(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3743963号
(P3743963)
(45)発行日　平成18年2月8日(2006.2.8)　(24)登録日　平成17年12月2日(2005.12.2)

(51)Int.Cl.　　F I
G06F 13/00 (2006.01)　　G06F 13/00 351A
H04L 29/10 (2006.01)　　H04L 13/00 309A

請求項の数 17　(全 33 頁)

(21)出願番号　特願平7-524142
(86)(22)出願日　平成7年3月15日(1995.3.15)
(65)公表番号　特表平9-510562
(43)公表日　平成9年10月21日(1997.10.21)
(86)国際出願番号　PCT/US1995/003183
(87)国際公開番号　WO1995/025311
(87)国際公開日　平成7年9月21日(1995.9.21)
審査請求日　平成14年3月14日(2002.3.14)
(31)優先権主張番号　08/213,197
(32)優先日　平成6年3月15日(1994.3.15)
(33)優先権主張国　米国(US)

(73)特許権者
ディジ　インターナショナル　インコーポレイテッド
アメリカ合衆国　ミネソタ　５５３４４，エデン　プレーリー，フライング　クラウド　ドライブ　６４００
(74)代理人
弁理士　亳　経夫
(74)代理人
弁理士　中村　壽夫
(74)代理人
弁理士　山田　卓二

最終頁に続く

(54)【発明の名称】リモートネットワーク装置を用いる通信システム及び方法

(57)【特許請求の範囲】

【請求項１】

複数の通信ポートを有するサーバーと、ネットワークを介してホストコンピュータを前記サーバーに接続するドライバを含む通信システムであって、

前記ドライバが前記サーバーの通信ポートの各々に対応するローカル通信ポートを形成することによって前記サーバーの通信ポートをエミュレートし、

さらに、前記ドライバは、アプリケーション・プログラム・インターフェース（ＡＰＩ）を含み、このインターフェースにより、前記ホストコンピュータ上で実行される応用プログラムが、前記サーバーの複数の通信ポートの１つを十分に制御可能であり、前記サーバーの複数の通信ポートが、ホストコンピュータに対してローカルであるようにハードウエアとソフトウエアのフロー制御を含んでいることを特徴とするシステム。

【請求項２】

前記ドライバは、応用プログラムが前記ドライバから付加的なローカル通信ポートを要求されたとき、前記ホストコンピュータから前記サーバーへの単一のネットワーク接続を維持することを特徴とする請求項第１項記載のシステム。

【請求項３】

前記ドライバは、ローカル通信ポートとしてＴＴＹ装置を形成することを特徴とする請求項第１項記載のシステム。

【請求項４】

前記ドライバは、アプリケーション・プログラム・インターフェースを介して応用プログ

ラムからの設定された入出力値（Ｉ／Ｏ）を受信し、更に、前記ドライバは、許可されたサーバーの通信ポートのハードウエア特性を形作るために前記サーバーに前記入出力値を通信することを特徴とする請求項第１項記載のシステム。
【請求項５】
前記サーバーの通信ポートは、シリアルポートであることを特徴とする請求項第１項記載のシステム。
【請求項６】
ネットワーク接続は、ＴＣＰ接続であることを特徴とする請求項第１項記載のシステム。
【請求項７】
さらに、ネットワーク上で信頼性のある双方向のバイトフロー接続として、前記ネットワーク接続を確立するために前記ホストコンピュータ上で実行するデーモン（214）を含み、該デーモンは、ＳＴＲＥＡＭＳインターフェース（218）を介して前記ドライバと連動する制御装置に接続し、前記信頼性があるバイトフロー接続のネットワーク接続を前記ドライバにリンクさせるために前記ＳＴＲＥＡＭＳインターフェース（218）を用いていることを特徴とする請求項第１項記載のシステム。
【請求項８】
ネットワーク接続を介して前記ホストコンピュータに接続されるリモートサーバーの複数の通信ポートをエミュレートするドライバを含む、ホストコンピュータ用のハードウエア装置であって、
前記ドライバは、前記サーバーの通信ポートの各々に対応するローカル通信ポートを形成し、さらに、前記ドライバは、アプリケーション・プログラム・インターフェース（ＡＰＩ）を含み、このインターフェースにより、ホストコンピュータ上で実行される応用プログラムが、前記サーバーの通信ポートの１つを十分に制御可能であり、前記サーバーの通信ポートが、ホストコンピュータに対してローカルであるように、ハードウエアとソフトウエアのフロー制御を含んでいることを特徴とするハードウエア装置。
【請求項９】
前記ドライバは、標準のアプリケーション・プログラム・インターフェース（ＡＰＩ）を介して前記ホストコンピュータ上で実行する応用プログラムからローカル通信ポートの１つを制御する要求を受信することを特徴とする請求項第８項記載のハードウエア装置。
【請求項１０】
前記ドライバは、ターミナルサーバーの通信ポートの１つに、ホストコンピュータに独占的にアクセスできるように、前記ターミナルサーバーにその要求を送受信することを特徴とする請求項第９項記載のハードウエア装置。
【請求項１１】
前記ターミナルサーバーは、要求されたローカル通信ポートに対応する前記サーバーの通信ポートに独占的にアクセスできるようにすることを特徴とする請求項第１０項記載のハードウエア装置。
【請求項１２】
前記ドライバは、応用プログラムが付加的なローカル通信ポートを要求するとき、前記ホストコンピュータから前記サーバーに単一のネットワーク接続を維持することを特徴とする請求項第８項記載のハードウエア装置。
【請求項１３】
前記ドライバは、ローカル通信ポートとしてＴＴＹ装置を形成することを特徴とする請求項第８項記載のハードウエア装置。
【請求項１４】
前記ドライバは、アプリケーション・プログラム・インターフェースを介して応用プログラムからの設定された入出力値（Ｉ／Ｏ）を受信し、更に、前記ドライバは、前記ターミナルサーバーの通信ポートのハードウエア特性を形作るためのターミナルサーバーに前記入出力値を通信することを特徴とする請求項第９項記載のハードウエア装置。
【請求項１５】

ターミナルサーバーの通信ポートは、シリアルポートであることを特徴とする請求項第８項記載のハードウエア装置。
【請求項１６】
ネットワーク接続は、ＴＣＰ接続であることを特徴とする請求項第８項記載のハードウエア装置。
【請求項１７】
さらに、ネットワーク上で信頼性のある双方向のバイトフロー接続として、前記ネットワーク接続を確立するために前記ホストコンピュータ上で実行するデーモン（214）を含み、該デーモンは、ＳＴＲＥＡＭＳインターフェース（218）を介して前記ドライバと連動する制御装置に接続し、前記信頼性があるバイトフロー接続のネットワーク接続を前記ドライバにリンクさせるために前記ＳＴＲＥＡＭＳインターフェース（218）を用いていることを特徴とする請求項第８項記載のハードウエア装置。
【発明の詳細な説明】
（技術分野）
この発明は、汎用ネットワークを通じての、コンピュータとコンピュータ周辺装置のようなディジタル装置間の通信分野に関する。
（背景技術）
互いに遠く離れたディジタルコンピュータとコンピュータ周辺装置間で通信を行うことの必要性が、長い間感じられていた。多くの技術がこの必要性のために設計されてきた。アナログ電話網上での長距離接続を確立するためのモデムの使用や、データ伝送媒体上でコンピュータと周辺装置との通信を行わせるネットワーキングを作ることによって行われた。
モデムは、二つのコンピュータ間での通信を確立させるために一対で用いられ、各モデムはアナログ電話網に接続されるばかりでなく、そのローカルコンピュータのシリアルポートに取り付けられている。シリアルポートはコンピュータに、モデムあるいはプリンタのような周辺装置へ連続的なビットの流れを通過させるようにする。コンピュータや他のマルチユーザコンピュータシステムに基づくＵＮＩＸ上で、この同じシリアルポートは、ユーザにコンピュータとの通信を許容するターミナルとの通信のために用いられる。結果として、単純なことであるが、遠隔地のターミナルが電話線上でマルチユーザコンピュータシステムと通信できるモデムを使用しているということになる。
最新のマルチユーザコンピュータは、同時に多数のユーザをサポートすることができるので、ターミナルがコンピュータと通信できる多くのシリアルポートを有することが望ましい。
あいにくにも、マルチユーザが、しばしば５０以上のターミナル接続をサポートしているので、シリアルポートのすべてを、直接コンピュータ上に取り付けることは論理的に困難である。結果として、マルチプレクサは、多数の内部的に基準となるシリアルポートからの連続トラフィックを単一の通信チャンネルに結合するために用いられていた。このように、ただ一つの通信ケーブルが、コンピュータに接続されることが必要である。
通信ケーブルの他端には、他の多重化装置があり、その装置が、結合されたデータトラフィックを個々のシリアルポートのためのデータに分離してそのシリアルポートに送り、それによってこのデータをアクセスできる。ポートコンセントレータと呼ばれるこのタイプの多重化装置は、コンピュータからリモート側のターミナルへ多数のシリアルポートを供給するために用いられる。主な通信ケーブル上のデータが、モデムを用いる電話線上で伝送されるようなシステム内での多重化を使用することさえ可能である。これは、コンピュータから非常に離れた所でも、多数のマルチプルシリアルポートへのアクセスを可能にする。
ネットワーク技術は、また、あるコンピュータと他のコンピュータあるいは周辺装置間での遠隔通信を可能にするために使用される。比較的、小エリア内でのコンピュータと周辺装置を接続するネットワークは、ローカルエリアネットワークあるいはＬＡＮと呼ばれている。汎用ネットワークは、標準ハードウエアや標準通信プロトコルを用い、マルチベン

ダの環境で操作される通信システムである。汎用ネットワークのハードウエアは、イーサネット(Ethernet)やトークンリング(Token Ring)のようなＬＡＮの技術を含む。標準ネットワークプロトコルは、ＴＣＰ／ＩＰやＳＰＸ／ＩＰＸを含む。
ＬＡＮは、一般的に伝送媒体によってコンピュータや周辺装置を接続することによって形成され、標準化された通信プロトコルに従って、伝送媒体上で通信が行われる。これらのプロトコルは、データがどのようにフォーマットされるべきか、またデータが特定の装置のためにどのように名付けられるべきかについて、必要条件を明らかにしている。データは、これらのプロトコルに適合しないとネットワーク上に送られない。
ホストオペレーテングシステム上のアプリケーションプログラムは、概して、ホストオペレーテングシステムに従うローカル通信装置のための、応用プログラムインタフェース（ＡＰＩ）に強く依存する。これは、ＵＮＩＸの従来のユーザインタフェースがターミナルコマンドライン操作であるので、ＵＮＩＸのようなシステム上で重要である。
汎用ネットワークは、コネクションレス アンリライアブル データグラム サービス(connectionless unreliable datagram service)や、信頼性双方向バイトストリーム接続の、二つの通信タイプをサポートしている。これら両サービスのＡＰＩは、シリアルポートのＡＰＩとは非常に異なっており、概して（ＴＴＹｓのように知られた）シリアル装置上で実行されるように書かれたプログラムは、正しく実行されるように適合層を必要とする。ＵＮＩＸの環境で、これは疑似ｔｔｙあるいはＰＴＴＹ装置として知られる仮想ＴＴＹ装置を用いるネットワークプロトコルであるテルネット(Telnet)やＲログイン(Rlogin)と共に、一般的になされる。ＰＴＴＹ装置は、実質、ソフトウエアであり、ハードウエアではなく、最大コマンドラインプログラムは、それらを充分に実行するようにＵＮＩＸ ＴＴＹｓをエミュレート（模擬）する。
これらＰＴＴＹｓと、実際のシリアルポートハードウエアを制御し、関連する真のＴＴＹｓと区別することは重要である。ＰＴＴＹは、スレーブ部とマスター部を有する。ＰＴＴＹのスレーブ部はＴＴＹエミュレーション装置であり、シリアルポートＡＰＩを機能させるようにプログラムによって使用される。マスター部は、ネットワークにデータを送ることができるテルネットやＲログインのようなネットワークプログラムとやりとりする。
ＰＴＴＹのスレーブ部は、ライン編集用キャラクタ、タブ拡張、及びローカルオペレーテングシステムのライン規則によってなされる他の機能のようなユーザオプションを変更することを要求する。しかしながら、それはボーレート、キャラクタサイズ、受け取ったエラー、ＢＲＥＡＫ信号等を変更することは無視する。
ＰＴＴＹのマスター部は、読み込み、書き込み、及びデータがスレーブとマスターの装置間で前後に通過する方法を変えるための非常に限定された入出力制御のセット（ｉｏｃｔｌｓ）を受け入れる。ＰＴＴＹのマスター部に書かれたデータは、あたかもシリアルポートに受け取ったかのように、ＰＴＴＹのスレーブ部に、受け取られたシリアルデータとして現れる。また、オペレーテングシステムのライン規則によって入力シリアルデータとして処理される。そのときデータは、ＰＴＴＹのスレーブ部から読み込まれるとき、受け取られたデータとして現れる。ＰＴＴＹのスレーブ部に書き込まれたデータは、ＵＮＩＸのライン規則によって連続出力データとして処理され、その後、マスター部に読み込まれる。
ＰＴＴＹのスレーブ部で行われたｉｏｃｔｌ操作は、マスター部では見ることができない。よってＰＴＴＹのマスター部をアクセスするプログラムでは、これらのコールがなされたことを検出することはできない。インタフェースは、ローカルオペレーテングシステムのライン規則では実行できないいくつかの操作をサポートするに充分でない。
多くのインプリメンテーション（実装）において、物理的装置とＰＴＴＹｓとの間に、１対１の対応はない。接続がテルネットあるいはＲログインによってＵＮＩＸシステムになされるとき、これらのプログラムは単にセッションのために次の利用できるＰＴＴＹを割り当てる。よってセッションからセッションへネットワークユーザは、異なったＰＴＴＹ装置を、明らかなパターンでなく、もっともらしく得られる。
これは、疑似ｔｔｙ装置に基づくシステムの安全化あるいはユーザオプションを実行する

