 리스트된 사건은 본 발명의 문맥내에서 잘 이해되는 통신 사건이다.

<97> 서버 상태 변수(38)에 저장된 변수의 마지막 두개 형태는 입력 순서 및 수신 버퍼 매개 변수와 관련된 출력 순서 및 송신 버퍼 매개 변수 이다. 이들 변수는 송신용 버퍼(42)와 각 포트와 연결되는 수신용 버퍼(44)의 상태를 나타낸다.

<98> 버퍼(42, 44)내에 데이터의 흐름을 통신하기 위해서, 서버의 양호한 실시예는 16개의 비트 순환 시퀀스 넘버를 사용한다. 두 개의 시퀀스 넘버는 각 버퍼와 관련된다. 제1 시퀀스 넘버는 버퍼내에 위치될 다음 바이트의 시퀀스 넘버를 전달한다. 이런 제1 시퀀스 넘버는 수신용 버퍼(44)을 위한 RIN 및 송신용 버퍼(42)을 위한 TIN으로 분류된다. 제2 시퀀스 넘버는 버퍼로부터 제거될 다음 바이트의 시퀀스 넘버를 전달한다. 이런 제2 시퀀스 넘버는 수신용 버퍼(44)를 위한 ROUT와 송신용 버퍼를 위한 TOUT으로 분류된다. 이들 시퀀스 넘버는 제로에서 시작하고 전송 및 수신된 데이터의 각 바이트에 대하여 하나만큼 전진한다. 시퀀스 넘버가 FFFF에 이를때, 데이터의 다음 바이트가 송출되거나 수신될 경우, 시퀀스 넘버는 제로로 되돌아 간후 계속된다.

<99> TIN, TOUT, RIN, 및 ROUT 변수는 클라이언트 사이에서 통신하기 위해서 이용되는 시퀀스 넘버 변수임을 유의해야 한다. 그것들은 전송용 및 수신용 버퍼(42, 44)내에 데이터의 물리적 위치를 특정 하지 않는다. 이들 버퍼는 공지의 기술 이어서, 여기에서는 기재하지 않는다.

<100> 이들 시퀀스 넘버와, 입력 시퀀스 및 수신용 버퍼 매개변수와 출력 시퀀스 및 송신용 버퍼 매개변수 형태를 구성하는 다른 변수가 표 7 및 표 8에 도시되어 있다.

<101> TIN, TOUT, RIN, ROUT와 같은 순환 시퀀스 넘버는 매우 도움이 되는 수학적 성질을 가진다. 그러나, 시퀀스 넘버는 이들 함수의 특별한 정의가 그들의 순서(wrap around) 성질의 결과로써 사용되지 않으면 않되기 때문에, 가감을 행할 때 특히 혼란을 야기시킬 수 있다.

<102> 이들의 용도를 명료하게 하기 위해서, 송신용 버퍼(42)의 실시예가 제3,4도에서 예시되는 바와 같이 서술될 것이다. 도 3의 실시예에서, TOUT 시퀀스 포인터(버퍼(41)로부터 포트(40)로 전송될 데이터의 다음 바이트를 지칭하여 도 3에서 화살표 50으로 나타냄)는 바이트 S로 설정된다.

<103> 데이터의 n바이트가 전송될 때, 전송된 바이트는 비록 S+(n-1)일지라도 넘버 S로 된다.

<104> 약속에 의해 버퍼가 S 와 (S+n)사이의 전송된 바이트를 가지는 것이라고 말할수 있다. 그래서, oxFFFF에서 랩하는 시퀀스 넘버에 통상의 수를 더하므로, 다음 공식을 이용하여 수식 S를 계산해야 한다.

$$(S+n)\ \&\ oxFFFF$$

<106> 여기서, &는 논리 AND동작을 나타낸다. 그러므로, S가 0xFFF8이고 n이 5이면, (S + n)의 계산 공식을 통해 0xFFFC를 구할 수 있다. 마찬가지로, S가 0xFFF8이고 n이 9이면, (S + n)은 2이다.

<107> n 바이트를 전송한 후에는, TOUT 시퀀스 포인터가 전송될 다음 바이트의 새로운 지점으로 진행되지 않으면 않된다. 이것은 TOUT의 이전값을 결산된 결과의 (S+n)으로 치환함으로써 행했다.

<108> 두 개의 시퀀스 넘버 사이에 있는 바이트 수를 결정하기 위해 차감법이 사용된다. 그러나 한 시퀀스 넘버에서 다른 시퀀스 넘버를 뺀 나머지를 구하는 차감법에는 특별한 법칙이 적용된다. 그러므로, 두 시퀀스 넘버 s1과 s2 사이에 있는 바이트 수를 결정하는 데에는 다음 공식이 사용된다.

$$(s2 - s1)\ \&\ 0xffff$$

<110> 이 공식은 시퀀스 s1이 시퀀스 s2보다 훨씬 큰 수일 때에도 유효하다. 예를 들어, s1이 0xFFF0이고 s2가 3이면, (s2 - s1) & 0xFFFF의 값은 정답 0x13이다.

<111> 사실상, 두 개의 시퀀스 넘버간의 거리 개념은 매우 중요하기 때문에 다음과 같이 정의하면 유용하다:

DIST(from, to) = (((to) - (from)) & 0xffff)

<113> 두 개의 시퀀스 넘버 s1과 s2가 주어졌을 때, s3은 s1과 s2 사이에 존재하는 바이트의 시퀀스 지점을 나타낸다:

DIST(s1, s3) < DIST(s1, s2)

<115> 이 공식은, "s3이 s1과 s2의 사이에 존재하고, s1에서 s3까지의 거리가 s1에서 s2까지의 거리보다 짧다."는 사실을 나타내는 것이다.

<116> 시퀀스 넘버는 데이터의 양 또는 원격 버퍼에서 자유 공간의 크기를 전달할 수 있다. 도표 4에 나타난 바와 같이, 특정 포트(40)에 대한 전송용 버퍼(42)에서 TOUT 시퀀스 넘버(50)가 S_1으로 설정되고, 포트(40)를 통하여 전송된 다음 바이트를 나타내는 TOUT 시퀀스 넘버(50)가 S_3로 설정되어 있다고 가정한다. 위치를 지적하는 화살표로 나타낸다. 포트(40)의 송신용 버퍼(42)의 안에 있는 바이트 수는 다음과 같이 결정된다:

DIST(s1, s2)

<118> 더불어, 만약 이 송신용 버퍼(42)가 일정 크기 SIZE를 갖는다면, 버퍼 내에 있는 여유 바이트 수는 다음과 같이 나타낼 수 있다:

SIZE - DIST(s1, s2)

<120> 본 발명에 따른 클라이언트(18)와 서버(20) 모두는 시퀀스간의 인-밴드 정보 흐름을 제어하기 위하여 랩어라운드 시퀀스 넘버의 특성을 이용한다. 클라이언트(18)와 서버(20) 모두는 미리 허가 받은 경우에만 인- 밴드 데이터를 다른 데에 보낸다. 이와 같은 사전 허락은 데이터를 수신할 수신기에게 허용되는 데이터의 양에 대한 정보를 전달함으로써 이루어진다. 송신기가 이러한 허용 공간의 크기를 모두 채울 만큼의 데이터를 보냈으면, 송신기는 수신기가 이 데이터의 일부를 제거하고 송신기에게 나머지 공간이 확보되었음을 알릴 때까지 전송을 중단한다.

<121> 클라이언트(18)로부터 서버(20)로의 흐름을 제어하기 위해서, 서버(20)는 클라이언트(18)에 그의 송신용 버퍼(42)의 크기와 송신용 버퍼(42)내에 현재 저장된 제 1 바이트의 시퀀스(TOUT)를 알려준다. 클라이언트(18) 자신의 시퀀스 넘버의 송출을 계속 유지하면서 클라이언트(18)는 서버(20)에게 먼저 전송된 데이터를 추적하게 된다. 서버(20)가 제공한 정보와 함께 이와 같은 전송 시퀀스 넘버를 사용하여, 클라이언트(18)는 서버의 송신용 버퍼(42)에 남아있는 빈공간의 크기를 계산할 수 있게된다. 그래서, 클라이언트(18)는 빈공간에 적당하지 않은 보다 큰 데이터는 더 이상 보내지 않게 된다.

<122> 이 과정은 도표 5의 흐름도에 상세히 나와있다. 흐름도의 박스 100으로 전송 활동의 시작의 지시가 표시되어 있다. 첫 단계는 서버(20)의 네트워크 인터페이스(30)를 통하여 클라이언트(18)로부터 데이터가 수신되었는지를 결정하는 단계이다. 박스 102에서 처럼 데이터가 수신된 경우라면, 그 데이터를 받아서 적당한 포트(40)에 대한 송신 버퍼(42) 안에 두어야 한다. 데이터가 송신 버퍼(42)에 등질때마다 박스 104에 표시된것처럼 TIN은 증가한다.

<123> 클라이언트(18)로부터 데이터를 수신하였든 그렇지 않든 간에 관계없이, 송신용 버퍼(42)내에 있는 데이터는 질문 박스 106에서처럼 포트(40)를 통하여 내보내 질 수 있다. 데이터가 전송용 버퍼 내에 있으면, 버퍼(42)내의 데이터는 박스 108에서 포트(40)를 통하여 내보내저야 하고, 박스 110에서 TOUT는 증가한다.

<124> 이 점에 있어서, 서버(20)는 클라이언트(18)에게 TOUT를 보고해야 할 것인지를 결정해야한다. 질문 박스 112, 114 그리고 116에 나와있는 것처럼, 본 발명의 바람직한 실시예에서는 세 가지의 테스트를 이용하고 있다. 첫 번째 테스트 112에서는 전송될 다음 출력 바이트를 나타내는 TOUT를 TPOS와 비교한다. TPOS는 서버 상태 변수(38)에 저장되어 있는 출력 시퀀스 및 송신용 버퍼 파라미터 변수 중의 하나이다. TROS는 클라이언트(18)에 보고된 TOUT의 마지막 값을 나타낸다. 만약, 위에서 상술한 시퀀스 넘버 차감 공식을 써서 재산한 TOUT와 TMAX의 차이가 TMAX보다 크면, TOUT는 보고된다. TMAX도 또한 서버 상태 변수(38)에 저장되고, TMAX는 TOUT가 보고될 전송 간격을 결정하기 위해 클라이언트(18)에 의해 정해진다. 박스 120에서 TOUT가 클라이언트(18)에 보고될 때 마다, 박스 122에서 TPOS는 TOUT의 값으로 정해진다.

<125> 박스 114에서, TOUT를 보고하는 두 번째 테스트는 데이터가 적어도 TTIME 밀리초 이전에 버퍼(42)내에 있었는지 그리고 그 이후에 시간 TOUT가 보고되었는지를 결정한다. 단, TTIME은 클라이언트(18)가 보낼 수 있는 서버 상태 변수(38)에 해당한다. 만약 그렇다면, TOUT는 클라이언트(18)에 보고된다.

<126> 마지막 테스트에서, TOUT값을 TREQ와 비교한다. 서버 상태 변수(38) 중의 하나로서 저장된, TREQ는 클라이언트가 지정하여 TOUT가 TREQ를 초과하였을 때 TOUT를 클라이언트(18)에 보고하도록 한다. 수학적으로, 랩어라운드 시퀀스 넘버 TOUT를 TREQ와 비교하기 위해서는 위에서 설명한 바와 같은 시퀀스 넘버 차감 공식을 써서 (TIN - TREQ)가 (TIN - TOUT)보다 크거나 같은지를 비교하여 그 관계를 검토해야 한다. TOUT값이 TREQ보다 크면, 박스 118에서 TREQ는 무효가 되고 박스 120에서 TOUT는 클라이언트(18)에 보고된다.

<127> 클라이언트(18) 드라이버 소프트웨어의 융통성을 허용하기 위해서, 클라이언트(18)는 클라이언트(18)가 서버의 송신용 버퍼(42)를 한정초과하지 않는한 그렇게 하는 것이 편리할때는 매번 서버(20)에게

데이터를 보낼 수 있다. 도표 6은 송신용 버퍼(42)가 한정초과 되지 않도록 하는 과정을 보여준다.

<128> 클라이언트(18)가 포트(40)에 데이터를 보내고자 할 때에는, 박스 130에서처럼 클라이언트(18)의 통신 프로토콜을 조작하는 클라이언트(18)상의 드라이버가 이를 위해 서버(20)로 하여금 포트(40)를 오픈하도록 요청해야 한다. 포트(40)를 오프닝하는 자세한 설명은 아래에 나와 있다. 포트의 오픈에 성공하였으면, 클라이언트(18)는 오픈 포트(40)와 연결된 송신용 버퍼(42)의 크기인 TSIZE와 TOUT가 0이라고 판단한다. 또한, 클라이언트는 자체의 전송 시퀀스 넘버를 재지정하는데, 이는 랩어라운드 시퀀스 넘버 변수로서 클라이언트(18)상에 저장되어 있다. 다음에, 클라이언트(18)는 박스 132에서처럼 서버로부터 TSIZE를 요청한다. 이 때, 아래에 설명된 바와 같이 박스 140에서처럼 TSIZE를 서버(20)로부터 받기전까지 클라이언트는 실질적으로 데이터를 받는 것이 아님을 주의해야한다.

<129> 그 다음에, 박스 134와 같이 클라이언트는 호스트 컴퓨터로부터 데이터가 수신되었는지를 결정한다. 만약 데이터가 수신되었으면 데이터는 박스 135에서와 같이 데이터 전송을 준비한다. 데이터 전송의 준비는 나중에 상세히 설명하기로 한다.

<130> 박스 136에서, 클라이언트(18)는 이어서 전송될 데이터가 있는지를 검사한다. 데이터를 보낼 준비가 되어 있으면, 클라이언트의 드라이버는 박스 138에서 송신용 버퍼(42) 한정량을 초과하지 않고 서버(20)가 받을 수 있는 최대 바이트 수를 계산해야 한다. 이 것은 클라이언트의 전송 시퀀스 넘버를 가장 최근에 보고된 TOUT값과 비교하여 TSIZE에서 최근의 TOUT값을 차감하여 구한다.

최대로 보낼 수 있는 바이트의 수 =

TSIZE - (전송 시퀀스의 수 - TOUT)

<132> 물론, 클라이언트의 전송 시퀀스 넘버에서 시퀀스 넘버 TOUT를 차감하는 것은 시퀀스 넘버를 차감하는 공식에 따라 행한다.

<133> 수신될 최대 바이트의 수가 0보다 큰 값으로 계산되었으면, 박스 140에서, 박스 144에서와 같이 그 값만큼의 바이트를 서버(20)에게 보낼 수 있다. 그러면, 클라이언트의 전송 시퀀스 넘버는 박스 146에서 전송된 바이트의 수만큼 증가하게 된다.

<134> 데이터의 전송여부에는 무관하게, 박스 148에서 클라이언트(18)는 이어서 서버(20)가 TOUT 또는 TSIZE값을 보냈는지를 결정한다. 결과가 그렇다면, 박스 149에서 클라이언트(18)에 저장된 TOUT 또는 TSIZE의 값을 새로 등록한다.

<135> 송신기가 작동중임에도 불구하고 작동중이 아닌것과 같은 허위보고를 피하기 위하여는 서버(20)는 아주 드문 경우를 인지한다. 송신용 버퍼(42)가 비어있고 UART가 인-밴드 데이터를 보내는 중이라면, 서버(20)는 TOUT 대신에 (TOUT - 1)을 보고한다. 이와 같은 변환으로, 클라이언트가 서버(20)에게 보낸 마지막 데이터의 전송 시퀀스 넘버와 같은 TOUT를 수신할때 모든 데이터가 성공적으로 전송된 것과 같이 클라이언트(18)는 인식하게 된다.

<136> 포트(40)로부터 수신하고 서버(20)로부터 클라이언트(18)에게 전송한 데이터는 도포 7과 같이 다루어진다. 서버(20)는 박스 150과 같이 RWIN 값을 받을 때까지는 데이터를 클라이언트(18)에게 전송하지 못한다. RWIN은 클라이언트(18)가 받기를 원하는 가장 높은 시퀀스 넘버를 지적하기 위하여 클라이언트(18)가 보낸 입력 시퀀스 및 수신 버퍼 변수이다.

<137> 초기 RWIN을 클라이언트(18)로부터 받으면, 서버(20)는 입력될 데이터가 수신용 버퍼(44)내에 수용될 수 있는지를 알아내기 위해 포트(40)를 검토한다. 만약 데이터가 수신되면, 수신용 버퍼(44)내에 들어오고 RIN 시퀀스 넘버는 증가하게 된다. 이와 같은 과정이 박스 154와 박스 156에 나타나 있다.

<138> 서버(2)는 클라이언트(18)가 얼마나 빨리 서버(20)로부터 데이터를 수신할 수 있는지를 모르기 때문에, 서버(20)는 포터(40)에 대한 입력 흐름 제어 능력을 갖추고 있다. 흐름 제어는 수신용 버퍼(44)내에 남아있는 데이터의 양(RIN - ROUT 로 계산됨)이 다른 입력 시퀀스 및 수신용 버퍼인 RHIGH의 값을 초과할 때 시작한다. 그러므로, 데이터가 수신용 버퍼(44) 내로 들어와서 RIN이 증가한 다음에 박스 164와 164에서처럼 서버(20)는 입력 제어를 시작할 것인가를 결정한다.

<139> 수신용 버퍼(44)의 데이터 양이 입력 시퀀스와 수신용 버퍼 RLOW에 못미칠 때 흐름 제어는 해제되고, 이는 박스 158과 160에서 나타난 바와 같이 데이터가 클라이언트(18)에 보내졌을 때 테스트한다.

<140> 만약 152 단계에서 포트(40)를 검토한 결과 포트(40)상의 새로운 데이터를 발견하지 못했으면, 서버(20)는 데이터가 수신용 버퍼(44)내로 보낼 수 있는 것인지를 결정하게 된다. 그 반대인 경우에는, 박스 166의 도표 7에서 지적한 것과 같이, 서버(20)는 다시 포트(4)에서 데이터를 기다린다.

<141> 수신용 버퍼(44)상에 데이터를 수용할 수 있게되면, 박스 168에서 서버(20)는 클라이언트에게 보낼 수 있는 최대 바이트 수를 계산해야 한다. 바이트 수는 간단한 공식(RWIN - ROUT)에 의해 결정하고 시퀀스 넘버 차감 공식에 따라 계산한다.

<142> 전송할 바이트의 최대수가 0이라면, 서버(20)는 질문 박스 170에서처런 RWIN이 새로 등록될 때까지 기다려야 한다. 정산된 수가 0을 넘으면, 서버(20)는 데이터 바이트를 동시에 클라이언트(18)에 전송하는 것이 적당한지를 결정한다. 데이터를 보내기 시작하는 데에는 두가지 테스트가 이용된다. 첫 번째 테스트에서는 박스 172에서와 같이 수신용 버퍼(44)의 바이트 수(RIN - ROUT)를 클라이언트(18)가 지정할 수 있는 입력 시퀀스와 수신용 버퍼인 RMAX와 비교한다. 만약 수신용 버퍼(44)내의 바이트 수가 RMAX를 초과하면, 박스 176에서처럼 최대 허용 바이트를 클라이언트(18)에게 보내고 따라서 박스 178에서처럼 ROUT는 증가하게 된다.

<143> 수신용 버퍼(44)내의 바이트 수가 RMAX를 초과하지 않는다면, 서버(20)는 데이터가 적어도 RTIME

밀리초 이전에 수신용 버퍼(44) 내에 존재했었는지를 결정하고 이후부터는 데이터를 클라이언트(18)에게 보내지 않게 된다. 그런 다음에는, 질문 박스 174에서 보인 것처럼 데이터는 클라이언트(18)에게 보낸다.

<144> 만약 데이터를 클라이언트(18)에 보내지 않으면, 데이터 전송 조건(172) 또는 (174) 중의 어느 하나가 참이 될 때까지 데이터는 수신용 버퍼(44) 내에 남아 있게 된다.

<145> 포트(40)를 오픈한 다음에, RMAX를 1, RTIME을 0, RLOW를 RSIZE의 1/4, RHIGH를 3/4 그리고 모든 시퀀스 넘버(sequence number)를 0으로 초기화한다. 그러나 이들 변수들을 임의로 변경할 수 있다.

<146> 일반적으로, 전송될 수 있는 계산된 바이트의 최대수가 제한되어 있지 않은 경우에, 서버(20)는 데이터를 포트(40)에 보낼 때 모든 데이터를 각 포트(40)에 보낸다. 이렇게 함으로써 서버(20)와 클라이언트(client)(18) 모두가 처리해야 하는 패킷의 수를 줄이고, 일반적으로 클라이언트(18)의 효율을 향상시킨다.

<147> 클라이언트 측면에서, 호스트 컴퓨터내 본 발명의 작동은 호스트 컴퓨터의 작동시스템에 따라 크게 다를 수 있다. 클라이언트(18)에서 본 발명의 작동의 목적은 도표 8에서 보는바와 같이 유닉스 호스트 시스템에 대한 보충 대안이다. 본 발명은, 구역내에 접속된 세트의 시리얼 포트(204)용의 드라이버를 에뮬레이션함으로써 시스템을 가동시키는 호스트의 API를 보충해주는 클라이언트 드라이버(200)의 사용을 통하여 서버(20)의 포터(40)에 접속할 수 있도록 한다. 이를 실행하기 위하여, 클라이언트 드라이버(200)는 각각의 원격 서버포트(204)에 대하여, 입력 또는 수신용 버퍼(receive buffer)(206), 출력 또는 송신용 버퍼(transmit buffer)(208)원격 서버(20)의 상태에 관한 정보 및 상태 메모리(status memory)(210)에 저장되어 있는 호스트 컴퓨터로의 인터페이스에 대하여 지원한다.

<148> 드라이버(200)도 범용의 네트워크(10)에 접속되어 있는 표준 통신 인터페이스(212)와 접속된다. 클라이언트 드라이버(200)에서 서버(20)로의 모든 통신은 네트워크(10)를 통하여 전송된다. 통신 인터페이스(212)는, 드라이버(200)가 통신 프로토콜하에서 다수의 포트(40)에 상관없이 각 서버(20)당 오직 한 개의 네트워크에 접속한다.

<149> 시스팀 부팅시, 작동 시스템의 시동으로 드라이버(200)는 초기화되고, 각 서버(20)들은 스트림즈 커넥션(STREAMS connection)(218)을 서버(20)에 오픈시키는 사용자 모드 데몬(deamon)(214)을 시동한다. 그 다음에, 데몬(214)은 스트림즈 커넥션(220)을 드라이버(200)와 연동하는 제어 장치(202)에 접속한다. 그다음, 데몬(214)은 스트림즈 인터페이스를 사용해서 드라이버(200)가 네트워크 접속상의 데이터를 데몬으로부터의 상호작용으로 서버(20)의 W/O에 직접 송수신하도록 작동시스템이 제어장치(202)가 보다 하위의 스트림(218)을 링크하는 것을 요구한다. 이렇게 함으로써, 드라이버(200)는 제어 장치의 제어하에 서버(20)로부터 스트림열의 TCP/IP 데이터를 송· 수신할 수 있는 것이다. 이로써, 종래에 요구되었던 태스크 스위치 오버헤드를 삭제한다.

<150> 접속은, 드라이버(200)가 신뢰할 수 있는 순차적인 데이터를 서버(20)로 송· 수신할 수 있도록 하는 바이트스트림 접속으로 이루어진다. 드라이버(200)는 이와 같은 접속의 신뢰성에 전적으로 의존하고 있으며, 손실된 패킷과 같은 것을 회복시키는 기능에 대하여는 구비되어 있지 않다. 드라이버(200)가 서버(20)로부터 받은 데이터에서 에러를 발견하면, 드라이버(200)는 접속해제 신호(hangup signal)를 데몬(214)으로 통과시켜 접속을 끊고 다시 서버(20)로 접속을 재시도하게 된다. 이와 같은 에러는 일반적인 작동시 발생할 수 없다. 에러는 원격 서버 소프트웨어나 통신 소프트웨어, 드라이버 소프트웨어 또는 어떤류의 컴퓨터 오류로 인해서만 발생한다. 호스트 컴퓨터 내의 표준 TCP/IP 소프트웨어는, 이러한 에러가 범용 컴퓨터 네트워크(10)에서 일상적으로 일어나는 손실되었거나 잘못 전달된 패킷에 기인하여 발생하지 않도록 효과적으로 하여준다.