ことを困難に、あるいは不可能にする。ある程度までこれを克服するために、特別なプログラムが、特別のＰＴＴＹのマスター部に所属するようなターミナルサーバーベンダによって、普通書かれる。そのＰＴＴＹのスレーブ部は、そのプログラムに所属されたままであり、他の使用に周期的に再割り当てされない。そのようなプログラムは、接続を受け入れ、ログインプログラムを実行するために用いられるが、しかし、しばしばダイアルアウトモデム、プリンタ、あるいは直結型シリアル装置に用いられる。この使用は、テルネットやＲログインより他の使用にＰＴＴＹを拡張する。しかし、真のＴＴＹ装置の制限されたエミュレーションのために、それらは変化したワークアラウンドを要求し、継続的なメインテナンスや両立性の問題を生じる。

テルネットは、また、ポートデータフローコントロールのためにＴＣＰ／ＩＰを使用し、データストリームの先頭データが処理されるまでコマンドが処理されないようにデータと連続してコマンドを送る。結果として、テルネットは、ユーザが割り込みを要求したような、あるコマンドを送るためにＴＣＰ／ＩＰの促進データ特性の使用を要求する。このインプリメンテーションは、中断が処理される前に、ユーザが中断した先頭データを捨てることを要求する。

あいにく、この実行はホストコンピュータばかりでなく、ネットワークそれ自体に非常に大きな負担を生じる。例えば、ネットワークに接続されたリモートターミナルでタイピング(typing)がなされると、個々のキーストローク(keystroke)が行われ、ネットワーク上のホストコンピュータに送られる。それから、ネットワーク上のリモートターミナルにエコーバック(echoed back)される。イーサネット ネットワーク上で、送信されたキャラクタが最少メッセージパケット６４バイトで送信されなければならない。又、アクノリッジ（承認）は６４バイトメッセージパケットである。そのように、二つの６４バイトパケットは、タイプされた各キャラクタに対して送信されなければならない。

さらに、ホストコンピュータによるキャラクタの受け取りは、キャラクタを受け取るためのＰＴＴＹ制御ソフトウエアを実行するタスクスイッチを要求し、また、エコーキャラクタを送信するためにＰＴＴＹ制御ソフトウエアへのタスクスイッチバックを要求する。

ターミナルサーバーと呼ばれる装置は、ネットワークのあるコンピュータへログされることをユーザに要求せずに、シリアルポートを通じてユーザにリモートログイン、テルネット及びＲログインの使用を可能にする。ターミナルサーバー上でシリアルポートに接続されるユーザは、ネットワークのマルチユーザコンピュータに接続するために、Ｒログインあるいはテルネットを使用することができる。

ターミナルサーバーのセカンドユーザは、また同じマルチユーザのコンピュータに接続可能であり、それによってそれに割り当てられたそれ自身のＰＴＴＹと第２の通信リンクを確立できる。テルネット、Ｒログイン、及び同様な手順を通じてターミナルサーバーによってなされた接続は、ネットワークの異なるコンピュータを通じてその手順を使用してなされた接続と同じ不利益を生じる。

（発明の要約）

本発明は、汎用ネットワークのリモートターミナルサーバーに配置された非同期ポートにホストコンピュータを全アクセスさせることによって、従来技術の不利益を克服する通信システムからなる。独特の装置ドライバーインタフェースを使用することによって、また通信プロトコルを多重化することによって、サーバーは単一の接続でマルチポートのためのデータ及び制御コマンドを送信することが可能である。それによって、ネットワーク上の要求を少なくする。

さらに、本発明はローカル通信ポートのすべての特性を有するネットワーク上の真のＴＴＹ装置を提供する。本発明は、モデムやプリンタのような共用装備に正しくアクセスしながら、ネットワークに配置された多数のホストコンピュータに、サーバー上のポートにアクセスすることを可能にする。結局、本発明はハードウエアにインプリメントされ、既存のホストコンピュータソフトウエアとの両立性を増進させ、さらにホストコンピュータのオーバーヘッドを減少させる。

本発明は、リモート通信ポートやそのようなポートを合体するサーバにアクセスする、あ

るいはアクセスを与えることを望むクライアント（通常ホストコンピュータ）間の汎用ネットワークによって供給される信頼性のある通信リンク上で動作する。
本発明は、汎用ネットワーク上の真のＴＴＹ装置が備えられた従来のＰＴＴＹ実装とは相違する。ＰＴＴＹ実装のようではなく、本発明のＴＴＹは１対１対応である。あたかも、シリアルポートがホストオペレーテングシステムに、直接に接続され、この使用に専念しているかのように、ホストオペレーテングシステムでそのハードウエアを制御させることができる。
それは、応用プログラムによってなされる仮定とを妨げることなく、本発明はこれら限定を緩める。そして、ネットワークの他のホストと共用してその機能を広げ、ＰＢＸ型追跡グループと動的ポートの割当てを共用する。本発明は、それぞれのオペレーテングシステムに真のＴＴＹｓと１対１の対応を維持させる。しかし、ＴＴＹｓが、本発明のオペレーテングシステムに使用されていないとき、他の目的に対してアクセスできるようにすることは任意である。
サーバー上で１あるいはそれ以上のポートを使いたいクライアントは、本発明の「通信プロトコル」によって準備されたプロトコルを使用するそのサーバーへ単一の接続をオープンする。クライアントと通信プロトコルを使用するサーバーとの間のすべての通信は、いかに多くのポートがクライアントによって使用されても、この単一接続を使用して続行する。クライアントとサーバ間の付加的接続がテルネットやＲログインのような従来技術のプロトコルのもとでオープンされてよい。しかし、これら従来技術のプロトコル接続の存在は、本発明の通信プロトコルの制御下でポートに影響しない。クライアントとサーバー間の接続が何らかの理由で破壊されるとき、通信プロトコル制御下ですべてのポートは閉鎖され、そのクライアントによって要求されているすべての資源はもどされる。
通信プロトコル接続の存在は、サーバー上のすべてのポート資源に、クライアントが自動的にアクセスすることを許さない。クライアントがポートにアクセスする前に、最初にポートを幸運にもオープンするに違いない。その後、クライアントがポートをクローズするまで、クライアントはポートへ排他的アクセスを許される。
一度、ポートがオープンされると、クライアントはボーレートを変更するために、要求を出してもよい。クライアントは、ポートにデータを送出し、そしてサーバーはポートに受け取られるすべてのデータを、クライアントに自動的に送る。通信プロトコル下で、クライアントはサーバーから最新のステータスを要求できる。最新のステータスは、要求すれば直ちに得られるか、あるいはクライアントは、クライアントの種々のステータスの状況の変化を知らせるようにサーバーに要求できる。
（発明を実施するための最良の形態）
図１は、ＵＮＩＸコンピュータ１２、デジタル エクィップメント社のＶＡＸコンピュータ１４および特別な場所で用いられる新規なＰＣのネットワーク１６を含む各種コンピュータを相互接続した、ネットワーク１０の概略を示したものである。
ネットワーク１０は、エラーフリー型バイトストリームのデータ伝送用に設けられたプロトコルであれば、ＴＣＰ／ＩＰ、ＳＰＸ／ＩＰＸあるいはＸ．２５等、標準的なネットワーク通信プロトコルを用いることができる。この通信プロトコルは、イーサネット（Ethernet）やトークンリング（Token Ring）のような特定の論理位相数学上で機能する。
図１に示した各種コンピュータとネットワークを接続する典型的なネットワークは、イーサネットのトップ上で作動するＴＣＰ／ＩＰプロトコルが用いられる。プロトコルのＴＣＰの部分は、バイトストリームデータを受け入れてパケットに変換する。ＩＰの部分はこのパケットを受け入れてデータグラムを形成し、その間にイーサネットがデータグラムを受け入れてフレームを形成する。本発明は、用いられるネットワークプロトコルには影響されないが、その説明の目的で典型的な形態が用いられている。
本発明のサーバー２０は、ネットワーク１０の種々のポート（図示を省略した）に接続するように作用する。これらのポートには、サーバー２０に対して複数の通信ライン２２を接続することができる。図１においては、たった５本の通信ライン２２しか示されていないが、サーバー２０は、典型的には１６本あるいはそれ以上をネットワーク１０に接続す

ることができる。
図２は、１つまたはそれ以上のクライアント１８と汎用ネットワークを介して接続されるサーバー２０との間の通信の主な要素を示したものである。クライアント１８は、ＵＮＩＸコンピュータ１２やＶＡＸコンピュータ１４、あるいはローカルネットワーク１６のようなホストコンピュータとすることができる。サーバー２０は、そのネットワークインタフェース３０を通じてネットワーク１０に接続される。ネットワークインタフェース３０は、ネットワーク１０上の他の装置と同様に、サーバー２０がネットワーク１０と通信するために必要とする産業用の標準ハードウエア及びソフトウエアを備えている。すべての産業において標準となるネットワークインタフェースと同様に、このネットワークインタフェース３０は、ネットワーク１０からデータフレームを受け取り、サーバー２０に宛てられたデータかどうかを判断し、もしそうならば、そのデータのネットワークフォーマットをはずしてフレーム、データグラムおよびパケットを組み立てることにより、バイトストリームに有効な形式のデータを作る。すなわち、本発明のネットワークフェース３０は、送信、ネットワーク、リンクの機能と国際標準化機構のオープンシステム相互接続（ＩＳＯ／ＯＳＩ）モデルの物理的な階層を備えている。
ネットワークインタフェース３０によるバイトストリームのデータは、本発明の通信プロトコルの装置を調整する中央処理装置（ＣＰＵ）３２に供給される。通信プロトコルの装置の調整に必要なプログラムモジュールは、サーバーメモリ３４のプログラムメモリ部３６に記憶されている。
ネットワークインタフェース３０から受け取ったデータをＣＰＵ３２が分析した後、生データはサーバー２０の適当なポート４０に通される。ポート４０は、非同期、同期あるいは並列ポートとすることができる。ポート４０に取り付けられるものは、プリンタやモデム等の各種装置とすることができる。
クライアント１８から送信されてポート４０に送られるデータは、ポート４０がデータを送信する準備をするまで、出力または送信バッファ４２に記憶される。各ポート４０を通ってサーバー２０に送られるデータは、同様にして扱われる。ポート４０から来てクライアント１８に送信されるデータは、クライアント１８にネットワーク１０が送信する準備をするまで、入力あるいはレシーブバッファ４４に記憶される。
本発明の通信プロトコルは、ＣＰＵ３２によって実行され、バイトストリームの伝送をネットワーク１０に必要とされるフレーム、データグラムおよびパケット内でフォーマットするステップは、ネットワークインタフェース３０によって処理される。
サーバメモリ３４は、通信プロトコルの装置に完全に必要な状態情報を備えたサーバーのステートバリアブル（状態変数）３８も備えている。本発明の望ましい実施例においてサーバーの状態変数３８に記憶される種類は、表１に示されている。
これらの変数形式は、多数のデータ構造で表され、サーバー２０の働きを制御すると共にクライアント１８とサーバー２０との間の通信を容易にするためのいろいろな方法で用いられる。これらの変数の目的を明らかにするため、以下にその詳細を説明する。
クライアント接続のリストは、通信プロトコルの下で開くサーバー２０への接続を一般的に行うクライアント１８のリストを保存している。好ましい形態では、サーバー２０は、８以上のクライアント接続を同時にサポートできることが望ましい。
メッセージビルドインターバルは、通信プロトコルに従って構成される、あるいはクライアント１８に伝送される情報が必要であるか否かを決定するために、サーバー２０がその状態を内部的に何回ポーリングを行うかを決めるためのポーリング間隔を制御する。好ましい形態では、このポーリングは、１秒あたりほぼ5回行われることが望ましい。その結果、ポーリングは最低でも２０ミリセコンド後には行われ、平均では１０ミリセコンド後に行われる。この間隔は、末端ユーザーに十分であり、ほとんどのスライディング式ウインドウプロトコルにも十分である。
メッセージビルトインターバルでのみ情報を伝送することにより、サーバー２０はそのインターバルの間に情報を蓄積することができる。これにより、ネットワーク１０とクライアント１８の双方の要求により制限されて、一度に１キャラクタのデータしか送れない場