<151> 일단 드라이버(200)가 네트워크에 엑세스하였으면, 드라이버는 서버(20)가 서버(20)상의 다수의 시리얼 포트(40) 및 서버(20)의 그 밖의 하드웨어나 소프트웨어에 관한 정보를 보내주기를 요구한다.

<152> 사용자 모드 태스크(216)가 TTY 시리얼 포트(204)를 오픈하려 하면, 호스트 작동시스템(215)은 드라이버(200)에 오픈 호출(open call)을 한다. 만약 포트(204)가 다른 사용자 모드 태스크(216)에 의해 오픈되어 있지 않으면, 드라이버(200)의 TTY 포트(204)와 연결된 서버(20)상의 포트(40)에 독점적으로 엑세스하기 위하여 드라이버는 서버(20)에게 오픈 요청 신호를 보낸다. 만약 포트(204)가 이용가능하면, 서버(20)는 클라이언트(18)가 포트(40)를 종료하거나 또는 클라이언트와의 접속이 해제될때까지 클라이언트(18)로의 독점적 엑세스를 허용하도록 응답한다.

<153> 만약 포트(40)가 이용할 수 없을때, 초기에 사용자 모드 태스크(216)가 보낸 오픈 호출의 종류에 따라 대응이 다르게 된다. 사용자 모드 태스크(215)가 오픈이 즉시 가능하지 않을 때에는 오픈 실패인 것을 요청 하거나, 포트(40)를 이용할 수 있을 때까지 요청을 대기시킬 것을 요청을 할 수 있다. 전자의 경우에 드라이버(200)는 서버(20)에 즉시 오픈할 것을 요청하는데, 만약 포트(40)가 사용중일 때에는 서버(20)가 이를 거절하며 드라이버(200)는 사용자 모드 태스크(216)에 에러 메세지를 보낸다. 후자의 경우에는, 드라이버(200)는 서버(20)에게 대기중인 것을 알리고 포트(40)가 최종적으로 사용될 수 있을 때까지 클라이언트에게 대기의 순서를 기다릴것인가를 묻는다. 그러면, 서버(20)는 요청이 순번 대기인것을 알리고 드라이버(200)는 서버(20)가 드라이버(200)에 포트(40)을 사용할 수 있다는 사실을 알릴 때까지 사용자 모드 태스크(216)를 정지시킨다.

<154> 일단 드라이버(200)가 성공적으로 서버 포트(40)를 오픈하였으면, 드라이버(200)는 전송율 발생기의 헤르쯔이나 포트(40)가 마크 및 스페이스 패리티(mark and space parity)를 지원할 수 있는지의 여부를 포함하는 포트(40)의 하드웨어적, 소프트웨어적인 특성을 학습하도록 서버(20)에게 조회 패킷(inquiry packet)을 보낸다. 드라이버(200)는 또한, 포트(204)에 대한 수신용 버퍼(206)의 크기를 서버(20)에 보내서 클라이언트(18)가 허가없이 받은 데이터의 양을 서버(20)가 알 수 있도록 한다. 서버(20)는 조회 내용에 대응하여, 송· 수신용 버퍼(420)(44)의 크기를 포함한 포트(40)에 관한 특성을 보낸다.

<155> 드라이버(200)는 서버(20)로부터 응답을 받으면, 송· 수신용 데이터를 포트(40)에 보낼 준비를 하고 호스트 작동 시스템의 오픈/종료/입력· 출력 제어 인터페이스로부터 요청이 있으면 사용자 모드 태

스크(216)를 대신하여 제어 및 상태에 관한 조회를 실시한다. 그 다음에, 서버(20)가 받은 포트(40)에 관한 모든 데이터는 자동적으로 클라이언트(18)에 보내져서 수신용 버퍼(206)에 저장된다. 서버(20)는 처음에 이 버퍼(206) 이상을 넘지 않는 적당한 양만큼의 입력포트 데이터를 보낼 수 있도록 허가 받게 되므로, 버퍼가 제한된 크기 이상으로 사용될 가능성도 있다.

<156> 사용자 모드 태스크(216)가 데이터를 읽기를 요청하면, 호스트 작동 시스템은 드라이버(200)에게 읽기 루틴(routine)을 호출한다. 서버(20)로부터 충분한 데이터가 들어왔으면, 읽기 요청의 파라미터에 따라 드라이버(200)는 호스트 작동 시스템(215)에 되돌려 보내어서 그 데이터를 사용자 모드 태스크(216)로 통과시키도록 한다. 만약 충분한 데이터가 서버(20)로부터 들어오지 않았으면, 드라이버(200)는 데이터가 모두 도착할 때까지 또는 호스트 작동 시스템(215)의 API에 따라 읽기 요청 시간이 끝나거나 중단될 때까지 사용자 모드 태스크(216)를 정지시킨다.

<157> 드라이버(200)가 수신용 버퍼(206)에서 받은 데이터를 지우면, 드라이버 (200)는 위에 상술한 바와 같이 시퀀스 넘버(RWIN)를 삭제한 바이트의 수만큼 증가시켜 데이터가 삭제되었음을 서버(20)에 알린다. 그러면 서버(20)는 데이터가 시퀀스 넘버 RWIN에 이를 때까지 데이터를 부가하여 보낼 수 있게 되는 것이다.

<158> 사용자 모드 태스크(216)가 데이터를 쓰기를 요청하면, 호스트 작동 시스템(215)은 드라이버(200)에 쓰기 루틴을 호출한다. 드라이버는 수신용 버퍼(208)에 들어갈만한 크기의 사용자 데이터를 복사한다. 만약 모든 데이터가 수신용 버퍼(208) 안에 들어갈 수 있으면, 서버(20)는 요청을 완료하여 사용자 모드 태스크(216)를 계속할 수 있도록 한다. 만약 모든 데이터가 수신용 버퍼(208)에 맞지 않게 되면, 드라이버(200)는 데이터를 서버(20)에 보낼 수 있을 때까지 사용자 모드 태스크(216)를 정지시켜서 수신용 버퍼(208)에 나머지 데이터를 채울 수 있도록 충분한 공간을 확보한다.

<159> 사용자 모드 태스크(216)가 제어 또는 조사 요청(입· 출력 요청으로 공지됨)을 하였을 때에는, 호스트 작동 시스템(215)은 드라이버(200)에 입· 출력 루틴을 호출한다. 이 호출로 인하여 포트(204)의 하드웨어 또는 소프트웨어적 정보가 변경하게 되는 경우에, 드라이버(200)는 상태 메모리의 변화를 인지하여 필요한 변화를 취할 수 있도록 서버(20)에 명령한다. 만약 서버(20)의 협조없이 응답할 수 있는 데이터를 요청하는 것이면, 드라이버(200)는 이에 응답한다. 그러나 서버(20) 안의 데이터를 요청하는 것이면, 드라이버(200)는 서버(20)에게 정보를 요구하고 이 정보를 사용자 모드 태스크(216)에 보낸다. 사용자 모드 태스크(216)가 할 수 있는 요청의 종류는 무척 다양한데, 이에 대응하는 방식도 여러가지이다. 어떤 경우에는 즉시 응답이 가능하고, 어떤 경우에는 요청이 완료될 때까지 사용자 모드 태스크(216)를 정지시켜야 한다.

<160> 사용자 모드 태스크(216)가 TTY 포트(204)에 최종 종료를 요청하였으면, 호스트 작동 시스템은 드라이버(200)에 종료 루틴을 호출한다. 이렇게 함으로써, 드라이버(200)는 송신용 버퍼(208) 안에 있는 모든 데이터가 전송될 때까지 그리고 서버(20)에 정지 요청이 완료될 때까지 기다리도록 사용자 모드 태스크(216)를 정지시킨다. 그리고는, 드라이버(200)는 서버(20)에 종료 요청을 보내고 서버(20) 포트(40)를 박탈하여 다른 클라이언트(18)에 재 부여한다. 그러면, 드라이버(200)는 종료를 완료하게 되고, 사용자 모드 태스크(216)는 다시 지속된다.

<161> 드라이버(200)의 작동 효율을 최적화하고 호스트 작동 시스템(215)에 대한 부하를 최소화하기 위해서는, 서버(20)로의 드라이버(200) 통신에 약 50회/초의 주기적인 간격이 있어야 한다. 각각의 간격에서, 드라이버(200)는 서버(20)에 오픈된 각각의 활동 포트(204)를 검사하고 모든 활동 포트들(204, 40)에 요청메세지와 데이터를 채워서 단일 동작으로 서버(20)에 보내야 한다. 이로써 서버(20)에 전달되어야 하는 TCP/IP 메세지(또는 기타 범용의 네트워크 메세지)의 총수를 최소화하여 시스템의 가동을 향상시킬 수 있는 것이다.

<162> 본 발명은 네트워크(10)을 통한 시리얼 포트 하드웨어를 시뮬레이션 하는데 사용하는 프로토콜을 제공하는 데 있다. 이와 같이 사용했을 때는, 로컬 시리얼 포트 보드와 유사한 호스트 컴퓨터에 시리얼 포트 인터페이스를 제공하는 방식으로 보드 또는 기타 하드웨어를 호스트 컴퓨터에 장착시키게 된다. 이와 같은 실시예에서 보드는, 클라이언트 드라이버(200)와 관련하여 위에서 상술한 시리얼 포트 인터페이스와 기능성을 제공하는 소프트웨어뿐만 아니라, 범용의 통신 프로토콜층을 구비한다. 이 같은 보드를 사용함으로써, 드라이버를 시리얼 포트 인터페이스만을 인지하고 네트워크(10)의 존재는 알 필요가 없는 호스트 작동 시스템 상에서 사용할 수 있게 되었다.

<163> 이와 같은 하드웨어적 보충대안은 몇가지 이점을 갖는다. 첫째로, 호스트 작동 시스템 상에 드라이버를 이미 갖추고 있는 (또는 장착이 쉬운) 하드웨어 또는 소프트웨어 인터페이스를 에뮬레이션 하는데에 사용될 수 있다. 둘 째, 이 보드는 통신용 하드웨어 또는 소프트웨어를 갖추지 않은 컴퓨터에 장착시킬 수 있어서, 호스트 컴퓨터에 통신 서비스를 제공한다. 세 째, 이 보드는 카드 위에 장착한 많은 수의 시리얼 포트를 덜어낸 부가적인 프로세서와 하드웨어를 포함한다. 마지막으로, 이와 같은 카드는 예를 들어 로컬 컴퓨터용이나 통신 가능한 적은 수의 물적 장치들을 포함함으로써 특성상 다기능적이다.

<164> 서버(20)와 클라이언트(18)간에 교환된 데이터 및 명령들을 포함하는 모든 종보들은 패킷이라고 불리는 작은 소자 내에 있는 통신 프로토콜이 제공한 바이트 스트림을 통해 보내진다. 이들 패킷의 포맷은 본 발명의 통신 프로토콜을 구성한다. 통신 프로토콜 패킷은 시퀀스의 초기 수 바이트가 포맷을 결정하고 뒤에 이은 데이터의 길이를 결정한다는 점에서 포맷상 CRT 단말기 에스케이프 시퀀스와 유사하다.

<165> 서버(20)에서 클라이언트(18)에 이르는 통신용 패킷은 서버(20)의 CPU 32에서 발생한다. 클라이언트(18)에서 서버(20)로 보내지는 패킷은 클라이언트(18)상의 특수 드라이버 오퍼레이팅으로 발생한다. 패킷은 포함하고 있는 정보의 종류에 따라 그 포맷과 길이가 다양하다. 각 패킷의 포맷은 이 정보를 순차적으로 처리함으로써 결정된다. 초기 바이트 값은 포맷과 다음에 이어지는 바이트의 크기를 결정한다.

<166> 표 9는 서버(20)와 클라이언트(18)간에 전달되는 여러 가지 패킷을 보여주고 있다. 또한 첫 번째 바이트가 패킷을 주 서브 타입으로 나누는 것을 보여준다. 금지를 나타내는 값은 본 발명의 실시예에서 사용하지 않는다.

<167> 데이터 패킷은 모든 인-밴드(in-band) 데이터를 클라이언트(18)와 서버(20)간에 전달하는 데에 이용된다. 서버(20)가 수신하는 경우, 데이터 패킷 안에 있는 데이터는 위에서 상술한 바와 같이 적당한 포트(40)로 전송된다.