合よりもより大きなパケットでデータが送られる。
ポートプンステート変数は、クライアント１８が既にオープンしたポート４０に接続され、あるいはクライアント１８がポート４０のオープンを待つ、複数のトラックのデータ構造となっている。すなわち、このデータは、オープンポート４０のために、一般にクライアント１８がポート４０を保持し、クライアント１８がオープン待ちリスト上にあるトラックを構築する。ポート４０のオープンのさらなる詳細を以下に説明する。
ポート・ボーレート変数は、ポート４０のボーレートを、例えば1,843,200というように、ポートボーレート変数によって分割されて記憶する。この表現は、５０ー１１５Ｋボーの標準的なボーレートをすべて表すために１６ビットだけ必要とする。
フラグ処理変数は、キャラクタサイズ、ハードウエア状態および各ポート４０の入力／出力キャラクタ処理を制御する。サーバーの状態変数に記憶されるフラグは４つの設定がある。すなわち、ＣＦＬＡＧＳ，ＩＦＬＡＧＳ，ＯＦＬＡＧＳ，およびＸＦＬＡＧＳ，である。ＧＦＬＡＧＳ，ＩＦＬＡＧＳ，ＯＦＬＡＧＳは、ＵＮＩＸオペレーテングシステムのターミオ（termio）インタフェースによって標準的に与えられており、本発明においても産業上標準的なものとして用いられている（ＵＮＩＸプログラマーマニュアルではTermio）。
ＣＦＬＡＧＳは、キャラクタサイズ、ストップビットおよびパリティを含むポート４０のＵＡＲＴハードウエアの設定を制御する設定を有している。サーバー２０の装置に理解させる実際のＣＦＬＡＧＳは、CBAUD,CSIZE,CSTOPB,CREAD,PARENB,PARODD,およびHUPCLを含んでいる。同様に、ＩＦＬＡＧＳの設定は、サーバー２０がポート４０から受け取ったデータの処理を制御する設定を有している。
ＩＦＬＡＧＳの設定は、ブレーク条件、エラーハンドリングおよびソフトウエアフロー制御設定の解釈上の制御を含む。サーバー２０で翻訳されるＩＦＬＡＧＳは、IGNBRK,IGNPAR,PARMRK,INPCK,ISTRIP,IXON,IXOFF,IXANY,DOSMODEである。
ＯＦＬＡＧＳは、ポート４０に出力されることによりサーバー２０のデータの処理を制御する。ＯＦＬＡＧＳの設定は、ＣＲおよびＮＬの設定やタブディレイといったことを制御し、OPOST,OLCUC,ONLCR,OCRNL,ONOCR,ONLRET,OFILL,OFDEL,NLDLY,CRDLY,TABDLY,ESDLY,VTDLY,FFDLYを占めるターミオによって定められる。
好ましい形態では、現在の通信設備ではもはや使用されない各種のフラグの設定は、サポートされていない。これらのフラグの設定は、OFILL,OFDEL,NLDLY,CRDLY,BSDLY,VTDLYおよびFFDLY、ＯＦＬＡＧＳからのすべてを含む。これらの各々の設定については、好ましい形態でわずかな修正のみを記載したが、本発明がこれらの記載された設定に限定されることはない。
ＸＦＬＡＧＳは、ターミオＬＦＬＡＧＳから取ったフラグの設定で、追加機能が加えられている。このＸＦＬＡＧＳの設定は、表２に記載されている。
ＸＦＬＡＧＳ変数（ＵＮＩＸラインの分野のXCASEおよびＬＦＬＡＧＳ変数を除く）は、ＵＮＩＸ機能の設定に拡がり、複数のオペレーテングシステムにインタフェースを拡張し、価値付加機能を備えている。ＸＦＬＡＧＳの設定は、ＣＦＬＡＧＳ、ＩＦＬＡＧＳおよびＯＦＬＡＧＳの設定と相互に作用する。この相互作用の詳細は、以下の通りである。
もし、ＸＦＬＡＧＳのＸＰＡＲ設定がリセットされ、あるいは基礎をなすハードウエアがマーク／スペースをサポートしていなければ、ＣＦＬＡＧＳのPARODDの設定がされている場合に残りのパリティを選択し、リセットされている場合にパリティさえも選択する。さもなければ、PARODDが設定されている場合にスペースパリティを、リセットされている場合にマークパリティを選択する。
もし、ＩＦＬＡＧＳのPARMRKおよびＸＦＬＡＧＳのXMODEMが設定されていれば、モデム信号の１またはそれ以上の変化が入力ストリームにFF80MM（書式送り）として入力される。なお、MMは、最新化されたモデム信号値である。
もし、ＸＦＬＡＧＳのXCASEが設定されていれば、所定のアルファベット化されない印刷キャラクタがＵＮＩＸ仕様の出力に相当する２つの文字に変換する。ＸＦＬＡＧＳのXTOSSは、制御の流れの間に出力を再開するキャラクタが無視されるかどうかを制御する。Ｉ

ＦＬＡＧＳのIXANYがリセットされると、キャラクタが受け取られたXONあるいはXXONと一致したときのみ、出力が再スタートされる。しかしながら、ＩＦＬＡＧＳのIXANYが設定されているときには、どのようなキャラクタを受け取っても出力を再開する。これにより、XTOSSが設定されていれば、出力を再開するキャラクタは捨てられる。もし、XTOSSがリセットされていれば、出力を開始するいかなる流れない制御キャラクタでもデータストリームに置かれる。
ＸＦＬＡＧＳのXIXONが設定されているときは、流れ制御キャラクタの出力の特別の設定が可能となる。XXOFFの受け取りは、これをXXONが再開している間に、出力を停止する。
フロー制御キャラクタ変数は、どのようなフロー制御キャラクタとなるかが決定される。２つのフロー制御キャラクタの出力の設定がサポートされ、１つのフロー制御キャラクタの入力の設定がサポートされている。さらに、フロー制御エスケープキャラクターは、インタープリテーションなしに、フロー制御キャラクターの入力を許すことをサポートしている。
フロー制御キャラクターは、1)フロー制御始動キャラクターの標準ソフトウエア（ＸＯＮ）、2)フロー制御停止キャラクターの標準ソフトウエア（ＸＯＦＦ）、3)フロー制御始動キャラクターの特別ソフトウエア（ＸＸＯＮ）、4)フロー制御停止キャラクターの特別ソフトウエア（ＸＸＯＦＦ）、5)フロー制御エスケープキャラクター（ＬＮＥＸＴ）を含むものである。
ＬＮＥＸＴキャラクタの受取後、次に受け取られるキャラクタはフロー制御キャラクタとして認識されない。そして両キャラクタは、入力ストリーム中に置かれる。
サーバー状態変数３８に戻ると、クライアント１８へのすべてのモデム制御ファシリティの準備は、８個のモデム制御変数により行われる。これら変数の各々はモデムの操作状態を表すために、マスクを使用する。７個のモデム変数は、表３のように明らかにされる。表３に示されるすべての変数は、表４に示されるビット割り当てを用いて、モデム信号のセットを示す。
異なったモデム制御変数の使用は、以下のとおりである。ＭＯＵＴモデム制御変数は、クライアント１８によって要求される直近のＲＴＳ及びＤＴＲの値を含む。これらの値は、サーバー２０への要求をすることによって変更される。ＭＦＬＯＷでのＲＴＳ／ＤＴＲピットがセットされるとき、ＭＯＵＴでの対応する値のセットは一時、無視され、これらの信号は、サーバー２０がデータを受け取る準備ができているかどうかを示す。受領が止まると、これらの信号はローの状態にされる。受領が再始動されると、信号はハイの状態にされる。
ＭＦＬＯＷは、「インバンド」出力データを禁止するために使用される。データ、コマンド、及びデータポート内外のデータの通常フローとして同じシーケンス内で取り扱われる応答は、インバンドデータと呼ばれる。データ、コマンド、及び通常データの先頭で伝送され、直ちに受け取られる応答は、アウトオブバンドデータと呼ばれる。ＣＴＳ／ＤＳＲ／ＤＣＤ／ＲＩのいずれかが、ＭＦＬＯＷにセットされるとき、そして、少なくとも対応するハードウエア入力信号がローであるとき、インバンド出力は禁止される。
フロー制御キャラクタ及びトランスミット イミーディエイト キャラクタ（即時伝送キャラクタ）は、フロー制御のソフトウエアあるいはＭＦＬＯＷによって、支配されない。それらは、ＭＣＴＲＬによって与えられるＣＴＳ／ＤＳＲ／ＤＣＤ／ＲＩフロー制御の束縛によって、伝送される。サーバー２０は、ＭＦＬＯＷにセットされないＭＣＴＲＬに何かのビットをリセットする。
ＭＳＴＡＴは、入出力モデム信号の現在の状態を含む。
ＭＬＳＡＴは、クライアント１８に報告されるべき直近のモデムの状態を含む。
ＭＴＲＡＮはクライアント１８に最後に報告されてから、変化があるときのモデム信号を含む。ＭＳＴＡＴがＭＬＡＳＴに等しくなることは可能であり、ＭＴＲＡＮにセットされたビットを有している。この状態は、ＭＳＴＡＴのいくつかの信号は、少なくとも一つのトランジションを作るということを示し、また、それからクライアント１８へ示した最後の値に戻ることを示している。６個のモデム信号のいずれかがＭＩＮＴにセットされる

とき、それらモデム信号の対応する変化は、クライアント１８に報告される。
本発明のサーバー２０は、ポート４０に生じる受信キャラクタエラーの数を見失わないように、サーバー状態変数３８内のポート統計カウンタ毎を維持している。これらのカウンタは、ラインエラーカウンタとして、参照される。異なったカウンタは、以下のカウントのそれぞれを追跡するために使用される：ＵＡＲＴオーバーランエラーの数、バッファオーバーフローエラーの数、フレーミングエラーの数、パリティエラーの数、及び受信されたブレーク（区切り）の数。これら変数のそれぞれは、ポート４０のカテゴリー内で起ったラインエラーの数を見失わないように、１６ビットラップアラウンドのカウンターである。
付加的なラインエラーカウンターの変数は、ミリセカンドでの報告時間を追跡する。もし、この変数がゼロならば、ラインエラーは自動的にクライアント１８に報告されない。さもなければ、サーバー２０は特定インターバルごとにカウンタを検査し、最後の報告があってから、変化があったかどうか、クライアント１８に完全な報告をする。
センドブレーク及びセンドイミーディエイト変数は、ポート４０へブレークあるいは他のキャラクタを直ちに送るかどうかを、クライアント１８に、サーバー２０へ要求することをさせる。センドイミーディエイト変数はクライアント１８が送ることを欲するキャラクタを含む、ワン キャラクタ バッファである。センドブレーク変数は、要求されたブレークのミリセカンドの長さを含む。特殊なケースは、センドブレーク変数の内容にある。ＦＦＦＦの値がセンドブレークに送られるとき、これは無限のブレーク時間を示す。００００がセンドブレークに送られると、進行中のいかなるブレークもキャンセルされる。サーバー２０に送られた他のいかなる値も、センドブレーク時間に単に加えられ、ブレークは二つの要求の結合された時間の間、実行する。
この発明は、ポートフロー制御のためにＴＣＰ／ＩＰフロー制御を使用しないので、クライアント１８あるいはサーバー２０によって受け取られたデータは、いつも直ちに実行される。結果として、インバンドデータの流れを分裂させずに、アウトオブバンドコマンドやデータを送出することは容易である。
サーバー２０は、またサーバー状態変数３８のステータス状況のリストを維持している。これらのステータス状況は、イベント報告変数内に記憶され、クライアント１８に一般的に関心のある状況を示している。クライアント１８は、これらに登録することができ、あるいはサーバー２０が、いつ状況が変化しても、適当なイベント情報を自動的に送ることを要求することができる。イベント報告変数は、サーバー２０にこのイベント情報の送出を追跡することをさせる。表５は、種々のイベント報告変数を示している。
ＥＩＮＴはサーバー２０によってクライアント１８に送られるイベント情報を生じる所定条件を創出するために、クライアント１８によって送られることができる。イベント情報は、（ＥＬＡＳＴ及びＥＩＮＴ）がゼロでない時はいつでも、クライアント１８に送られる。
ＥＳＴＡＴがＥＬＡＳＴに等しく、また、ＥＴＲＡＮ内にビットセットを有しているままである時、可能であることを記しておく。この条件はＥＳＴＡＴ内のある条件が、少なくともワン トランジッションを作り、それからそれらがクライアント１８に報告できる前に、それらのＥＲＡＳＴの値にもどることを示している。
これらの変数に記録されるイベントは、指摘されるイベントがどのようなものであるかを示すマスクに沿って、表６に示されている。表６にあげたイベントは、この発明の文脈内でよく理解される通信イベントである。
サーバー状態変数３８内に記憶された最後の二つの変数は、入力シーケンス及び受信用バッファ・パラメータ変数であり、また関連する出力出力シーケンス及び伝送用バッファ・パラメータである。これら変数は、伝送用バッファ４２及び各ポート４０と関連する受信用バッファ４４の状態を示す。
バッファ４２、４４のデータの流れを通信するために、サーバー２０の好ましい実施例は、１６ビット・ラップアラウンド・シーケンスナンバーを使用している。二つのシーケンスナンバーは、各バッファは４２，４４と関連している。第１のシーケンスナンバーは、

バッファ内におかれた次のバイトのシーケンスナンバーを伝達する。この最初のシーケンスナンバーは、伝送用バッファ４２のためのＴＩＮ及び受信用バッファ４４のためのＲＩＮというラベルである。第２のシーケンスナンバーは受信用バッファ４４のためのＲＯＵＴ及び伝送用バッファ４２のためのＴＯＵＴというラベルである。
これらシーケンスナンバーは、ゼロで始まり、送出されるあるいは受け取られるデータの各バイトに対して一つ進む。シーケンスナンバーがＦＦＦＦに達すると、データの次のバイトが送出され、あるいは受け取られるとき、それらはゼロにラップバックする。変数ＴＩＮ、ＴＯＵＴ、ＲＩＮ、及びＲＯＵＴは、クライアント１８とサーバー２０の間で通信するために用いられるシーケンスナンバー変数である、ということが注意されるべきである。それらは、伝送用及び受信用バッファ４２、４４内のデータの物理的位置を特定しない。これらバッファは周知の技術であり、ここでは特に記載しない。
これらシーケンスナンバー、そして入力シーケンス及び受信用バッファパラメータや、出力シーケンス及び伝送用バッファパラメータの変数型をまとめる他の変数は、表７及び８に示されている。
ＴＩＮ、ＴＯＵＴ、ＲＩＮ及びＲＯＵＴのようなラップアラウンドシーケンスナンバーは、いくつかの非常に有効な数学的性質を有する。
しかしながら、シーケンスナンバーはこれらの関数の特別な定義がラップアラウンド特性の結果として使用されなければならないので、加減が行われるとき、特に混乱する。
それらの用途を明らかにするため、伝送用バッファ４２の実施例が図３及び４に図示され、検討されている。実施例３において、ＴＯＵＴシーケンスポインタ（バッファ４２からポート４０へ送出されるデータの次のバイトを示している。そして、矢印５０によって図３に示される。）は、バイトＳにセットされる。ｎバイトのデータが送出されるとき、送出されたバイトは、Ｓ＋（ｎ－１）であるけれども、ナンバーＳとされる。約束によって、われわれは、バッファがＳがＳ＋（ｎ－１）との間の送出バイトを有するということができる。しかしながら、われわれはＯｘＦＦＦＦでラップするシーケンスナンバーに通常の数を加えているので、われわれは次の公式を用いる表現Ｓ＋ｎを計算しなければならない：
（Ｓ＋ｎ）＆０ｘＦＦＦＦ
ここで、＆は、理論的ＡＮＤ操作を示す。このようにして、Ｓが０ｘＦＦＦ８で、ｎが５であれば、Ｓ＋ｎを計算する公式は、０ｘＦＦＦＣを与える。同様に、Ｓが０ｘＦＦＦ８でｎが９であれば、Ｓ＋ｎは２である。
ｎバイトを伝送した後、ＴＯＵＴシーケンスポインターは、伝送されるべき次のバイトの新しい位置に前進させなければならない。これは、古いＴＯＵＴの値を計算された結果Ｓ＋ｎに置き換えることにより行われる。
２つのシーケンスナンバーの間に存在するバイト数を決定するために、引き算が用いられる。しかしながら、特別の規則では、あるシーケンス数を他のシーケンス数から減算する結果を決定することは用いるに違いない。シーケンスナンバーs_1，s_2の間のバイトの数を決定するために、次の公式が用いられる。
（s_1-s_2）＆0xffff
この公式は、シーケンスナンバーs_1がシーケンスナンバーs_2よりも数値的に大きい場合には有効である。例えば、s_1＝0xffffかつs_2＝３であるならば、その時、（s_1-s_2）＆0xFFFFは、0x13が正しい答である。
実際、シーケンスナンバー間の距離の概念は、次のように定義すると有効となることが重要である。
ＤＩＳＴ(from, to)＝(((to)－(from))＆0xffff)
一対のシーケンスナンバーs_1，s_2を与えると、次の式の条件では、s_3は、シーケンスナンバーs_1，s_2の間のバイトのシーケンス位置にあることは真である。
ＤＩＳＴ（s_1，s_3）＜ＤＩＳＴ（s_1，s_2）
この公式は、s_3は、s_1とs_2の間にあり、s_1からs_3までの距離がs_1からs_2までの距離よりも小さい場合に当たる。

シーケンスナンバーは、データの量、すなわち、リモートバッファ内の自由空間の量を伝達できる。図４に示す特定ポート４０に対する伝送用バッファ４２において、ＴＯＵＴシーケンスナンバー５０がs_1に設定され、一方ポート４０から伝送されるべき次のバイトを表すＴＩＮシーケンスナンバー５２がs_2に設定される。図４において、ＴＯＵＴとＴＩＮの両方は、伝送用バッファ４２における位置に矢印で示されている。伝送用バッファ４２におけるバイトの数は、そのポートに対して決めることができる。

ＤＩＳＴ（s_1，s_2）

さらに、そのポートに対する伝送用バッファが、固定サイズＳＩＺＥを有する場合、そのバッファ内の自由バイト数は、次式で定まる。

ＳＩＺＥ－ＤＩＳＴ（s_1，s_2）

本発明におけるクライアント１８とサーバー２０は、ラップアラウンド・シーケンスナンバーの特性を利用して、両者間の情報のバンド単位の流れを制御することができる。クライアント１８とサーバー２０は、予めそうすることが定められているときにのみ、インバンドーデータを他方に送る。この予権限は、データの量を伝送することにより完成するが、他方はそのデータ量に伝送することが許されている。送り手がこのスペースを満たすのに十分なデータを送ったとき、この送り手は、受け手がいくらかのデータを移送し、かつ送り手に追加スペースが利用できることを知らせるまで、伝送を休止する。

クライアント１８からサーバー２０への流れを制御するために、サーバー２０は、クライアント１８にバッファに伝送するサイズを与え、また伝送用バッファ４２(TOUT)における現在の第１バイトのシーケンスを与える。クライアント１８は、クライアント自身のシーケンスナンバーを送り続けながら、以前にサーバーに送ったデータのトラックを保持する。この送られたシーケンスナンバーを用いることによって、そのサーバー２０に与えられた情報に沿って、クライアント１８は、サーバーの伝送用バッファ４２内に残った空のスペースの量を計算することができる。それゆえ、クライアント１８は、その空きスペースに適合するデータ以上のデータを送らなくなる。

このプロセスは、図５のフローチャート図に示されている。伝送スライドを始めるための指示は、図のボックス100に示されている。第１ステップは、データがクライアント１８からサーバー２０のネットワークを介して受信されたか否かを決定する。もしそうであるなら、ボックス102に示すようにデータを受け取り、そしてそのデータを最適なポート４０に対する伝送用バッファ４２に格納される。データは、伝送用バッファに格納されると、ボックス104に示すように、ＴＩＮは、インクリメントされる

データをクライアント１８から受信したか否かに係わらず伝送用バッファ４２内にあるデータは、ポート４０から送り出すことができる。データがある場合、バッファ４２内のデータは、ポート４０上に送り出さなければならない(ボックス108)。そして、ＴＯＵＴは、インクリメントされる(ボックス110)。

この時点で、サーバー２０は、ＴＯＵＴをクライアント１８に報告することが必要であるかどうかを決定しなければならない。判断ボックス112,114,116によって示された３つの判断は、好ましい実施例において利用される。第１の判断112は、伝送されるべき次の出力バイトを示すＴＯＵＴとＴＰＯＳとを比較する。ＴＰＯＳは、出力シーケンス及び伝送用バッファの変数パラメータの１つで、サーバーの状態変数３８内に格納されている。ＴＰＯＳは、クライアント１８に報告された最終ＴＯＵＴの値を示す。上記減算公式のシーケンスナンバーを用いて計算されたＴＯＵＴとＴＭＡＸ間の差が、ＴＭＡＸよりも大きい場合、ＴＯＵＴはクライアントに報告される。ＴＭＡＸは、サーバーの状態変数３８内に格納され、そしてクライアント１８によって、ＴＯＵＴが報告された時点で伝送間隔を決定するために設定される。ＴＯＵＴがクライアント１８に報告されているときは、いつもボックス120で示すように、ＴＰＯＳがその時ボックス122でＴＯＵＴの値が設定される。ボックス114において、ＴＯＵＴを報告するための第２の判定は、データがバッファ４２内に少なくともＴＴＩＭＥミリセコンド前にあったかを決定し、そして、時間ＴＯＵＴが報告されなかったので、サーバーの状態変数３８であるＴＴＩＭＥが、クライアント１８によって設定される。その場合、ＴＯＵＴはクライアント１８に報告される。

最後の判定116は、ＴＯＵＴの値とＴＲＥＱの値とを比較する。サーバーの状態変数３８の１つとして格納されたＴＲＥＱは、クライアント１８に設定され、その結果、ＴＯＵＴはＴＲＥＱを通過するとき、ＴＯＵＴはクライアント１８に報告される。数学的に、ラップアラウンド・シーケンスナンバーＴＯＵＴとＴＲＥＱを比較すると、上記シーケンスナンバーの減算公式を用いることにより、(TIN－TREQ)が(TIN－TOUT)より大きいか又は等しいかどうかを比較して両者の関係を判定することが必要である。ＴＯＵＴがＴＲＥＱを通過した場合、その時、ＴＲＥＱは、無効にされ（ボックス118）、ＴＯＵＴはクライアント１８に報告される(ボックス120)。

クライアント１８のドライバソフトウエアにおける柔軟性を許容するために、クライアント１８はそうすることが便利である場合には、クライアント１８がサーバーの伝送用バッファ４２がオーバーフローでない限りは、サーバー２０にデータを送ることができる。伝送用バッファ４２がオーバーフローでないことを確実にするための手順は、図６に示されている。

クライアント１８がデータをポート４０に伝送することを望む場合、クライアント１８用の通信プロトコルを処理するクライアント１８側のドライバは、サーバー２０がポートを開くことを要求しなければならない。うまく開くと、クライアント１８は伝送用バッファ４２のサイズＴＳＩＺＥがオープンポート４０に接続され、ＴＯＵＴはゼロとなる。

さらに、クライアント１８は、それ自身の伝送用シーケンスナンバーをリセットし、このシーケンスナンバーは、クライアント１８側に格納され、かつ維持されたラップアラウンドシーケンスナンバーである。

クライアント１８は、その時ボックス132で示すように、サーバーからＴＳＩＺＥを要求する。クライアントは、ボックス140で示すように、ＴＳＴＺＥが以下で説明するようにサーバー２０から受信するまで実際にデータを送ることができない。

クライアント１８は、次に、ホストコンピュータからデータを受信したかどうかを判断する(ボックス134)。データが受信されると、データは伝送用として待機しなければならない(ボックス135)。この伝送の待機については、詳細に後述される。

クライアント１８は、次にボックス136で送られるべきデータがあるかどうかを判定する。送るべきデータが準備されている場合、クライアントのドライバは、伝送用バッファ４２をオーバーフローさせることなくサーバー２０が受け入れることができるバイトの最大数を計算しなければならない(ボックス138)。これは、クライアントの伝送用シーケンスナンバーと最も近時に報告されたＴＯＵＴの値とを比較し、そして、この差を、ＴＳＩＺＥから減算する。

送るべき最大バイト数＝

ＴＳＩＺＥ－（伝送シーケンスナンバー－ＴＯＵＴ）

もちろん、シーケンスナンバーＴＯＵＴのクライアントの伝送シーケンスナンバーからの減算は、減算シーケンスナンバーのための公式に従って行わなければならない。もし、送るべきバイトの計算最大値が、ゼロよりも大きい場合(ボックス140)、バイトの計算最大値まで、サーバー２０に伝送することができる(ボックス144)。クライアント１８の伝送シーケンスナンバーは、その時、送られたバイト数によりインクリメントされる(ボックス146)。

データが送られたか否かに係わらず、クライアント１８は、サーバー２０がボックス148で示すように、ＴＯＵＴまたはＴＳＩＺＥに対する値を伝送したかどうかを次に判定しなければならない。

その場合、クライアント１８で格納されたＴＯＵＴまたはＴＳＩＺＥの値が最新である(ボックス149)。

送信機がだめで、それが報告がされずに、偽りの報告を避けるために、サーバー２０は、非常にまれなケースを認識する。伝送用バッファ４２が空のときで、ＵＡＲＴがまだインバンドーデータを送るのに忙しいときには、サーバー２０は、ＴＯＵＴの代わりに(TOUT－1)を報告する。この変換により、クライアント１８は、サーバー２０に伝送された最後のデータの伝送シーケンスナンバーに等しいＴＯＵＴをクライアントが受けるとき、全て