<168> 윈도우 시퀀스 패킷은 수신기에 송신기의 시퀀스 넘버의 상황을 알려준다. 이와 같은 정보의 전달은 위에서 개략적으로 설명한 것처럼 본 발명의 데이터 흐름 제어 작업에서 필수적이다. 윈도우 시퀀스 패킷은 서버(20)가 TOUT를 클라이언드(18)로, 클라이언트(18)가 RWIN을 서버(20)로 보내는 데에 사용된다.

<169> 커맨드 패킷은 바이트 내 커맨드 타입 필드(command type field)의 값에 따라 그 포맷과 길이가 다양하다. 커맨드는 서버 상태 변수(38) 값을 지정하고 묻기 위하여 전달된다. 또한, 커맨드 패킷은 클라이언트(18)가 서버(20)와 동기화하고 서버의 송· 수신용 버퍼(42)(44)의 크기와 특정 포트(40)의 용량을 확보하고, 송· 수신용 버퍼(42)(44)를 회복시키고 브레이크 또는 즉각적인 특성을 보내고 입· 출력을 중지 또는 재시도할 수 있도록 포트(40)를 오픈하는 데에 사용된다. 본 발명의 실시예에 따른 커맨드 패킷은 표 10에 나타내었다.

<170> 두 가지의 가장 중요한 커맨드 패킷은 클라이언트(18)와 포트(40)를 오프닝 하는 것과 관련하여 서버(20)간의 통신과 관계가 있다. 포트(40)를 오프닝하라는 오픈 요청 명령을 서버(20) 상의 포트(40)에 우선적으로 접속코자 하는 클라이언트(18)가 서버(20)에 보내면, 서버(20)는 포트(40)의 오픈을 시도하고 그 결과를 클라이언트(18)에 알리는 오픈 응답 패킷으로 답한다.

<171> 오픈 요청 커맨드 패킷과 오픈 응답 커맨드 패킷은 표 11과 표 12에 나타나있다.

<172> 포트(40)를 오픈하라는 커맨드 패킷은 클라이언트(18)가 원하는 오픈 커맨드 패킷의 종류에 따라 형태가 다른데, 즉시 오픈(immediate open)과 지연 오픈(persistent open), 도래 오픈(incoming open)이 있다.

<173> 즉시 오픈인 경우, 포트(40)가 유효화 이어서 즉각 사용가능한 경우에만 클라이언트(18)에 받아드려서 지정된다. 그러나 포트(40)를 즉시 사용할 수 없게 되면 요청은 거절된다.

<174> 지연 오픈인 경우, 포트(40)가 사용가능하기만 하면 요청이 받아들여져 클라이언트(18)에 지정된다. 그러나 포트(40)가 사용중이면, 서버(20)는 오픈 요청 패킷을 클라이언트(18)에 돌려보내서 포트(40)가 사용중임을 알리고 클라이언트(18)를 해당 포트(40)를 사용하고자 하는 대기자 명단에 올려놓는다. 차례가 되어 포트(40)를 사용할 수 있게 되면, 포트(40)는 오픈되고 클라이언트(18)는 성공했음을 나타내는 두 번째 오픈 요청을 통지받게 된다.

<175> 오픈 요청의 도래 제 3 형식은 도래 오픈에 대한 요청이다. 이요청은 포트(40)가 사용가능하면 즉시 이루어지고, 캐리어(carrier)가 포트(40) 상에 있게된다. 그러나 포트(40) 가 사용중이면, 캐리어는 없고 서버(20)가 해당하는 오픈 요청 코드를 돌려보내어 클라이언트(18)를 사용 대기자 명단에 올려놓는다. 차례가 되어 포트(40)를 사용할 수 있게 되면, 포트(40)는 오픈되고 클라이언트(18)는 성공했음을 나타내는 두 번째 오픈 요청을 통지받게 된다.

<176> 클라이언트(18)는 또한 특정 포트(40)에 대하여 대기 취소 명령을 서버(20)에 보낼 수 있는데, 서버(20)는 클라이언트가 해당 포트(40)의 사용 대기자 명단에 있는지를 체크한다. 클라이언트(18)가 명단에 있으면, 클라이언트(18)는 명단에서 삭제되고 서버(20)는 오픈 요청을 돌려보내 성공했음을 알린다.

<177> 서버(20)상의 포트(40)를 추적용 포트 그룹(port hunt group)으로 조직할 수 있다. 이 때, 서버(20)는 추적용 포트 그룹으로 지정된 한 개 이상의 논리적으로 포트들을 제공하여 이들 포트들 중 하나를 오픈케 함으로써 실제 추적용 포트 그룹의 일원으로 배열된 포트(40)를 오픈할 수 있게 된다. 따라서, 클라이언트(18)가 포트 헌트 그룹의 일부인 한 포트(40)를 요청하면, 요청한 포트(40)와 다른 포트(40)가 오픈하게 된다. 클라이언트(18)는 오픈에 성공한 후에 오픈 응답의 PNUM 부위를 검색하여 그 포트(40)에 부속된 모든 포트 번호를 사용해야 한다.

<178> 동기화 요청 명령을 하면 이를 수신한 쪽인 동기화 응답으로 응답하게 된다. 그 외에 다른 영향은 없다. 이 명령의 첫째 목적은 클라이언트가 하여금 서버와의 동기화할 수 있도록 하는데에 있다. 예를 들어, 클라이언트(18)가 호출 프로그램으로 돌아가기 전에 요청한 전송율의 변경을 확인하면 그 다음에 동기화 요청을 실행한다. 서버(20)가 순차적으로 명령을 처리하기 때문에 클라이언트(18)가 동기화 요청을 받을 때 클라이언트(18)는 전송율이 이미 변경되었음을 확인할 수 있게된다.

<179> 시퀀스 요청 및 응답 패킷은 클라이언트(18)가 서버(20)로 들어가고 나오는 데이터의 흐름을 정확하게 알 수 있도록 한다. 서버(20)는 클라이언트(18)에 RIN과 TOUT를 제공함으로써 이 패킷에 응답한다.

<180> 클라이언트(18)는 서버(20) 내의 ESTAT 및 MSTAT의 흐름 상태를 인식하기 위해 상태 요청 패킷을 보낼 수 있다. 서버(20)는 상태 응답 패킷 내 요청된 데이터를 돌려보내고 다른 대응은 하지 않는다. 이 명령은 클라이언트가 서버(20)의 상태를 알 수 있게 되는 두가지 방법 중의 하나가 된다. 클라이언트(18)는 또한 서버로 부터 선택된 상태의 변경사항을 자동적으로 보고받을 수 있도록 선택 이벤트 조건 패킷(select event conditions packet)을 이용할 수도 있다.

<181> 클라이언트(18)는 라인 에러 요청 패킷을 보내어 라인 에러 카운터를 조사하고 라인 에러 카운터에 관한 주기적인 보고를 파악하고 해당 카운터를 제거한다. 서버(20)는 에러 카운터에 대한 조사에 응하여 라인 에러 요청과 주기적인 간격을 두어 변수 LTIME을 파악한다. 모든 라인 에러 요청 패킷은 서버 변수 LTIME을 정한다. LTIME값이 0인 경우에는 자동 보고를 할 수 없고, FFFF의 값을 가질 때에 LTIME는 바뀌지 않는다.

<182> 클라이언트(18)가 0이 아닌 값으로 LTIME을 정하게 되면, 서버(20)는 LTIME밀리초마다 (약 20 밀리초의 정확성을 가지고) 에러 카운터의 상태를 검사하여 마지막 보고 이후 카운터에 변경이 있었으면 자

동 라인 에러 응답 패킷을 보낸다.

<183> 버퍼 요청 패킷은 클라이언트(18)가 서버 송신용 버퍼(42)와 수신용 버퍼(44)의 크기를 알고자 할 때 보낸다. 서버(20)는 버퍼 응답 패킷으로 응답한다. 오픈에 성공한 후와 데이터를 서버(20)에 보내기 전에 클라이언트(18)는 서버의 버퍼(42, 44) 크기를 확인해야 한다. 클라이언트(18)가 얼만큼의 데이터를 보낼 수 있는지를 알기 위해서는 송신용 버퍼(42)의 크기를 알아야 하고, 따라서 서버(20)가 흐름 제어의 수준을 지정하려면 수신용 버퍼(44)의 크기가 필요하다.

<184> 포트 용량 요청 패킷은 일반적으로 클라이언트(18)가 포트(40)의 하드웨어와 소프트웨어의 용량을 알고 싶을 때 보낸다. 다른 포트 수준 명령들과는 다르게, 이 명령이 내려질 때 포트(40)를 오픈할 필요가 없다.

<185> 전송율 지정 명령은 클라이언트(18)가 포트의 전송율, CFLAGS, IFLAGS, OFLAGS 및 XFLAGS를 지정하고자 할 때 사용한다.

<186> 클라이언트(18)는 어떤 ESTAT 및 MSTAT 조건으로 서버(20)로 하여금 이벤트 패킷을 보내게 할 것인가를 지정하기 위해 선택 이벤트 조건 명령을 보낸다. 클라이언트는 또한 상태 요청 패킷을 사용하여 ESTAT 및 MSTAT 조사를 할 수 있다.

<187> 윈도우 개시 지정 명령은 클라이언트(18)가 서버 상태 변수(38)의 하나인 TREQ를 정할 수 있도록 한다. TSEQ는 윈도우 시퀀스 패킷이 보내질 요청수준이다. TREQ는 출력 버퍼내의 데이터 범위밖에 있는 것이면, 서버(20)는 윈도우 시퀀스 패킷으로 즉시 응답한다.

<188> 지정 모뎀 출력 및 흐름 제어 패킷은 클라이언트(18)가 모뎀 출력을 지정하고 모뎀-신호 하드웨어 흐름 제어를 선택고자 할 때 보낸다. RTS 및 DTR 모뎀 흐름제어를 선택했을 때 MOUT 내 TRS 및 DTR의 값은 무시됨을 알아두자.

<189> 클라이언트(18)는 수신 흐름 제어의 수준을 정하기 위해 지정 수신용의 고/저 워터마크 지시 패킷을 보내고 지정 수신용의 고/저 워터 마크 지시 패킷의 흐름 제어를 설정한다. 서버(20)가 이를 처리하면 효과가 즉시 나타난다. 입력 버퍼의 문자수가 RHIGH를 초과하면, 흐름 제어가 즉시 시작된다. 만약 입력 버퍼의 문자수가 LHIGH 미만이면, 흐름 제어는 즉시 완화된다.

<190> 클라이언트(18)는 서버(20)에 소프트웨어 흐름 제어 문자를 지정하기 위해 흐름 제어 문자 패킷을 보낸다. 마찬가지로, 클라이언트는 RMAX 지정 명령이나 RTIME 지정 명령을 보내서 RMAX 및 RTIME 파라미터를 정하게 된다.

<191> 클라이언트(18)는 전송 문자 즉시 패킷(send character immediate packet)을 보내서 인-밴드 데이터에 앞선 문자를 보낼 수 있다. 즉시 데이터(immediate data)를 보낼 때, 소프트웨어 흐름 제어는 무시되고 하드웨어 흐름 제어가 변수 MFLOW 대신에 MCTRL를 사용하게 된다. 클라이언트(18)는 하드웨어 브레이크 신호를 보내야 할 때 전송 브레이크 즉시 패킷(send break immediate packet)을 보낸다. 소프트웨어 및 하드웨어 흐름 제어에 상관없이 이 브레이크 신호는 모든 인-밴드 데이터에 앞서 보내지게 된다.

<192> 클라이언트(18)는 서버 입력 및 출력 버퍼의 어느 하나 또는 모두를 확보할 때 플러쉬 버퍼 패킷(flush buffer packet)을 보낸다. 입· 출력 중단 패킷은 서버 내 소프트웨어 흐름 제어 지정을 중단시킨다. 수신 버퍼내에 있는 문자수가 RLOW 미만인 경우에는 입력을 중단하는 것이 비효과적이다.
마지막으로, 입· 출력 지속 패킷은 서버내 소프트웨어 흐름 제어 지정을 지속시킨다. 수신 버퍼내에 있는 문자 수가 RHIGH 이상인 경우에는 입력을 지속하는 것이 비효과적이다.