のデータをうまく送ることができるようすることができる。
ポート４０から受け取られ、かつサーバー２０からクライアント１８に伝送される処理は、図７に示されている。このサーバー２０は、サーバーがボックス150で示すようにＲＷＩＮに対する値を受け取るまで、そのデータをクライアント１８に伝送することを許されていない。ＲＷＩＮは、クライアント１８により送られる入力シーケンス及び受信用バッファ変数であり、クライアント１８が受入を望む最高のシーケンスナンバーを示すものである。
初期ＲＷＩＮがクライアント１８から受信した時、サーバー２０、入来データが受信用バッファ４４内に格納されるべく利用できるかどうかを判断するために、ポート４０を調べる(ボックス152)。データが受信されると、受信用バッファ４４内にデータが格納され、そして、ＲＩＮシーケンスナンバーは、従ってインクリメントされる。これは、ボックス154とボックス156に示されている。
クライアント１８が如何に早くサーバー２０からデータを受信できるかということはサーバー２０にとっては知ることができないが、サーバー２０は、ポート上の入力フロー制御を実現する能力を有している。
フロー制御は、受信用バッファ４４内に残るデータの量（(RIN－ROUT)として計算される）がもう１つの入力シーケンス及び受信用バッファのＲＨＩＧＨの値を越える時に開始される。こうして、データが受信用バッファ４４内に格納され、かつＲＩＮがインクリメントされた後、サーバー２０は、入力制御が実施できるか否かを判断する(ボックス162,164)。
フロー制御は、受信用バッファ４４内のデータ量が、入力シーケンス及び受信用バッファ、ＲＬＯＷの値以下に降下するときに解放される。このことは、ボックス158,160で示すように、データがクライアント１８に送られた後に判断する。ステップ152におけるポート４０の検査でポート４０上に新しいデータが見つけられなかった場合、サーバー２０は、データが受信用バッファ４４に送られて利用できるかどうかを決定する。もし利用できない場合には、図７でボックス166に示すように、サーバー２０は、ポート４０にデータが送られるまで再待機する。
データが受信用バッファ４４で利用できるならば、その時、サーバー２０は、クライアント１８に送ることができるバイトの最大数を計算しなければならない(ボックス168)。この最大数は、シーケンスナンバーの減算公式に従って、単純な式(RWIN－ROUT)によって決定できる。この送られるべきバイトの最大数がゼロならば、サーバー２０は、判断ボックス170で示された最新となるべきＲＷＩＮのために待機しなければならない。この計算機がゼロよりも大きければ、サーバー２０は、データバイトをその時点でクライアント１８に送ることが適当であるか否かを判断しなければならない。
２つの判定は、データを伝送するトリガとして使用する。第１の判定(ボックス172)は、受信用バッファ４４内のバイト数(RIN－ROUT)と、クライアント１８によって送ることができる入力シーケンス及び受信用バッファのＲＭＡＸとを比較する。
受信用バッファ４４内のバイトがＲＭＡＸを越えると、バイトの最大数まで、クライアント１８は、送ることができる(ボックス176)。したがって、ＲＯＵＴは、インクリメントされる(ボックス178)。
受信用バッファ４４内のバイト数がＲＭＡＸを越えない場合、サーバー２０は、データが少なくともＲＴＩＭＥミリセコンド前にあったか否かを、さらに、時間データがクライアント１８に送られなかったかを判断する。その場合、判断ボックス174で示すように、データはクライアント１８に送られる。
データがクライアント１８に送られていない場合、データを送るための２つの条件172または174の１つが真となるまで受信用バッファ４４内にデータが残る。
ポート４０が開くと、ＲＭＡＸは１に、ＲＴＩＭＥはゼロに、ＲＬＯＷは１／４に、ＲＳＩＺ，ＲＨＩＧＨは３／４に、全てのシーケンスナンバーはゼロに初期化される。しかし、これらの変数は望ましい値に変更できる。
一般的に、サーバー２０は、そのポート４０にデータを送る時、送ることができるバイト

の計算最大数によって制限されなければ、各ポート４０に対して全ての利用可能なデータを送る。この規定は、サーバー２０とクライアント１８の両方が処理しなければならないパケット数を減少させ、そしてクライアント１８の効率を改善する。
クライアント１８側では、ホストコンピュータにおいて、本発明のオペレーションは、ホストコンピュータのオペレーテングシステムにより大いに変化させることができる。本発明のオペレーションを開示するために、クライアント１８側では、ＵＮＩＸホストシステム上の実行は、図８に示すように概略が示されている。本発明は、クライアントドライバ200を用いることでサーバー２０側のポート４０にアクセスでき、このクライアントドライバは、ローカルに接続された１組のシリアルポート204に対して１つのドライバを対応させることによってホストオペレーテングシステムのＡＰＩを実行する。これを行うために、クライアントドライバ200は、各リモートサーバーボート204に対して入力用または受信用バッファ206、出力用または伝送用バッファ208、及びリモートサーバー２０の状態に関する情報並びに状態メモリ210に格納されるホストコンピュータへのインタフェースに対してサポートする。
また、このドライバ200は、汎用ネットワークに接続される標準ネットワークインタフェース212に対しても接続できる。クライアントドライバ200からサーバー２０への全ての通信は、ネットワーク10上で伝送される。ネットワークインタフェース212は、通信プロトコルの下でドライバ200が接続される多数のポート４０に関わらず、各サーバー２０のためのネットワーク上の１つポートだけに接続が可能である。
このシステムが放逐されたとき、このオペレーテングシステムの立ち上げ処理はドライバ200を始動させる。そして、各サーバー２０のために、STREAMS接続部218をサーバー２０に接続するユーザモードのデーモン214を始動させる。このデーモン214は、STREAMS接続部218をドライバ200と連動する制御装置202に接続する。
次に、デーモン214は、STREAMSインタフェースを用いて、ドライバ200がネットワーク接続上のデータをデーモン214からの相互作用でサーバー２０のＷ／Ｏに直接に送受信できるように、制御装置202より下位のオペレーテングシステムリンクのストリーム218を要求する。そうすることにより、ドライバ200は、制御装置のSTREAMSの待機サービスルーチンにおいて、サーバー２０からのＴＣＰ／ＩＰのデータを送受信することができる。これは、従来技術で得られたタスク切換用オーバーヘッドオペレーションを削除する。
接続モードは、バイトフロー接続であり、ドライバ200がサーバー２０へ信頼性のあるシーケンスデータの送受信を行えるようにする。ドライバ200は、この接続の信頼性に完全に頼っており、失われたパケット及び同等物を回復させる機能は備えていない。このドライバ200は、サーバー２０から受信下データ内のエラーまでも保護するとすると、ドライバ200はデーモン214までのハングアップ信号を通過させ、その時、サーバー２８への接続を閉じたりまた再び開くようにする。このようなエラーは通常のオペレーション中に起こすことはできない。このエラーは、リモートサーバーソフトウエア、ネットワークソフトウエア、ドライバソフトウエア等の失敗、あるいは、ある種のコンピュータの失敗によってのみ起こる。ホストコンピュータ内の標準ＴＣＰ／ＩＰソフトウエアは、汎用コンピュータネットワーク上で通常起こるような失なわれたまたは歪曲したパケットニよるこの種のエラーが起こらないように有効に保証している。
一旦、ドライバ200がネットワーク接続にアクセスすると、ドライバは、サーバー２０にサーバー２０上の多数のシリアルポート４０及びサーバー２０の他のハードウエア及びソフトウエアの両特性に関する情報を戻すように要求する。
ユーザーモードタスク216は、ＴＴＹシリアルタスク204を接続しようとするとき、ホストオペレーテングシステム215は、ドライバ200にオープンコールする。このポート204は他のユーザモードタスク216によってすでに接続されていない場合、ドライバは、サーバー２０へのオープン要求を送り、ドライバ200のＴＴＹポート204に連動するサーバー２０上のポート４０への独占的アクセスを得る。ポート４０が利用可能である場合、サーバー２０は、クライアント１８がポート４０を閉じるまで、あるいはクライアント１８の接続が失われるまで、独占的なアクセスを許すように応答する。

ポート４０が利用できない場合、ユーザモードタスク216によって作られたオリジナルのオープン要求のタイプに従ってアクションが取られる。ユーザモードタスク216は、要求が直ちに行うことができない場合にオープン失敗であるという要求を行うことができ、あるいはポート４０が利用可能となるまで、要求を待機させるという要求を行うことができる。前者の場合、ドライバ200は、サーバー２０に迅速なオープンを要求し、このサーバー２０は、ポート４０が混んでいる場合にその要求を拒絶し、その時、ドライバ200は、ユーザモードタスク216にエラーを戻す。後者の場合、ドライバ200は、サーバー２０にオープン待機中であることを知らせ、ポート４０が最終的に利用可能となるまでクライアントが待機の順番を待つようにするかを尋ねる。その時、サーバー２０は要求が順番待ちであるという指示を伝え、ドライバー200は、ポート４０が利用可能であることをサーバー２０がドライバ200に知らせるまで、ユーザモードタスク216を休止状態に置く。

ドライバ200は、サーバーポート４０がうまく開かれると、サーバー２０へ照会パケットを送り、ボーレート発生器の周波数値及びポート４０マーク及びスペースパリティーをサポートできるかどうかを含む、ポート４０のハードウエア及びソフトウエアの特性を学習する。

また、ドライバ200が、サーバー２０にポート204のための受信用バッファ206のサイズを送ることから、サーバー２０は、更なる規定なしでクライアント１８により送られるデータがどれ位かを知ることになる。

サーバー２０は、照会に応じて、またポート４０の伝送用及び受信用バッファ４２，４４のサイズを含む、ポート４０の特性を送る。

ドライバ200がサーバー２０からの返答を受け取った時、ドライバ200は、ポート４０に送受信用データを送る準備をし、かつホストオペレーテングシステムのオープン／クロース／ｉｏｃｔｌ（入力／出力制御）インタフェースによって所望の要求を行う時、ユーザモードタスク216のために制御及び状態の照会をする。

その後、サーバー２０のポート４０上に受信した全てのデータは、自動的にクライアント１８に送られ、そこで、データは受信用バッファ206に格納される。サーバー２０は、受信用バッファ206に適合できるだけの入力ポートデータを最初に送るように定められており、それ以上のバッファのオーバーランの可能性はない。

ユーザモードタスク216がデータの読取りを要求すると、ホストオペレーテングシステムは、ドライバ200の読取ルーチンを呼び出す。読取り要求のパラメータに従って十分なデータがサーバー２０から受信された時、ドライバ200はホストオペレーテングシステム215にそのデータを戻し、順次データをユーザモードタスク216へ通過させる。十分なデータが利用できない場合、ドライバ200は、データが到着するまで、あるいはホストオペレーテングシステム215のＡＰＩに従って中断されるまで、ユーザモードタスク216を休止状態に置く。

ドライバ200が受信用バッファ206からデータを転送したとき、ドライバ200は、このデータがシーケンスナンバーＲＷＩＮを転送させたバイト数だけインクリメントすることによって転送されたことをサーバー２０に知らせる。サーバー２０は、その時、そのデータがシーケンスナンバーＲＷＩＮに達した後に付加的なデータを送ることができる。

ユーザモードタスク216がデータを書き込むために要求されるとき、ホストオペレーテングシステム215は、ドライバ200の書込みルーチンを呼び出す。ドライバ200がそれをコピーする場合、伝送用バッファ208に適合するだけの量のユーザデータをコピーする。全てのデータが伝送用バッファ208に置かれたならば、サーバー２０は要求を実行し、ユーザモードタスク216を続行することができる。もし、伝送用バッファ208に適合する全てのデータがない場合、ドライバ200はデータがサーバー２０に送られるまでユーザモードタスク216を休止状態にする。そして、伝送用バッファ208内の空きスペースが十分となって、残りのデータがバッファ208内に格納できるようになる。

ユーザモードタスク216が、ｉｏｃｔｌ要求として知られている制限または照会要求をするとき、ホストオペレーテングシステム215は、ドライバ200内のｉｏｃｔｌルーチンを呼び出す。この要求が、ポート204のハードウエア及びソフトウエアの両特性を変える場合

、ドライバ200は、状態メモリ210における変化を示す。そして、サーバー２０に必要な変更をさせるために命令を送る。この要求が、ドライバ200がサーバー２０に問い合わせることなく応答できる照会要求である場合、ドライバ200はこれを行う。この要求が、サーバー２０内のデータを必要とする場合、ドライバ200は、この情報のためにサーバー２０に照会要求を送り、そして、ユーザモードタスク216の要求を満足させるために戻された情報を用いる。

ユーザモードタスク216によって作ることができるこの種の要求は、非常に多様であり、その動作も広く変化する。ある場合、この要求は、直ちに満足できるが、また、ある場合には、この要求が完了するまでユーザモードタスク216が休止状態に置かれる。

ユーザモードタスク216がＴＴＹポート204に最後のクローズをするとき、ホストオペレーテングシステムは、ドライバ200のクローズルーチンを呼び出す。これにより、ドライバ200は、ユーザモードタスク216を休止状態にさせ、伝送用バッファ208内の全てのデータが伝送されるまで待機させ、また、サーバー２０に継続中の要求が完了するまで待機させる。

ドライバ200は、その時、サーバー２０にクローズ要求を送る。このクローズ要求が受け入れられると、サーバー２０は、ポート４０を割り当てそしてのクライアント１８にそのポートの再割り当てを可能にする。ドライバ200は、クローズを完了し、そしてユーザモードタスク216を続行する。

ドライバ200のオペレーションの効率を最適化し、かつホストオペレーションシステム215上の負荷を最小化するために、ドライバ200はサーバーとの通信を約５０回／秒の周期間隔で行う。この各間隔で、ドライバ200は、そのサーバー２０に接続されている各活動ポート204を調べ、このようなポート204，４０のすべてに対して要求及びデータを共通のメッセージとして与え、さらにこのメッセージを１つのオペレーションとしてサーバー２０に送る。この動作は、サーバー２０に送らなければならないＴＰＣ／ＩＰメッセージの総数、あるい他の汎用ネットワークメッセージを最小化し、またシステムのオペレーションを高める。