<193> 서버(20)가 클라이언트(18)에게 ESTAT 및 MSTAT 조건의 변경을 알리고자 할 때에는 이벤트 패킷을 보낸다. 서버(20)는 (ETRAN & EINT) 또는 (MTRAN & MINT)가 0이 아닐 때마다 각각의 포트(40)에 이벤트 패킷을 보낸다.

<194> 이 같은 요청 명령 효과가 틀림없이 이루어지도록 하고 만약 그렇지 않은 경우에는 이벤트 패킷에 보고하도록 하기 위해, 서버(20)는 먼저 다음과 같이 ELAST 및 MLAST를 정하고 클라이언트(18)에게 ELAST와 MLAST를 보고한다.

```
ELAST ^= (ESTAT ^ ELAST) | (ETRAN & EINT)
                         |
ETRAN = ESTAT ^ ELAST

MLAST ^= (MSTAT ^ MLAST) | (MTRAN & MINT)
                         |
MTRAN = MSTAT ^ MLAST
```

<196> 모듈 선택 패킷은 포트(40)의 특정 그룹을 지정하는 데에 쓰인다. 통신 밴드 폭을 줄이기 위해 포트(40)는 모듈로 논리적으로 그룹화되고 각 모듈은 16개의 포트(40)(0-15 포트)를 포함하게 된다. 논리적인 모듈 그룹화와 서버(20)상 포트(40)의 실제 배열은 서로 무관하다. 포트(40)를 모듈로 그룹화함으로써 모든 명령과 응답 레퍼런스를 지정하는 데에 현재의 모듈내에 어떤 포트(0-15)들을 사용할 것인가에 대한 정보만 있으면 된다. 모듈(0)은 디폴트로 선택된다. 모듈(0-7)은 1바이트 모듈 선택 패킷으로 선택할 수 있다. 모듈(8-255) 선택에는 2바이트가 필요하다. 모듈-포트쌍과 실제 포트번호를 서로 바꿀 수 있

다. 모듈 n이 선택되면 모든 종속 패킷내에서 포트 K는 16*n+K로 나타낸다.

<197> 클라이언트(18)나 서버(20) 중 어느 한 쪽이 원격으로부터 자기 정보를 얻고 싶을 때 ID 요청 패킷을 보내면 해당하는 ID 응답 패킷이 응답하게 된다.

<198> 디버그 패킷(debugging packet)은 서버가 클라이언트(18)와 서버(20)간의 디버그 데이터를 전송하는 것과 같은 디버그를 목적으로 포트(40)에 접근코자 할 때 필요하다. 이 명령에 의해, 클라이언트(18)는 서버 프로그램 메모리(36)에 접근하여 프로그램 메모리(36)를 검토하거나 여기에 저장되어 있는 서버(20)의 프로그램을 변경시킬 수 있다.

<199> 클라이언트(18) 또는 서버(20)가 접속을 해제하기 전에 원격으로 프로토콜 에러를 통지할 때에는 리셋 패킷을 보낸다. 패킷은 ASCII 포맷으로 리셋의 사유의 설명을 포함하고 있다. 리셋 패킷은 통신 중에 가장 마지막 수단인 것이다.

산업상이용가능성

<200> 본발명은 서버상의 포트의 엑세스를 가능하게하여 복수의 모뎀이나 프린터등의 공용장치로 공평한 엑세스가 될 수 있도록 네트워크상에 배치된 호스트 컴퓨터를 다중화 할 수 있고, 또한 하드웨어에 실장되어 기존의 호스트 컴퓨터의 소프트웨어와 양립성을 향상시켜서 호스트 컴퓨터의 오버헤드를 감소시킬 수 있으며 또한, 서버가 하나의 접속으로 다수의 포트에 데이터와 제어명령을 통신함으로써 네트워크상에서의 복수의 요청을 감소시킬 수 있다.

<201> 표 1. 서버 상태 변수

서버 상태 변수의 종류	빈도
클라이언트 접속	서버당
메시지 구축 간격	서버당
포트 오픈 상태	포트당
포트 전송율	포트당
처리 플래그	포트당
흐름 제어 문자	포트당
모뎀 제어	포트당
라인 에러 카운터	포트당
전송 브레이크 및 전송 즉시	포트당
이벤트 보고	포트당
입력 시퀀스 및 수신 버퍼 파라미터	포트당
출력 시퀀스 및 송신 버퍼 파라미터	포트당

<203> 표 2. XFLAG 설정

마스크	명 칭	설 명
0001	XPAR	패리티를 마크한다 혹은 공간으로 하는 것을가능하게 한다.
0002	XMODEM	인-밴드 모뎀 신호를 가능하게 한다.
0004	XCASE	출력에서 특수 문자를 다중-문자 시퀀스로 변환
0040	XTOSE	IXANY 캐릭터를 문자로 토스
2000	XIXON	흐름제어 캐릭터의 출력 소프트웨어의 제 2 설정을 가능하게 한다.

<205> 표 3. 모뎀 변수

이 름	설 명
MOUT	클라이언트가 지정 모뎀 출력 값을 특정한다
MFLOW	인-밴드 데이터의 모뎀 흐름 제어
MCTRL	아웃-오브-밴드 데이터의 모뎀 흐름 제어
MSTAT	현재 모뎀 상태
MLAST	클라이언트에 전송되는 최종 모뎀 상태
MTRAN	변경되었으나 변경 사항을 클라이언트에 아직 전송하지 않은 모뎀 신호세트
MINT	클라이언트에 보고될 모뎀 신호 변화

<207> 표 4. 모뎀 변수내의 모뎀 신호

마스크	명 칭	설 명
01	DTR	데이터 단말기 준비완료
02	RTS	전송 요청을 하다.
10	CTS	전송 해제를 하다
20	DSR	데이터 세트 준비완료
40	RI	링 인디케이터
80	DCD	데이터 캐리어 검출

<209> 표 5. 이벤트 보고 변수

명 칭	설 명
ESTAT	이벤트 상태의 현재 상태
ELAST	클라이언트에 전송된 최종의 이벤트 상태
ETRAN	전이가 있었으나 전이사실을 클라이언트에 보고하지 않은 이벤트 상태의 세트
EINT	발생되는 이벤트를 일으키는 이벤트 상태의 세트

<211> 표 6. 기록된 이벤트

마스크	명 칭	설 명
0001	OPU	클라이언트에 의해 무조건 중단된 출력
0002	OPS	정규의 소프트웨어 흐름제어로 중단된 출력
0004	OPX	특별한 소프트웨어 흐름제어 캐릭터로 중단된 출력
0008	OPH	하드웨어 흐름제어로 중단된 출력
0010	IPU	클라이언크에 의해 무조건 중단된 입력
0020	IPS	고/저의 워터마크에 의해 중단된 입력
0040	TXB	전송 브레이크 속행중
0080	TXI	전송 직접 속행중
0100	TXF	전송 흐름 제어 캐릭터 속행중
0200	RXB	브레이크 수신

<213> 표 7. 입력 시퀀스 및 수신용 버퍼 변수

명 칭	설 명
RIN	UART에서 읽혀져서 로컬 수신용 버퍼에 저장될 데이터의 다음 바이트 시퀀스 넘버
ROUT	수신용 버퍼로부터 클라이언트로 전송될 다음 바이트의 시퀀스 넘버
RWIN	클라이언트가 받고자 하는 최대 시퀀스 넘버
RMAX	수신용 최대 트리거
RTIME	수신용 트리거시간
RSIZE	로컬 수신용 버퍼의 크기
RLOW	소프트웨어적 흐름 제어 로우 워터마크
RHIGH	소프트웨어적 흐름 제어 하이 워터마크

<215> 표 8. 출력 시퀀스 및 송신용 버퍼 변수

명 칭	설 명
TSIZE	송신용 버퍼의 바이트 크기
TIN	송신용 버퍼에 저장될 데이터의 다음 바이트 시퀀스 넘버
TOUT	송신용 버퍼로부터 포트로 수신될 다음 시퀀스 넘버
TPOS	클라이언트에 마지막으로 보고된 TOUT의 값
TMAX	TOUT의 보고 최대치
TTIME	TOUT의 보고 시간 (밀리초)
TREQ	클라이언트가 요청한 시퀀스 넘버

<217> 표 9. 패킷의 종류

니블 0	니블 1	설 명
0	포트 0-15	데이터 패킷. 데이터 1바이트가 특정 포트에 뒤따름
1	포트 0-15	데이터 패킷. 데이터 2바이트가 특정 포트에 뒤따름
2	포트 0-15	데이터 패킷. 데이터 3바이트가 특정 포트에 뒤따름
3	포트 0-15	데이터 패킷. 데이터 4바이트가 특정 포트에 뒤따름
4	포트 0-15	데이터 패킷. 데이터 5바이트가 특정 포트에 뒤따름
5	포트 0-15	데이터 패킷. 데이터 6바이트가 특정 포트에 뒤따름
6	포트 0-15	데이터 패킷. 데이터 7바이트가 특정 포트에 뒤따름
7	포트 0-15	데이터 패킷. 데이터 8바이트가 특정 포트에 뒤따름
8	포트 0-15	데이터 패킷. 바이트1의 패킷은 특정 포트에 뒤따르는 데이터의 1-255바이트를 특정한다
9	포트 0-15	데이터 패킷. 바이트1-2의 패킷은 특정 포트에 뒤따르는 이 데이터의 1-65535바이트를 특정한다.
10	포트 0-15	윈도우 시퀀스 패킷 바이트1-2가 윈도우 시퀀스 넘버를 특정한다.

11	포트 0-15	커맨드 패킷 바이트1이 데이터길이를 결정하고 이것에 뒤따르는 데이터의 포맷을 결정하는 커맨드 넘버를 특정한다.

니블 0	니블 1	설 명
12	포트 0-15	이벤트 패킷. 상태 3바이트가 뒤따름
13	*	금지
14	*	금지
15	모듈 0-7	모듈 선택 패킷. 니블 1은 선택될 모듈 넘버포함
15	8	모듈 선택 패킷. 바이트1은 레인지 0-255로 모듈선택 코드를 특정한다.
15	9-10	금지
15	11	ID 요청 패킷 ID응답 요청
15	12	ID 응답 패킷 1바이트로 다음의 데이터 크기 결정
15	13	디버깅 및 제어 패킷.
15	14	디버그 패킷. 아주 무시될 다음 바이트의 수를 특정. 스트레스 테스트에 사용됨
15	15	리셋 패킷. 리셋의 사유를 설명하는 미완료의 ASCII 스트링에 의해 뒤따름

<221> 표 10. 커맨드 종류의 할당 및 길이

커맨드 종류 (1바이트 값)	커맨드 길이 (바이트)	커맨드 명칭
10	3	오픈 요청
11	6	오픈 응답
12	3	동기화 요청
13	3	동기화 응답
14	2	시퀀스 요청
15	6	시퀀스 응답
16	2	상태 요청
17	5	상태 응답
18	6	라인 에러 요청
19	14	라인 에러 응답
20	2	버퍼 요청
21	6	버퍼 응답
22	2	포트 용량 요청
23	32	포트 용량 응답

40	12	세트 바우드속도, CFLAGS, IFLAGS, OFLAGS 및 XFLAGS의 설정
42	5	이벤트 상황 선택
43	4	윈도우 트리거 설정
44	5	모뎀 출력 및 흐름 제어 설정
45	6	고/저 워터마크 설정
46	7	흐름 제어 캐리턱 설정
47	6	RMAX 및 RTIME 설정
48	6	TMAX 및 TTIME 설정
60	3	캐릭터 즉시 전송
61	4	브레이크 즉시 전송
62	3	입력/출력 확보
63	3	입력/출력 중단
64	3	입력/출력 지속

<224> 표 11. 오픈 요청 패킷

포지션	명 칭	설 명
니블 0	MTYPE	메시지 타입 11 · 커맨드
니블 1	PORT	포트 번호 0-15
바이트1	CTYPE	커맨드 타입 10 : 오픈 요청
바이트2	REQ	요청 타입 :
		0=즉시 오픈
		1=지연 오픈
		2=도래 오픈
		3=즉시 오픈 해제/종료
		4=지연 또는 도래 오픈 해제/종료

<226> 도표 12. 오픈 요청 패킷

포지션	명 칭	설 명
니블 0	MYYPE	메시지 타입 11 : 커맨드
니블 1	PORT	포트 넘버 0-15
바이트1	CTYPE	커맨드 타입 11 : 오픈 요청
바이트2	REQ	요청 타입 : 0=즉시 오픈 1=지연 오픈 2=도래 오픈 3=즉시 오픈 해제/종료 4=지연 또는 도래 오픈 해제/종료
바이트3	RESP	요청 타입 . 0=성공 1=포트 사용중 2=NO CARRIER 3=리소스 할당 문제, 재시도 4=요구는 이장치에서 지원 않됨
4-5 바이트	PNUM	본 요청으로 오픈한 실제 포트 번호 (16* 모듈+포트)

(57) 청구의 범위

청구항 1

복수의 통신 포트를 가지는 서버: 그리고

네트워크 접속을 통하여 서버로 호스트 컴퓨터를 통신가능하게 접속하는 드라이버를 가지는 호스트 컴퓨터를 구비하고,

상기 드라이버는 서버의 통신 포트의 각각을 위한 상응하는 로컬 통신 포트를 규정함으로써 서버

의 통신포트를 에뮬레이트하며, 또한 상기 드라이버는 서버의 통신포트가 호스트 컴퓨터에 직접 연결된 것처럼, 하드웨어 및 소프트웨어 흐름제어를 포함하여, 호스트 컴퓨터상에서 실행하는 응용프로그램이 서버의 통신 포트들 중 하나의 완전제어(full controll)를 허용하는 응용프로그래밍 인터페이스(API)를 포함함을 특징으로 하는 통신 시스템.