本発明は、ネットワーク１０間のシリアルポートハードウエアをシミュレートさせるために用いられるプロトコルを提供することである。このように使用するとき、ボードあるいは他のハードウエアは、ホストコンピュータに設置され、ローカルシリアルポートボードに類似するホストコンピュータへのシリアルポートインタフェースを備えている。

本実施例では、ボードは、汎用ネットワークのプロトコルスタックを含み、さらに、シリアルポートインタフェースを備え、またクライアントドライバ200と接続した上記機能を備えるソフトウエアを付加する。このボードを用いて、複数のドライバは、シリアルポートインタフェースだけが知られかつネットワーク１０の存在が知られていない、ホストオペレーテングシステム上で使用することができる。

このようなハードウエアのインプリメンテーション（実装）は、いくつかの利点がある。第１に、ハードウエア用またはソフトウエア用のインタフェースをエミュレートさせて使用することができ、このため各ドライバは、ホストオペレーテングシステム上で利用可能または容易に採用することができる。第２に、このインターフェースボード（またはカード）は、コンピュータに設置することができ、そこでたやすく利用できないハードウエアまたはソフトウエアをネットワークさせ、これにより、ホストコンピュータにネットワークサービスを提供できる。第３に、このインターフェースカードは、カード上にオーバーヘッドされる多くのシリアルポートをオフロードする付加的なプロセッサ及びハードウエアを含ませることができる。最後に、このようなカードは、本来、多機能であり、例えば、ローカルコンピュータのためにリザーブされ、あるいはアクセス可能なネットワークとなることができる小さな物理的装置を含んでいる。

サーバー２０とクライアント１８の間で交換可能なデータ及びコマンドを含む全ての情報は、パケットと呼ばれる小さな単位でネットワークプロトコルによって与えられるバイトフローよって送られる。これらのパケットのフォーマットは、本発明の通信プロトコルを作り上げる。この通信プロトコルパケットは、フォーマットにおいてＣＲＴターミナルの

エスケープシーケンスと類似しており、その中で、シーケンスのわずかな第１バイト数が次に続くデータのフォーマット及び長さを決定する。
サーバー２０からクライアント１８への通信用パケットの形成は、サーバー２０のＣＰＵ３２において行われる。クライアント１８からサーバー２０に送られるパケットは、クライアント１８側で動作する特定ドライバによって作り出される。このパケットは、それ自体、パレットに含まれる情報の種類による可変フォーマット及び可変長さを有している。各パケットのフォーマットは、パケットを順次アクセスする処理によって決定される。より前のバイトの値が次に続くバイトのフォーマット及び内容をすべて決定する。
表９は、サーバー２０とクライアント１８との間で伝送される各パケットの異なる形式を示す。この表は、１つのパケットを主たる特殊形に分解された第１バイトがどのようになるかを示している。禁止(illegal)を示す各値は、本発明の好ましい実施例としては用いられない。
データパケットは、クライアント１８とサーバー２０との間のインバンドーデータのすべてを伝送するのに使用される。サーバー２０によって受信されると、このデータパケットのデータは、上記のように適当なポート４０を介して伝送される。
ウインドウ・シーケンスパケットは、受け手に送り手のシーケンスナンバーの状態について知らせる。この情報の伝送は、上記で概略説明したように、本発明のデータフロー制御の働きに関して重要である。ウインドウ・シーケンスパケットは、クライアント１８にＴＯＵＴを送るためにサーバー２０によって用いられ、そして、サーバー２０にＲＷＩＮを送るためにクライアント１８によって用いられる。
コマンドパケットも、１バイトにおけるコマンド型フィールドの値により、異なるフォーマットと異なる長さを有している。各コマンドは、サーバー２０の状態変数３８の各値について設定しかつ照会するために伝送される。さらに、コマンドパケットは、ポートを開くために利用され、クライアント１８がサーバー２０と同期が取れるように、またサーバーの送受信用バッファ４２，４４のサイズを確かめるために、特定ポート４０の容量を確かめるために、サーバー２０のバッファ４２，４４をフラッシュするために、ブレーク又は中間のキャラクタを伝送するために、また、入力または出力を休止あるいは再始動させるために利用される。このコマンドパケットは、表１０において設定されている。
２つの最も重要なコマンドパケットは、開いているポート４０と接続する場合のクライアント１８とサーバー２０との間の通信を包含している。ポート４０を開くための要求コマンドは、サーバー２０上のポート４０に独占的にアクセスすることを望むクライアント１８によって送られる。サーバー２０が、このオープン要求コマンドを受け取ると、サーバー２０は、クライアント１８のためにポート４０を開くように試み、そして、クライアント１８にオープン要求の結果を知らせるオープン応答パケットによって返答する。
オープン要求コマンドパケット及びオープン応答コマンドパケットのフォーマットは、表１１，１２に設定されている。
ポート４０を開くためのコマンドパケットは、要求するクライアント１８によって望むオープンコマンドの形式に従って異なる形式を取ることができる。オープンコマンドの可能な形式は、即時オープン、持続オープン、入来オープンがある。即時オープンのための要求は、ポート４０が有効であって直ちに利用可能である場合にのみ、受け入れられ、そして、ポート４０が要求するクライアント１８に割り与えられる。ポート４０か直ちに利用可能でない場合、要求は拒絶される。持続オープン要求がなされると、この要求は、ポート４０が利用可能であるならは、受け入れられ、そしてポート４０はクライアント１８に割り与えられる。もし、ポート４０が混んでいる場合、サーバー２０は、クライアント１８へのオープン応答パケットに戻り、クライアントにポート４０が混んでいることを示し、そして、クライアント１８をそのポート４０に対する待機リスト上に置く。ポート４０が順次利用可能になると、ポート４０は開かれ、そして、クライアント１８は、第２のオープン応答が受け入れ可能であることを知ることができる。
オープン要求の第３形式は、入来オープンとして要求である。この要求は、ポート４０が利用可能であるならば、直ちに受け入れられ、そして、キャリアがポート４０上に現れる

。ポート４０が混んでいる場合、またはキャリアが現れない場合、サーバー２０は、対応するオープン応答コードに戻り、そして、クライアント１８をそのポート４０のための待機リスト上に置く。ポート４０が続いてキャリアの出現とともに利用可能になると、ポート４０は開かれ、クライアント１８は、第２のオープン応答が受け入れ可能であることを知ることができる。

クライアント１８は、特定ポート４０に対するオープン要求のキャンセル待機形式を送ることもできる。このコマンドをクライアント１８から受け取ったとき、サーバー２０は、クライアント１８がそのポート４０の待機リスト上にあるかどうかをチェックできる。クライアント１８がそのリスト上にあることがわかると、クライアント１８は、取り除かれ、サーバー２０はオープン応答が受け入れ可能である指示に戻す。サーバー２０上のポート４０を追跡用ポートに組織化することが可能である。

このようにすると、サーバー２０は、追跡用ポートとして指定した１つ又はそれ以上の論理ポートを用意し、その結果、追跡用ポート群の１つをオープンにするための試行によって、追跡用ポート群の１つとして形成される物理的ポート４０を開く。こうして、クライアント１８が追跡用ポート群の一部であるポート４０を要求する場合、開かれた実際のポート４０は、要求されたポート４０とは異なることになる。クライアント１８は、オープンが受け入れられたのち、オープン応答のＰＮＵＭを調べなければならない。そして、そのポートの番号がポート４０に当たるか順次参照する。

同期要求コマンドは、その受け手が同期応答コマンドに応答するようにさせる。これは、その他のことに関しては何の効果もない。このコマンドの第１の目的は、クライアント１８がサーバーと同期できるように保証することである。例えば、クライアント１８は、呼び出しプログラムに戻る前に、要求したボーレート変更を一致させることを望む。クライアント１８は、第１にボーレート変更をサーバー２０に伝送することによりこれを達成する。そして、それに続いて同期要求を行う。サーバー２０は順次コマンドを処理するので、クライアント１８が同期応答のマッチングを受けると、クライアント１８はボーレート変更が完了したことがわかる。

シーケンス要求と応答パケットが与えられて、クライアント１８はサーバー２０へ入出力するデータの流れを正確に追跡することができる。サーバー２０は、クライアント１８にＲＩＮ及びＴＯＵＴを与えることによりこのパケットに応答する。

クライアント１８は、状態要求パケットを送り、サーバー２０におけるＥＳＴＡＴとＭＳＴＡＲＴの現在の常識を識別する。サーバー２０は、状態応答パケットにおける要求データを戻し、そして他の動作は取らない。このコマンドは、クライアント１８がサーバー２０の状態を学習する２つの方法のうちの１つである。クライアント１８は、選択された状態変化をサーバーによって自動的に報告させるために、選択イベント条件パケットを用いることができる。

クライアント１８は、ラインエラーカウンタを選ぶために、このラインエラーカウンタの周期的報告を形成し、そして、カウンタをクリアにするためにラインエラー要求パケットを送る。サーバー２０は、エラーカウンタの選別に応答してラインエラー応答を送り、そして、これは、変数ＬＴＩＭＥとして形成される周期的間隔で送られる。ラインエラー要求パケットは、すべてサーバーの変数ＬＴＩＭＥとして設定される。ＬＴＩＭＥの値が０では、自動的に報告を行わない。ＬＴＩＭＥの値がＦＦＦＦでは、ＬＴＩＭＥは変化しない。

クライアント１８がＬＴＩＭＥを０でない値に設定した後、サーバー２０は、ＬＴＩＭＥミリセコンド毎に（ほぼ２０ミリセコンドの正確さで）各エラーカウンタの状態を調べる。そして、その状態が最後の報告まで変わらないならば、自動ラインエラー応答パケットを送る。

バッファ要求パケットは、クライアント１８によって送られ、サーバーの伝送用バッファ４２及び受信用バッファ４４のサイズを学習する。サーバー２０は、バッファ応答パケットに応答する。ポートがうまく開いた後、そしてサーバー２０にデータを送る前、クライアント１８は、サーバーのバッファ４２，４４のサイズを確認する必要がある。伝送用バ

ッファ４２のサイズは、クライアント１８が伝送できるデータがどの位かを知るために必要である。受信用バッファ４４のサイズは、サーバー２０がフロー制御の流れ容量の高低を設定するのに必要である。
ポートの能力要求パケットは、ポート４０のハードウエア及びソフトウエアの各能力を学習するためにクライアント１８によって送られる。他のポートのコマンドはそのレベルが違っており、このコマンドが発生しても、ポート４０は開くことはできない。
設定ボーレートコマンドは、クライアント１８がポートのボーレート、すなわち、ポート４０のＣＦＬＡＧＳ，ＩＦＬＡＧＳ，ＯＦＬＡＧＳ，ＸＦＬＡＧＳを設定するのに使用する。
クライアント１８は、選択イベント条件コマンドを送り、どのＥＳＴＡＴ，ＭＳＴＡＴの条件がサーバー２０に１つのイベントパケットを送らせることができるかを特定する。このクライアント１８は、また、状態要求パケットを用いてＥＳＴＡＲＴ及びＭＳＴＡＲＴを調査することができる。
設定ウインドウトリガーコマンドにより、クライアント１８はサーバーの状態変数３８の１つであるＴＲＥＱを設定することができる。ＴＲＥＱは、ウインドウシーケンスパケットが送られる時点で、クライアント１８が要求したレベルである。ＴＲＥＱが出力バッファにおける現在のデータ範囲から外れている場合、サーバー２０は直ちに１つのウインドウシーケンスパケットに応答する。
設定モデム出力及びフロー制御パケットは、クライアント１８がモデム出力を設定し、そして、モデム信号ハードウエア用のフロー制御を選択するするために送られる。ＲＴＳとＤＴＲのモデムフロー制御が選択されたとき、ＭＯＵＴにおけるＲＴＳとＤＴＲの値は無視される。
クライアント１８は、設定受信用の（高／低）ウオーター指示パケットを送り、受信用の（高／低）フオーター指示のフロー制御を設定する。このパケットはサーバー２０によって処理されると、直ちに影響を与える。入力バッファにおけるキャラクタの数がＲＨＩＧＨを越えると、フロー制御は、直ちに実施が強制される。入力バッファがＲＬＯＷより小さいと、フロー制御は、直ちに緩やかに実施される。
クライアント１８は、設定フロー制御キャラクタパケットを送り、サーバー２０にソフトウエアフロー制御キャラクタを設定する。同様に、クライアント１８は、設定ＲＭＡＸとＲＴＩＭＥコマンドを送り、各パラメータＲＭＡＸ，ＲＴＩＭＥを設定し、さらに設定ＴＭＡＸとＲＴＩＭＥコマンドを、ＴＭＡＸとＴＴＩＭＥに設定する。
クライアント１８は、インバンドーデータの頭文字を送るために、伝送用キャラクタ即時パケットを送る。伝送用即時データにおいて、ソフトウエアフロー制御は、無視され、ハードウエアフロー制御は、ＭＦＬＯＷの代わりに変数ＭＣＴＲＬに用いられる。クライアント１８は、ハードウエアブレーク信号を送るために伝送用ブレーク即時パケットを利用する。このブレーク信号は、また、すべてのインバンドーデータの前に送られ、そして、ソフトウエアまたはハードウエアのフロー制御に関わらず送られる。
クライアント１８は、サーバーの入力及び出力バッファの両方またはいずれか一方をフラッシュするためにフラッシュバッファパケットを送る。休止（入力／出力）パケットは、サーバーにおけるソフトウエアフロー制御条件の幾組かを休止する。受信用バッファ内のキャラクタの数がＲＬＯＷより少ない場合、入力を休止するための試みは無駄になる。最後に、反休止（入力／出力）パケットは、サーバーにおけるソフトウエアフロー制御条件の幾組かを休止させない。受信用バッファ内のキャラクタの数がＲＨＩＧＨより多い場合、入力を休止させないための試みは無駄になる。
イベントパケットがサーバー２０によって送られて、ＥＳＴＡＴとＭＳＴＡＴの状況の変化をクライアント１８に知らせる。サーバー２０は、（ＥＴＲＡＮ及びＥＩＮＴ)あるいは（ＭＴＲＡＮ及びＭＩＮＴ）が非ゼロであるときはいつでも、各ポート４０にイベントパケットを送る。
この要求を確実にすることがイベントパケットに、別な方法で報告され、失われるようなトランジッションを引き起こさない。すなわち、サーバー２０は、最初にＥＬＡＳＴを設