청구항 2

제 1 항에 있어서,

상기 드라이버는 응용프로그램이 드라이버로부터 부가적인 로컬통신 포트를 요청할 때 호스트 컴퓨터로부터 서버로 단일 네트워크 접속을 유지하게 구성된 통신 시스템.

청구항 3

제 1 항에 있어서,

상기 드라이버는 TTY장치를 로컬 통신포트로서 규정하는 통신 시스템.

청구항 4

제 1 항에 있어서,

상기 드라이버는 응용프로그램 인터페이스를 매개하여 응용프로그램으로부터 설정된 입/출력(I/O)을 수신하고, 또한 상기드라이버는 허용된 서버 통신포트의 하드웨어 특성을 구성하기 위하여 서버에 설정한 상기 I/O와 통신하도록 구성된 통신 시스템.

청구항 5

제 1 항에 있어서,

상기 서버 통신포트는 시리얼 포트인 통신 시스템.

청구항 6

제 1 항에 있어서,

상기 네트워크 접속은 TCP 접속인 통신 시스템.

청구항 7

제 1 항에 있어서,

네트워크를 통하여 신뢰할 수 있는 쌍방향 바이트스트림 접속과 같은 네트워크 접속을 확립하고 상기 드라이버에 결합된 제어장치를 오프닝하며, 신뢰할 수 있는 쌍방향 바이트스트림 네트워크 접속을 상기 드라이버에게 연결시키도록 STREAMS를 이용하기 위하여 호스트 컴퓨터에서 실행하는 유닉스 데몬(UNIX daemon)을 더 구비하는 통신 시스템.

청구항 8

네트워크 접속을 매개하여 호스트 컴퓨터에 통신 가능하게 결합된 원격서버의 복수의 통신포트를 에뮬레이터하는 드라이버를 포함하고, 상기 드라이버는 서버의 각 통신 포트를 위한 상응하는 로컬 통신포트를 규정하고, 서버의 통신 포트가 호스트 컴퓨터에 직접 연결된 것처럼, 하드웨어와 소프트웨어를 포함하여, 호스트 컴퓨터에서 실행하는 응용 프로그램이 서버의 통신포트들중 하나의 완전제어를 허용하는 응용프로그래밍 인터페이스(API)를 포함하여 이루어짐을 특징으로하는 호스트 컴퓨터용 하드웨어장치.

청구항 9

제 8 항에 있어서,

상기 드라이버는 표준 응용 프로그래밍 인터페이스(API)를 매개하여 호스트 컴퓨터에서 실행하는 응용프로그램으로부터 로컬 통신포트중의 하나를 제어하는 요청을 수신하도록 구성된 호스트 컴퓨터용 하드웨어 장치.

청구항 10

제 9 항에 있어서,

상기 드라이버는, 터미널 서버가 상기 터미널 서버의 통신포트들중의 하나로 독점적 엑세스를 호스트 컴퓨터에게 허용하도록 터미널 서버로 요청을 전달하게 구성된 호스트 컴퓨터용 하드웨어 장치.

청구항 11

제 12 항에 있어서,

상기 터미널 서버는 요청된 로컬 통신포트에 상응하는 서버 통신 포트의 독점적 엑세스를 허용하게 구성된 호스트 컴퓨터용 하드웨어 장치.

청구항 12

제 8 항에 있어서,

상기 드라이버는 응용프로그램이 부가적 로컬 통신포트를 요청할 때 호스트 컴퓨터에서 서버로

단일 네트워크 접속을 유지하게 구성된 호스트 컴퓨터용 하드웨어 장치.

청구항 13

제 8 항에 있어서,

상기 드라이버는 TTY 장치를 로컬 통신포트로서 규정하는 호스트 컴퓨터용 하드웨어 장치.

청구항 14

제 9 항에 있어서,

상기 드라이버는 응용프로그래밍 인터페이스를 매개하여 응용프로그램으로부터 설정한 입/출력(I/O)를 수신하고 또한 터미널 서버 통신포트의 하드웨어 특성을 구성하기 위해 터미널 서버에 설정한 I/O를 전달하게 구성된 호스트 컴퓨터용 하드웨어 장치.

청구항 15

제 8 항에 있어서,

상기 터미널 서버 통신포트는 시리얼 포트인 호스트 컴퓨터용 하드웨어 장치.

청구항 16

제 8 항에 있어서,

상기 네트워크 접속을 TCP 접속인 호스트 컴퓨터용 하드웨어 장치.

청구항 17

제 8 항에 있어서,

네트워크를 통하여 신뢰할 수 있는 쌍방향 바이트스트림 접속과 같은 네트워크 접속을 확립하고, 드라이버에 결합된 제어장치를 오프닝하며, 드라이버에 신뢰할 수 있는 바이트스트림 네트워크 접속을 연결시키도록 STREAMS 인터페이스를 이용하기 위하여 호스트 컴퓨터에서 실행하는 UNIX을 더 구비하는 호스트 컴퓨터용 하드웨어 장치.

요약

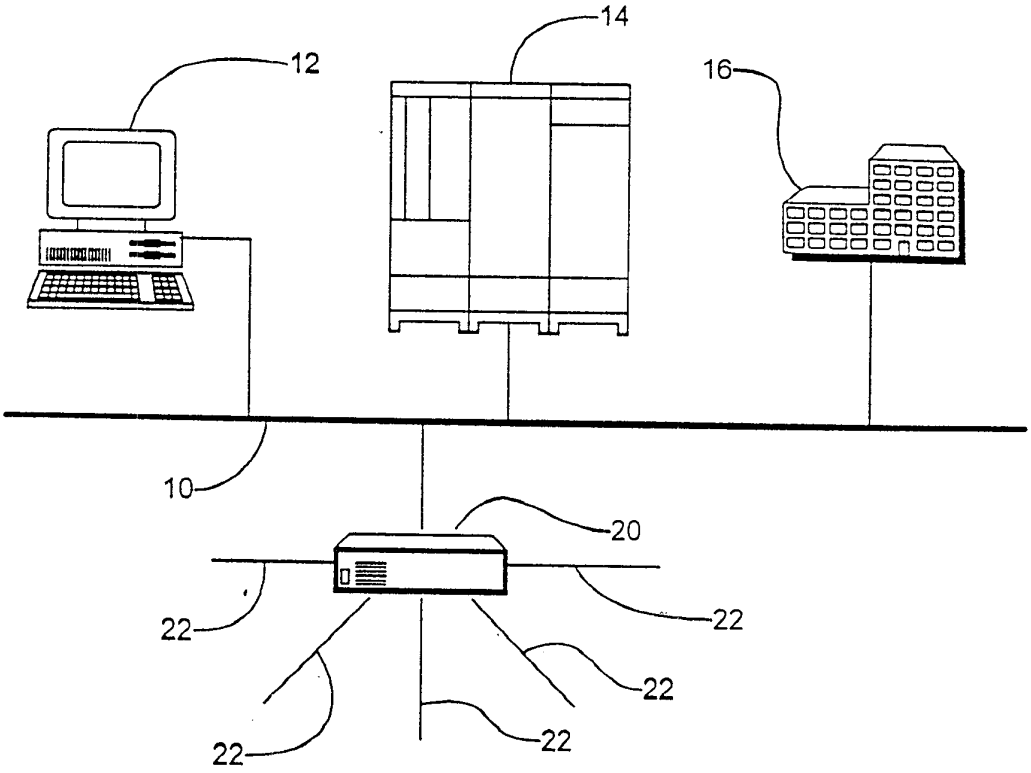
범용 네트워크(10)간에서, 호스트 컴퓨터와, 동기, 비동기 또는 병렬의 통신 포트와의 사이에서 통신을 행하는 터미널 서버(20)는, 하나의 독특한 장치 드라이버 인터페이스와 다중화 통신 프로토콜을 이용하도록 설계 되어 있다. 이 서버(20)는 하나의 접속으로 다수의 포트(40)에 데이터와 제어 병렬을 통신하며, 이것에 의해 네트워크상에서의 복수의 요청을 감소시킨다, 또한 전자의 TTY의 장치가 복수의 로컬 통신포트의 전체 특성을 가진 범용 네트워크 사이에서 이용되도록 만들어진다. 또한, 본발명은 서버(20)상의 포트(40)로의 엑세스를 가능하게 하여 복수의 모뎀이나 프린터등의 공용장치로 공평한 엑세스가 될 수 있도록 네트워크상(10)에 배치된 호스트 컴퓨터를 다중화한다. 그리고, 본발명은 하드웨어에 실장되어서 기존의 호스트 컴퓨터의 소프트웨어와 양립성을 향상시켜서, 호스트 컴퓨터의 오버헤드를 감소시킬 수 있다.