定し、続けてＭＬＡＳＴを設定する。それから、ＥＬＡＳＴ及びＭＬＡＳＴをクライアント１８に報告する。

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

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

モジュール選択は、ポート４０の特定グループを選択するために使用される。通信の帯域幅を減少するために、ポート４０は１６のポート４０（ポート０－１５）のみを含む各モジュールと共に、論理モジュールにグループ化される。論理モジュール群とサーバ（６）上のポート４０の物理的配置の間には、関係がない。ポート４０をモジュールに一括することによって、すべてのコマンド及び応答リファレンスにおけるアドレッシングは、ポート（０－１５）が、このモジュール内で望まれることに関連していることのみが重要である。モジュール０は不履行の際、選択される。

モジュール０－７は、１バイトモジュール選択パケットと共に選択可能である。モジュール８－２５５は、２バイトのパケットを要求する。モジュールとポートペアと物理的ポートナンバーとの間で、転換することが可能である。モジュールｎが選択されるとき、すべてのそれに続くパケットのポートＫは、物理的ポート１６＊ｎ＋Ｋに関連する。

ＩＤ要求パケットは、リモート側からの機器構成の情報あるいは一致を得るため、クライアント１８あるいはサーバー２０のどちらかに送られる。リモート側はＩＤ応答パケットに応じる。

デバッキングパケット型は、クライアント１８とサーバー２０との間で、デバッキングデータを伝送するような、デバッグをする目的でサーバー２０のポート４０にアクセスすることをサポートする。これらのパケットはクライアント１８がサーバープログラムメモリ３６にアクセスすることも可能にし、プログラムメモリ３６に記憶されているようなサーバ２０のプログラムを代えること、あるいはプログラムメモリ３６の内容をレビューすることを許容する。

リセットパケット型は、接続が停止される前に、クライアント１８あるいはサーバー２０によって、リモート側にプロトコルエラーを報告するために送られる。パケットはＡＳＣＩＩフォーマットのリセットのための理由の説明を含む。リセットパケットは、対話で送られた直前のパケットでなければならない。

この議論は、ＵＮＩＸ以外の他のオペレーテングシステムについて特になされていないが、ノーベル、ウインドウズ、ウインドウズＮＴ、ＯＳ／２及びＤＯＳのアプリケーションプログラミングインタフェースを満足させるために要求されるＸＦＬＡＧＳ、エラーカウンタやイベントフラグで実証される特徴は、当業者に明らかである。結果として、これらオペレーテングシステムのいずれでも、本発明のインプリメンテーションは、当業者に明らかである。

本発明はそのすべての詳細に制限されることはなく、本発明の精神あるいは範囲内でなされる変形や修正されたものにも制限されない。

【図面の簡単な説明】

図１は、種々のホストコンピュータと本発明のサーバーを、汎用コンピュータにネットワーク接続することを示す。

図２は、汎用ネットワーク上のクライアントのコンピュータに、サーバーの主要構成要素を含む、本発明のサーバーを取り付けたことを示す。

図３は、シーケンス空間内の位置に向けて指定する単一のシーケンスナンバーを有するラップアラウンド・シーケンス空間を示す。

図４は、シーケンス空間内の位置に向けて指定する２つのシーケンスナンバーを有するラップアラウンド・シーケンス空間を示す。

図５は、サーバーが、クライアントからサーバーを通じてポートへ伝送されるデータの流

れを、制御する過程を示す。
図６は、クライアントからサーバーを通じてポートへ伝送されるデータの流れを、クライアントが制御する過程を示す。
図７は、サーバーが、ポートからサーバーを通じてクライアントへ伝送されるデータの流れを、制御する過程を示す。
図８は、汎用ネットワーク上のクライアントのコンピュータに、クライアントのドライバーの主要構成要素を含む、本発明のサーバーを取り付けたことを示す。

表１：サーバーの状態変数

サーバーの状態変数のタイプ	頻度
顧客接続	各サーバーに付き
メッセージ　ビルド　インターバル	各サーバーに付き
ポート　オープン　ステート	各ポートに付き
ポート　ボー　レート	各ポートに付き
フラグ　処理	各ポートに付き
フロー　制御　キャラクタ	各ポートに付き
モデム　制御	各ポートに付き
ライン　エラー　カウンタ	各ポートに付き
センド　ブレーク　及び　センド　イミーディエイト	各ポートに付き
イベント　リポーティング	各ポートに付き
入力シーケンス及び受信用バッファのパラメータ	各ポートに付き
出力シーケンス及び伝送用バッファのパラメータ	各ポートに付き

表２：ＸＦＬＡＧＳのセット

マスク	名前	説明
０００１	ＸＰＡＲ	パリティをマークする、あるいは空間にすることを可能にする。
０００２	ＸＭＯＤＥＭ	インバンドモデム信号を可能にする。
０００４	ＸＣＡＳＥ	特殊文字を出力で多重化文字シーケンスに変換
００４０	ＸＴＯＳＳ	ＩＸＡＮＹキャラクタを文字にトス
２０００	ＸＩＸＯＮ	フロー制御キャラクタの出力ソフトウエアの第２のセットを可能にする。

表３：モデム変数

名前	説明
ＭＯＵＴ	顧客がモデム出力値を特定する
ＭＦＬＯＷ	インバンドデータのモデムフロー制御
ＭＣＴＲＬ	アウトオブバンドデータのモデムフロー
ＭＳＴＡＴ	最新のモデム状態
ＭＬＡＳＴ	顧客に送られた直近のモデム状態
ＭＴＲＡＮ	顧客にはその変更が送られていない、変更されたモデム信号のセット
ＭＩＮＴ	顧客に報告されるべきモデム信号の変化

表４：モデム変数でのモデム信号

マスク	名前	説明
０１	ＤＴＲ	データ　ターミナル　レデイ
０２	ＲＴＳ	送信要求をする
１０	ＣＴＳ	送信をクリアする
２０	ＤＳＲ	データ　セット　レデイ
４０	ＲＩ	リング　インジケータ
８０	ＤＣＤ	データ　キャリア　ディテクト

表５：イベント　リポーテイング　変数

名前	説明
ＥＳＴＡＴ	イベント状況の最新の状態
ＥＬＡＳＴ	顧客に送られた直近のイベント状況
ＥＴＲＡＮ	顧客には報告されていないトランジッションがあるイベント状況のセット
ＥＩＮＴ	発生するイベントを起こすイベント状態のセット

表６：記録されたイベント

マスク	名前	説明
０００１	ＯＰＵ	出力は顧客によって無条件で止められた
０００２	ＯＰＳ	出力は正規のソフトウエアフローで止められた
０００４	ＯＰＸ	出力は特別のソフトウエアフローキャラクタによって、止められた
０００８	ＯＰＨ	出力はハードウエアフローによって、止められた
００１０	ＩＰＵ	入力は顧客によって無条件で止められた
００２０	ＩＰＳ	入力は（高／低）ウオーターマークによって停止した
００４０	ＴＸＢ	伝送ブレーク続行中
００８０	ＴＸＩ	伝送イミーディエイト続行中
０１００	ＴＸＦ	伝送フロー制御キャラクタ続行中
０２００	ＲＸＢ	ブレーク受け入れ

表７：入力シーケンスと受信用バッファの変数

名前	説明
ＲＩＮ	ＵＡＲＴから読み込まれ、ローカル受信用バッファ内におかれるデータの次のバイトのシーケンスナンバー
ＲＯＵＴ	受信用バッファから顧客に伝送される、次のバイトのシーケンスナンバー
ＲＷＩＮ	顧客が受け取りたい最大シーケンスナンバー
ＲＭＡＸ	受信用最大トリガー
ＲＴＩＭＥ	受信用トリガー時間
ＲＳＩＺＥ	ローカル受信用バッファのサイズ
ＲＬＯＷ	ソフトウエア　フロー　ロー　ウオーター　マーク
ＲＨＩＧＨ	ソフトウエア　フロー　ハイ　ウオーター　マーク

表８：出力シーケンスと伝送用バッファのパラメータ

名前	説明
ＴＳＩＺＥ	伝送用バッファのバイトのサイズ
ＴＩＮ	伝送用バッファにおかれる次のバイトのシーケンス
ＴＯＵＴ	伝送用バッファからポートへ送られる次のバイトのシーケンス
ＴＰＯＳ	顧客へ報告された最近のＴＯＵＴの値
ＴＭＡＸ	ＴＯＵＴの報告最大値
ＴＴＩＭＥ	ＴＯＵＴの報告時間（ミリセカンド）
ＴＲＥＱ	顧客が要求したシーケンスナンバー

表９：パケットタイプ

ニブル　0	ニブル　1	説明
0	ポート０－１５	データパケット。１バイトのデータが特定ポートに続く。
1	ポート０－１５	データパケット。２バイトのデータが特定ポートに続く。
2	ポート０－１５	データパケット。３バイトのデータが特定ポートに続く。
3	ポート０－１５	データパケット。４バイトのデータが特定ポートに続く。
4	ポート０－１５	データパケット。５バイトのデータが特定ポートに続く。
5	ポート０－１５	データパケット。６バイトのデータが特定ポートに続く。
6	ポート０－１５	データパケット。７バイトのデータが特定ポートに続く。
7	ポート０－１５	データパケット。８バイトのデータが特定ポートに続く。
8	ポート０－１５	データパケット。バイト１のパケットは特定ポートに続く１－２５５バイトを、特定する。
9	ポート０－１５	データパケット。バイト１－２のパケットは、特定ポートに続く１－６５５３５バイトを特定する。
１０	ポート０－１５	ウインドウシーケンスパケット。バイト１－２のデータは、ウインドウシーケンスナンバーを特定する。
１１	ポート０－１５	コマンドパケット。バイト１は、長さを決定し、それに続くデータのフォーマットをきめるコマンドナンバーを特定する
１２	ポート０－１５	イベントパケット。３バイトのステータスが続く。
１３	*	禁止
１４	*	禁止
１５	モジュール0-7	モジュール選択パケット。ニブル１は、選択されたモジュールナンバーを含む。
１５	８	モジュール選択パケット。バイト１はレンジ０－２５５でモジュール選択コードを特定する。
１５	９－１０	禁止
１５	１１	ＩＤ応答パケット。バイト１は続くデータのサイズを決定する。
１５	１３	デバッギングとパケット
１５	１４	デバッグパケット。まったく無視される多くの続くバイトを特定する。ストレステストを使用。
１５	１５	リセットパケット。リセットの理由を記述する未終了のＡＳＣＩＩストリングに続く。

表１０：コマンド型の割り当て及び長さ

コマンド型（バイト１の値）	コマンド長さ（バイト）	コマンドの名前
１０	３	オープン要求
１１	６	オープン応答
１２	３	同期要求
１３	３	同期要求
１４	２	シーケンス要求
１５	６	シーケンス応答
１６	２	ステータス要求
１７	５	ステータス応答
１８	６	ラインエラー要求
１９	１４	ラインエラー応答
２０	２	バッファ要求
２１	６	バッファ応答
２２	２	ポート能力要求
２３	３２	ポート能力応答
４０	１２	ボーレート、ＣＦＬＡＧＳ、ＯＦＬＡＧＳ、ＸＦＬＡＧＳの設定
４２	５	イベント状況選択
４３	４	ウインドウトリガーの設定
４４	５	モデム出力及びフローの設定
４５	６	（高／低）ウオーターマークのフローの設定

４６	７	フローキャラクタの設定
４７	６	ＲＭＡＸ及びＲＴＩＭＥの設定
４８	６	ＴＭＡＸ及びＴＴＩＭＥの設定
６０	３	キャラクタの即時送出
６１	４	ブレークの即時送出
６２	３	フラッシュ　入力／出力
６３	３	休止　入力／出力
６４	３	反休止　入力／出力

表１１：オープン要求パケット

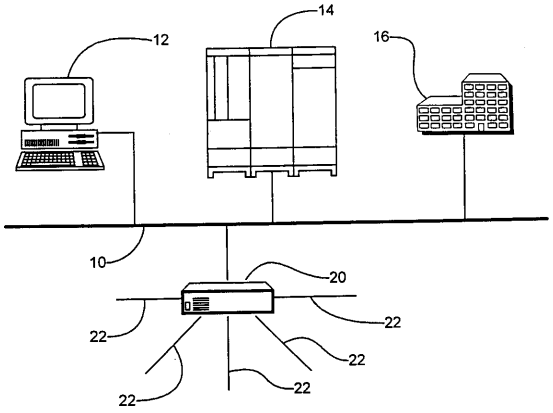
位置	名前	説明
ニブル　０	ＭＴＹＰＥ	メッセージタイプ１１：コマンド
ニブル　１	ＰＯＲＴ	ポートナンバー　０－１５
バイト１	ＣＴＹＰＥ	コマンドタイプ10：オープン要求
バイト２	ＲＥＱ	要求タイプ： ０＝即時オープン １＝持続オープン ２＝入来オープン ３＝即時オープンのキャンセル／クローズ ４＝持続あるいは入力オープンのキャンセル／クローズ

表１２：オープン要求パケット

位置	名前	説明
ニブル　0	ＭＴＹＰＥ	メッセージタイプ１１：コマンド
ニブル　1	ＰＯＲＴ	ポートナンバー０－１５
バイト１	ＣＴＹＰＥ	コマンドタイプ11：オープン応答
バイト２	ＲＥＱ	要求タイプ： ０＝即時オープン １＝持続オープン ２＝入来オープン ３＝即時オープンのキャンセル／クローズ ４＝持続あるいは入力オープンのキャンセル／クローズ
バイト３	ＲＥＳＰ	要求　コード： ０＝サクセス １＝ポート　混雑 ２＝ノー　キャリア ３＝リソース　アロケーション　プロブレム、再試行 ４＝要求はこの装置でサポートされなかった
バイト4-5	ＰＮＵＭ	実際のポートナンバー（１６＊モジュール＋ポート）がこの要求によって開いた。

【図１】

Fig. 1



【図２】

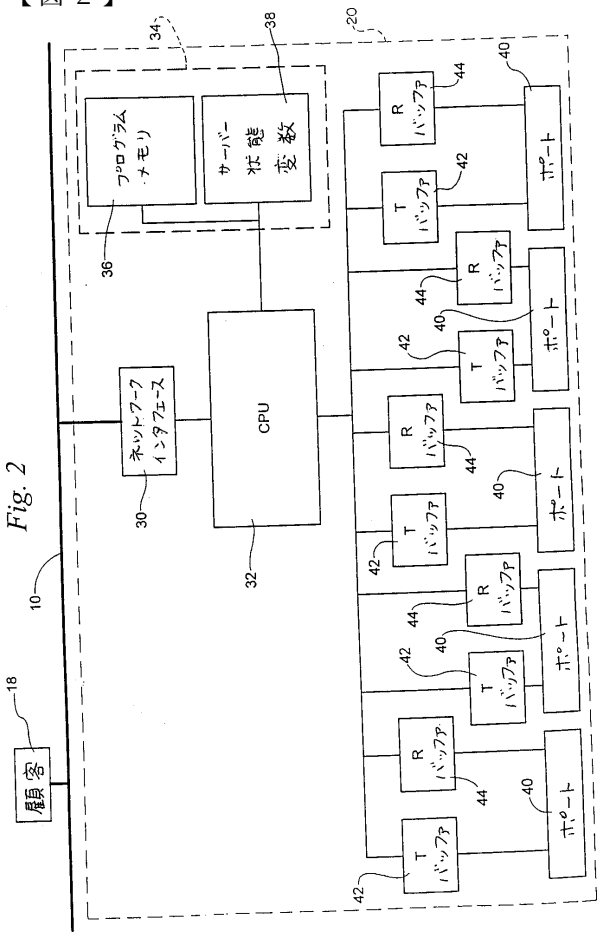


Fig. 2

【図３】

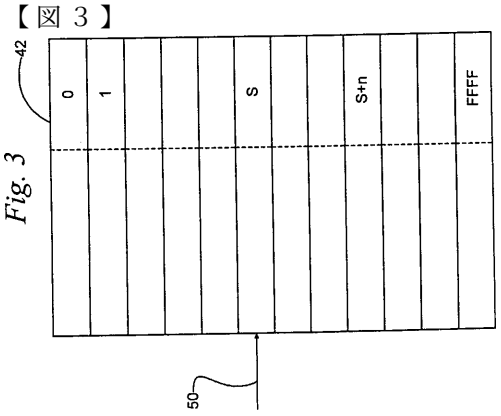


Fig. 3

【図４】

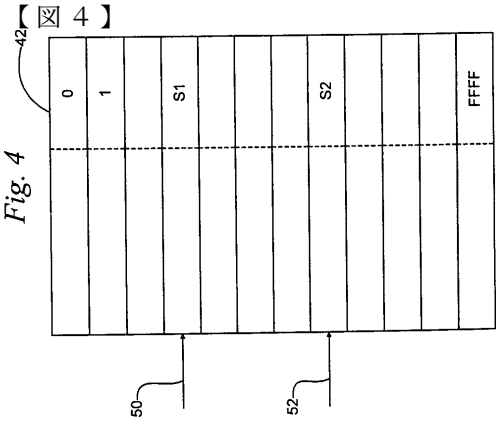
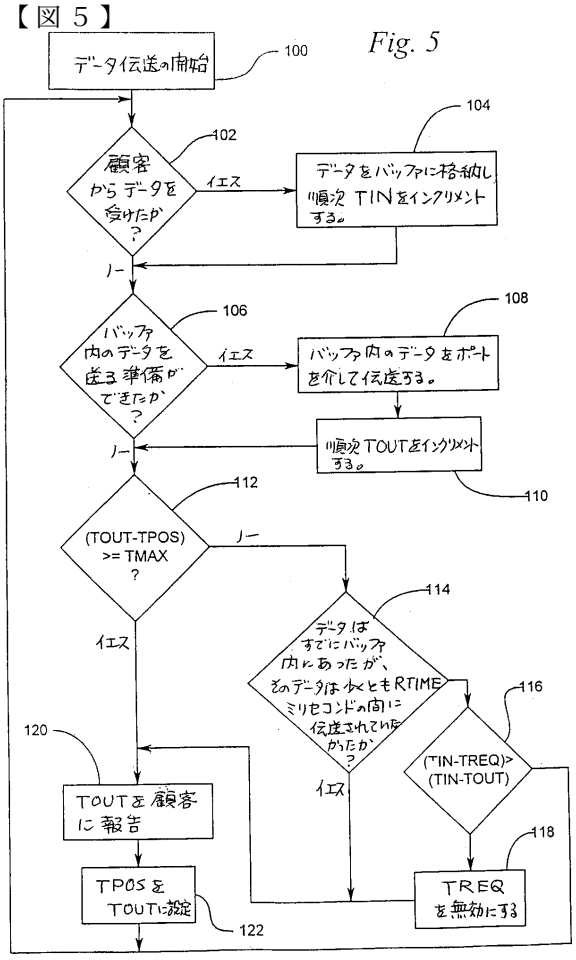


Fig. 4

【図５】

Fig. 5



【図６】

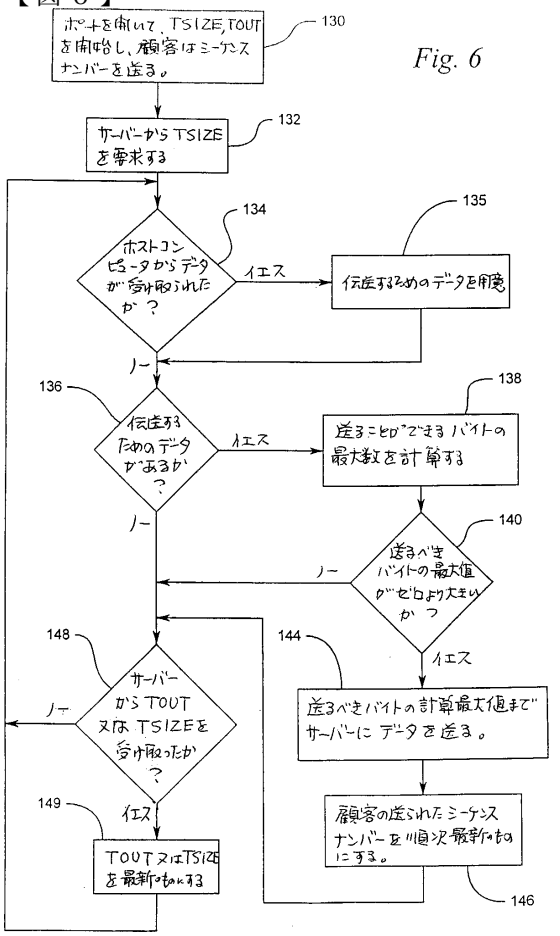


Fig. 6

【図７】

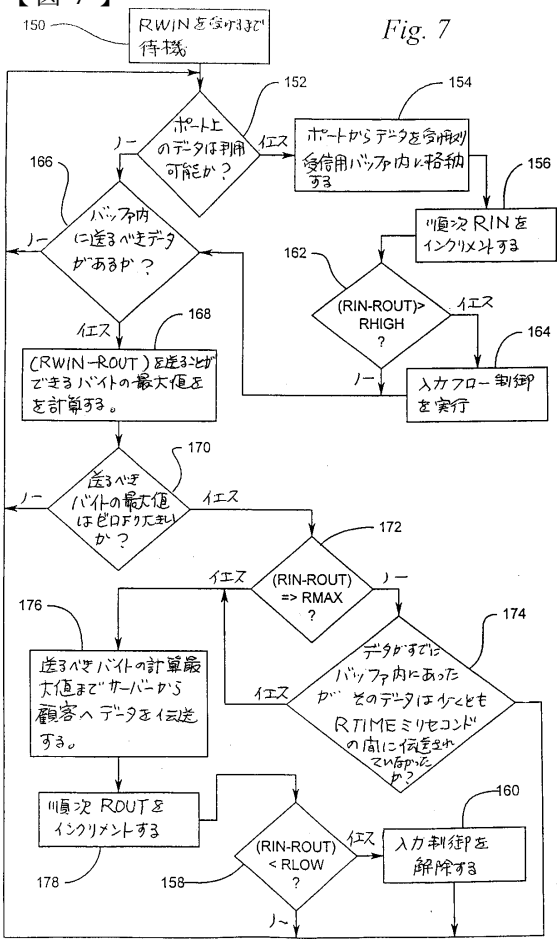
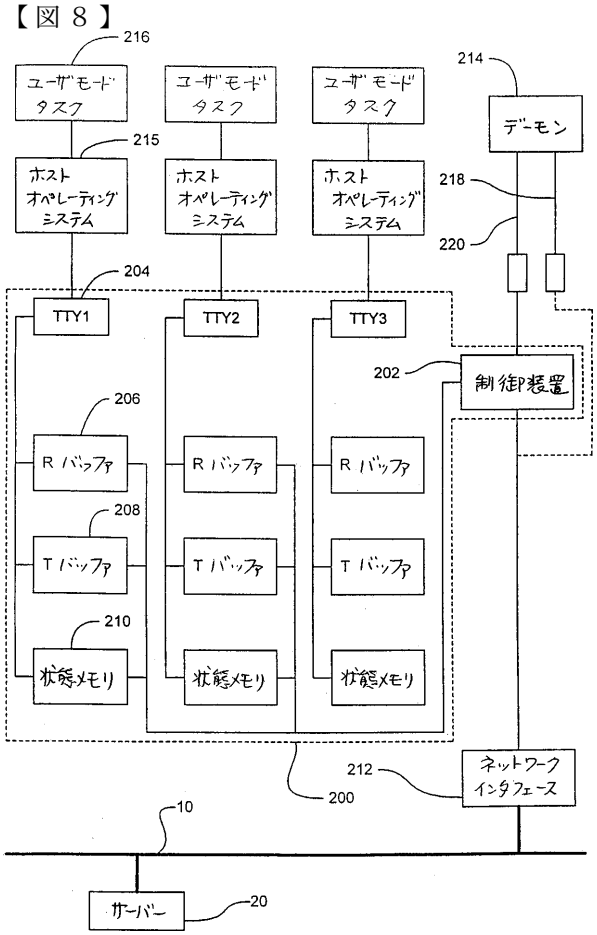


Fig. 7

【図８】



フロントページの続き

(72)発明者　オルソン，ジェン，エイチ.
アメリカ合衆国　ミネソタ　５５４１９，ミネアポリス　アルドリッチ　アベニュー　サウス　５１３１

審査官　石井　茂和

(56)参考文献　特開平０３－１５８９４７（ＪＰ，Ａ）
特開平０３－１０５６５０（ＪＰ，Ａ）
特開平０４－２９９７５８（ＪＰ，Ａ）
特開平０６－０２２００５（ＪＰ，Ａ）
特開平０５－２９８２２４（ＪＰ，Ａ）
特開平０５－１８９３３３（ＪＰ，Ａ）
特開平０３－２４７１４１（ＪＰ，Ａ）
特開昭５６－０６３６２６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 13/00 351
H05L 12/40
H04L 12/56
H04L 29/10
WPI(DIALOG)