대표도

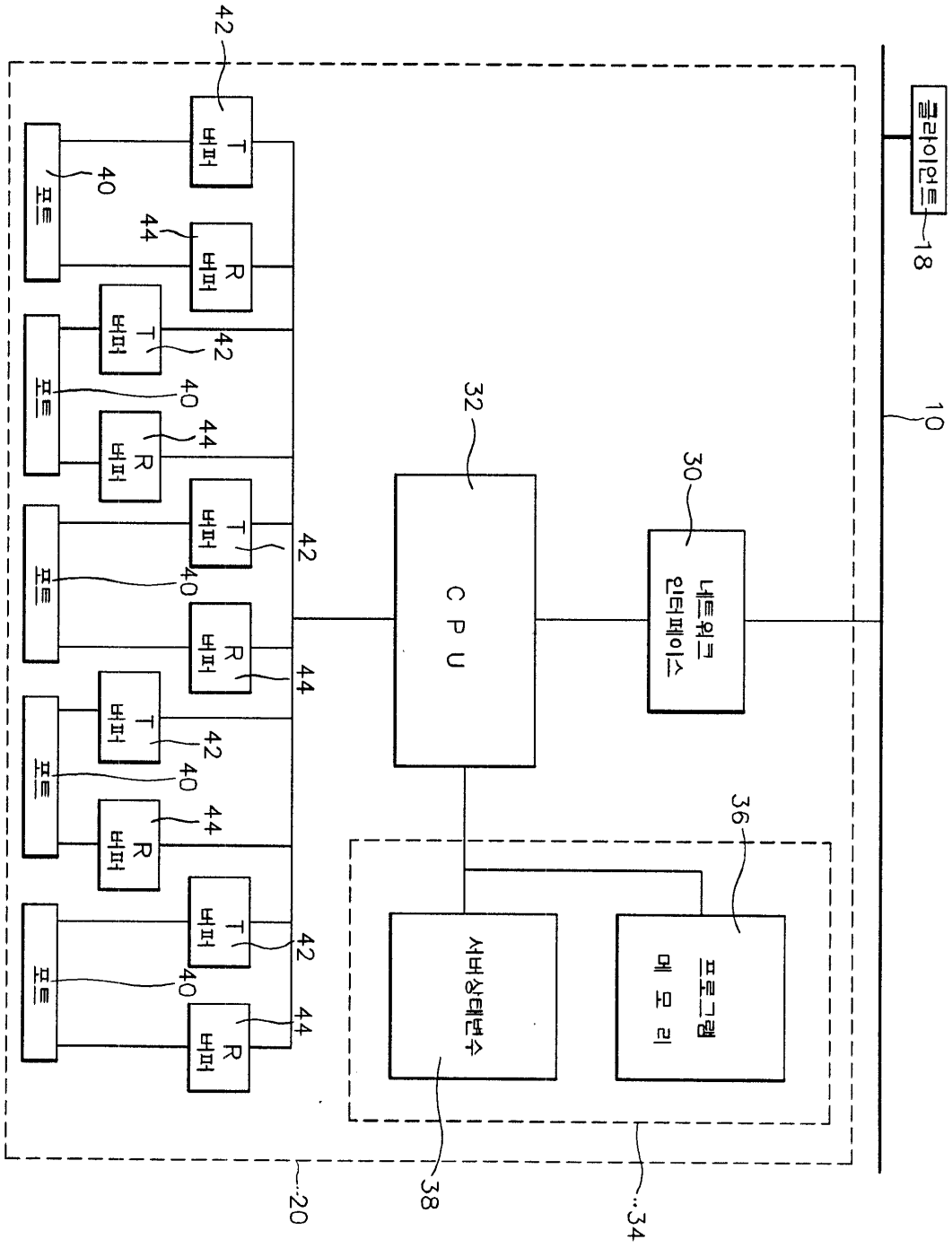
도1

도면

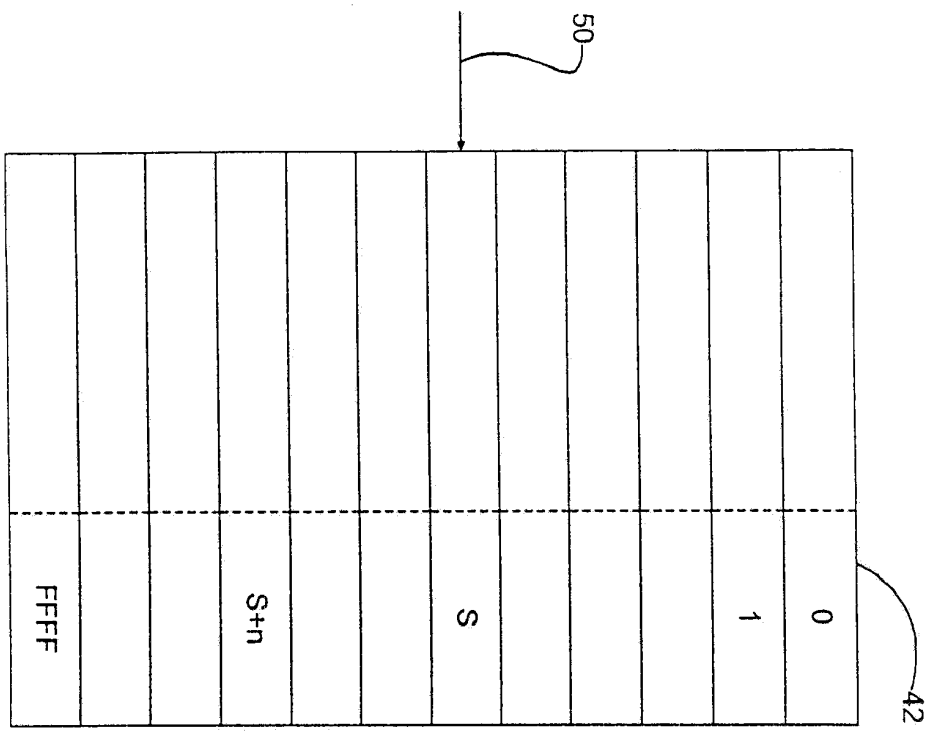
도면1



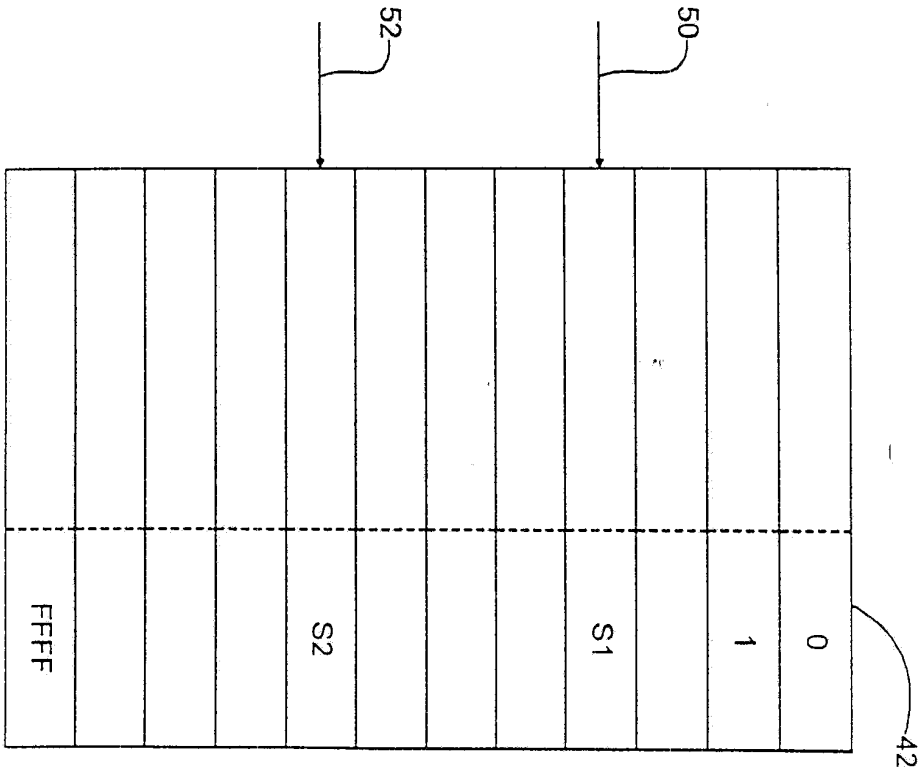
도면2


클라이언트
18
10
네트워크 인터페이스
30
CPU
32
프로그램 메모리
36
서버상태변수
38
34
20
포트
40
T 버퍼
42
R 버퍼
44

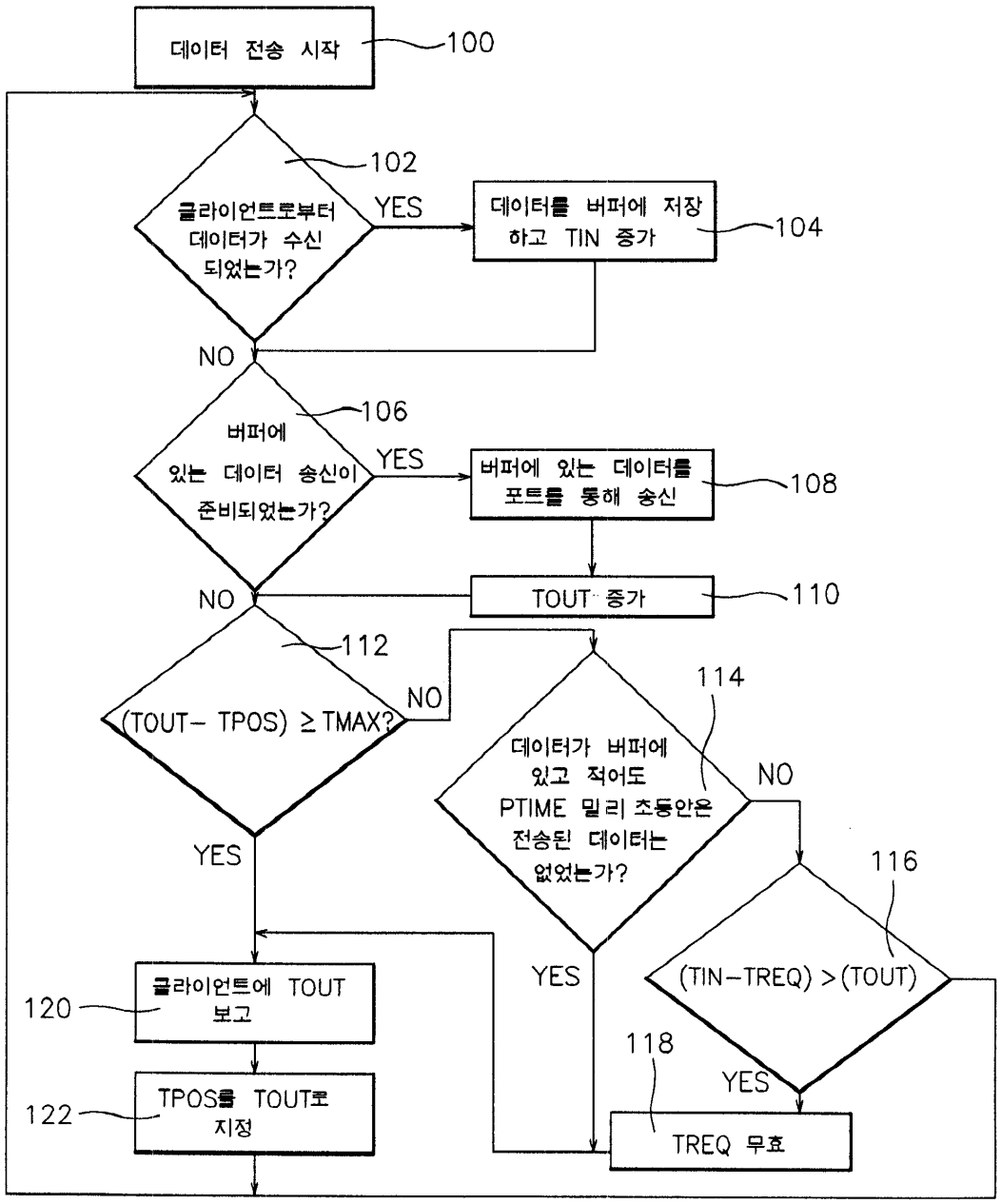
도면3



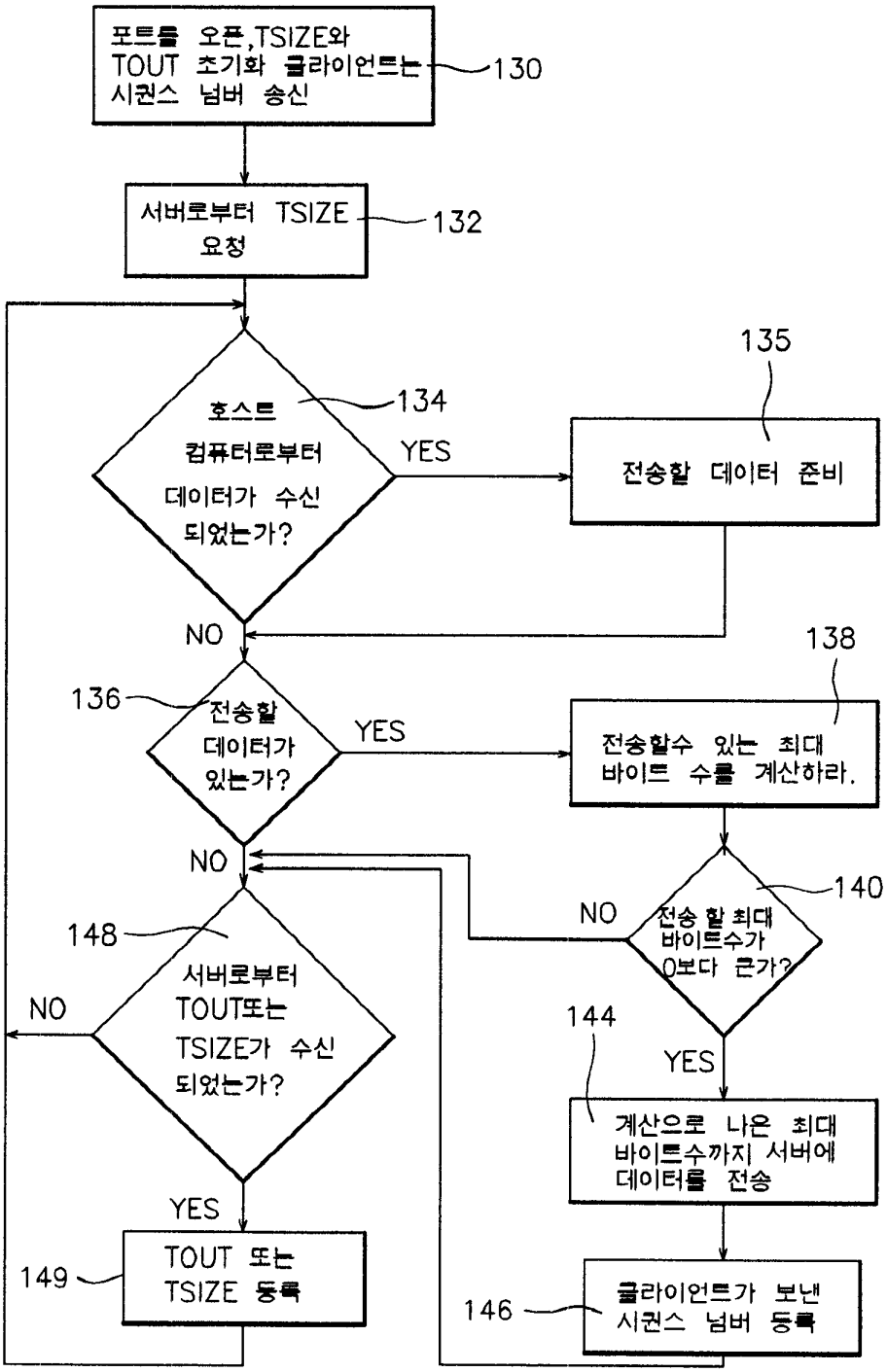
도면4



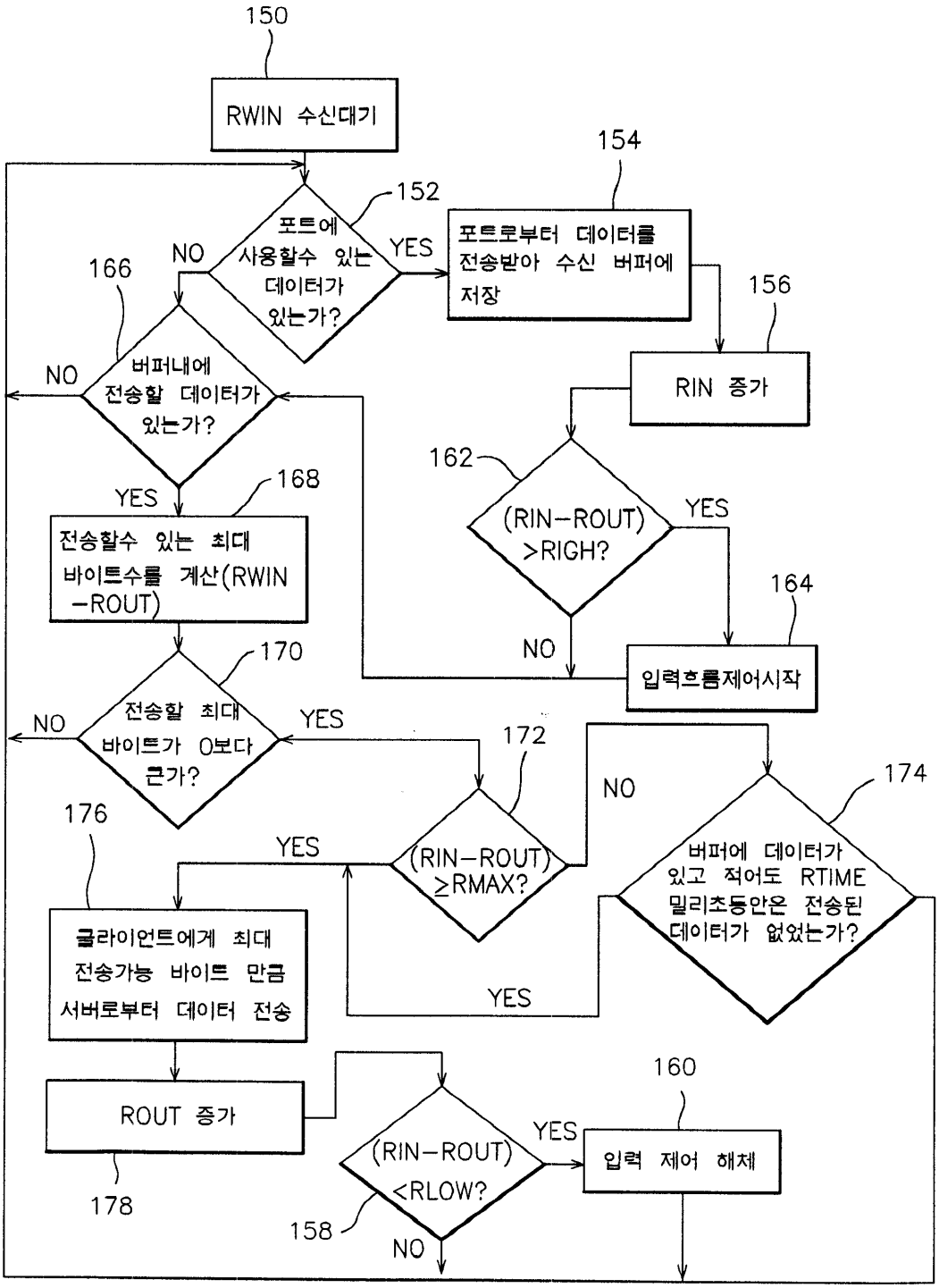
도면5


데이터 전송 시작
100
102
글라이언트로부터 데이터가 수신 되었는가?
YES
데이터를 버퍼에 저장하고 TIN 증가
104
NO
106
버퍼에 있는 데이터 송신이 준비되었는가?
YES
버퍼에 있는 데이터를 포트를 통해 송신
108
NO
TOUT 증가
110
112
(TOUT- TPOS) ≥ TMAX?
NO
114
데이터가 버퍼에 있고 적어도 PTIME 밀리 초동안은 전송된 데이터는 없었는가?
NO
116
(TIN-TREQ) > (TOUT)
YES
YES
118
TREQ 무효
120
글라이언트에 TOUT 보고
122
TPOS를 TOUT로 지정

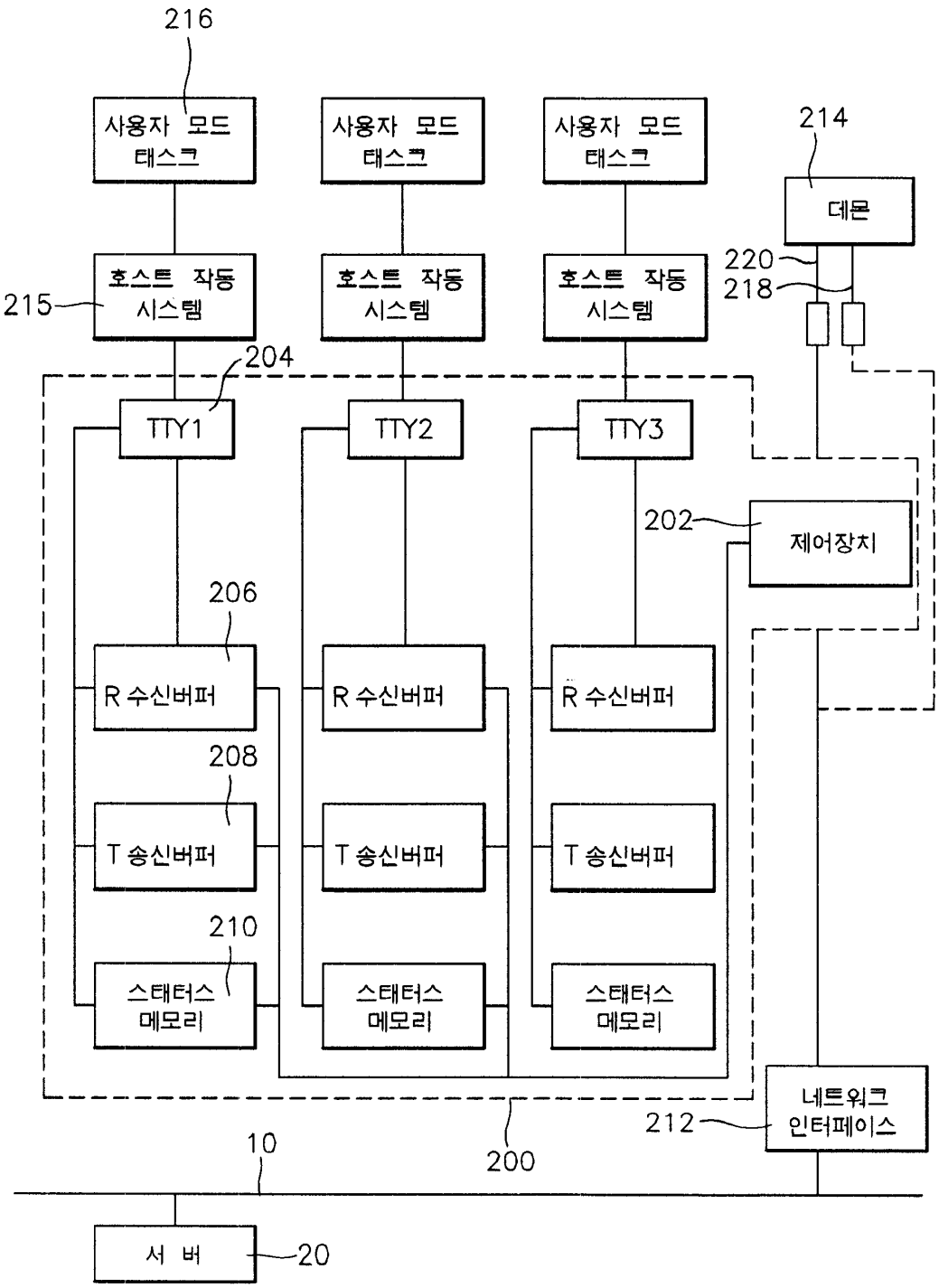
도면6


포트를 오픈,TSIZE와 TOUT 초기화 클라이언트는 시퀀스 넘버 송신
130
서버로부터 TSIZE 요청
132
호스트 컴퓨터로부터 데이터가 수신 되었는가?
134
YES
135
전송할 데이터 준비
NO
136
전송할 데이터가 있는가?
YES
138
전송할수 있는 최대 바이트 수를 계산하라.
NO
140
전송 할 최대 바이트수가 0보다 큰가?
NO
148
서버로부터 TOUT또는 TSIZE가 수신 되었는가?
NO
144
YES
계산으로 나온 최대 바이트수까지 서버에 데이터를 전송
YES
149
TOUT 또는 TSIZE 등록
146
클라이언트가 보낸 시퀀스 넘버 등록

도면7


150
RWIN 수신대기
152
포트에 사용할수 있는 데이터가 있는가?
YES
NO
154
포트로부터 데이터를 전송받아 수신 버퍼에 저장
156
RIN 증가
166
버퍼내에 전송할 데이터가 있는가?
NO
YES
162
(RIN-ROUT) >RIGH?
YES
NO
164
입력흐름제어시작
168
전송할수 있는 최대 바이트수를 계산(RWIN -ROUT)
170
전송할 최대 바이트가 0보다 큰가?
NO
YES
172
(RIN-ROUT) ≥RMAX?
NO
YES
174
버퍼에 데이터가 있고 적어도 RTIME 밀리초동안은 전송된 데이터가 없었는가?
YES
176
클라이언트에게 최대 전송가능 바이트 만큼 서버로부터 데이터 전송
ROUT 증가
178
(RIN-ROUT) <RLOW?
158
YES
NO
160
입력 제어 해체

도면8


216
사용자 모드 태스크
사용자 모드 태스크
사용자 모드 태스크
214
데몬
220
218
215
호스트 작동 시스템
호스트 작동 시스템
호스트 작동 시스템
204
TTY1
TTY2
TTY3
202
제어장치
206
R 수신버퍼
R 수신버퍼
R 수신버퍼
208
T 송신버퍼
T 송신버퍼
T 송신버퍼
210
스태터스 메모리
스태터스 메모리
스태터스 메모리
네트워크 인터페이스
212
200
10
서 버
20