(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4896328号
(P4896328)
(45)発行日　平成24年3月14日(2012.3.14)　(24)登録日　平成24年1月6日(2012.1.6)

(51)Int.Cl.　　F I
G06F 13/00 (2006.01)　　G06F 13/00　354A
G06F 15/00 (2006.01)　　G06F 15/00

請求項の数 19　(全 17 頁)

(21)出願番号　特願2001-525498 (P2001-525498)
(86)(22)出願日　平成12年9月22日 (2000.9.22)
(65)公表番号　特表2003-523552 (P2003-523552A)
(43)公表日　平成15年8月5日 (2003.8.5)
(86)国際出願番号　PCT/US2000/026097
(87)国際公開番号　WO2001/022195
(87)国際公開日　平成13年3月29日 (2001.3.29)
審査請求日　平成19年9月13日 (2007.9.13)
(31)優先権主張番号　09/405,608
(32)優先日　平成11年9月24日 (1999.9.24)
(33)優先権主張国　米国(US)

(73)特許権者　502104413
アカンバ　コーポレイション
アメリカ合衆国　カリフォルニア　９５０３２，　ロス　ゲイトス，　ロス　ゲイトス　ブールバード　１５０５５
(74)代理人　100078282
弁理士　山本　秀策
(72)発明者　スミス，　ジャック　ジェイ.
アメリカ合衆国　カリフォルニア　９５０７０，　サラトガ，　オードリー　スミス　レーン　２８０２１
(72)発明者　バーライト，　リチャード　ティー.
アメリカ合衆国　カリフォルニア　９５１２４，　サン　ノゼ，　ドライ　クリーク　コート　２２５８

最終頁に続く

(54)【発明の名称】クライアントとサーバ間の接続を管理するためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
原サーバの内部バスに動作可能に結合されたインターフェースデバイスにおいて、少なくとも１つのクライアントと該原サーバとの間の接続を管理するための方法であって、該インターフェースデバイスは、ネットワークコントローラとアダプタバスとを含み、該ネットワークコントローラは、該アダプタバスに接続されており、
該方法は、
ネットワークを介して、該ネットワークコントローラとクライアントのうちの１つとの間のネットワーク接続を確立する工程と、
該ネットワーク接続を介して、該クライアントから通信を受ける工程と、
該インターフェースデバイスの該アダプタバスと該原サーバの該内部バスとの間のバス接続を確立する工程と、
該バス接続を介して、該原サーバに、該クライアント通信を転送する工程と、
該バス接続を介して、該原サーバからの該クライアント通信への応答を受信する工程と、
該ネットワーク接続を介して、該クライアントへの該応答を転送する工程とをさらに包含する、方法。
【請求項２】
前記原サーバからの応答を受信する工程は、該応答をバッファ内に格納する工程を包含する、請求項１に記載の方法。

【請求項３】
前記原サーバからの応答を受信する工程は、該応答が受信された後、前記バス接続を終了する工程を包含する、請求項２に記載の方法。
【請求項４】
前記クライアント通信は、ＨＴＴＰリクエストを含む、請求項１に記載の方法。
【請求項５】
前記原サーバからの応答は、ＨＴＭＬページを含む、請求項４に記載の方法。
【請求項６】
電子デバイスに請求項１に記載の工程を実行させるコードを内部に組み込んだ、コンピュータ読み出し可能媒体。
【請求項７】
電子デバイスに請求項２に記載の工程を実行させるコードを内部に組み込んだ、コンピュータ読み出し可能媒体。
【請求項８】
電子デバイスに請求項３に記載の工程を実行させるコードを内部に組み込んだ、コンピュータ読み出し可能媒体。
【請求項９】
電子デバイスに請求項４に記載の工程を実行させるコードを内部に組み込んだ、コンピュータ読み出し可能媒体。
【請求項１０】
電子デバイスに請求項５に記載の工程を実行させるコードを内部に組み込んだ、コンピュータ読み出し可能媒体。
【請求項１１】
ネットワークとの原サーバ通信を管理するために、原サーバの内部バスに動作可能に結合するためのアダプタカードであって、
該アダプタカードは、
該ネットワーク上のクライアントと通信するためのネットワークコントローラと、
データとコードとを格納するためのメモリデバイスであって、該コードはリバースプロキシアプリケーションを含む、メモリデバイスと、
該メモリデバイスに結合された、該コードを実行するための処理ユニットと、
該処理ユニットに結合されたプロトコルアダプタと、
該ネットワークコントローラ、該メモリデバイス、該処理ユニット、該プロトコルアダプタの間に結合されたアダプタバスと
を備え、
該リバースプロキシアプリケーションは、該処理ユニットによって実行されると、該原サーバと通信するために、該原サーバの該内部バスに結合するように該プロトコルアダプタを制御し、
該リバースプロキシアプリケーションは、クライアント通信を受信し、かつ、該プロトコルアダプタを介して該原サーバに該クライアント通信を転送するように動作可能であり、該クライアント通信は、該ネットワークコントローラを介して受信され、
該リバースプロキシアプリケーションは、該プロトコルアダプタを介して、該原サーバからの該クライアント通信への応答を受信し、かつ、該ネットワークコントローラを介して該クライアントへの該応答を転送するように動作可能である、アダプタカード。
【請求項１２】
前記コードは、通信プロトコルスタックをさらに含む、請求項１１に記載のアダプタカード。
【請求項１３】
前記通信プロトコルスタックは、標準ＴＣＰ／ＩＰプロトコルスタックを含む、請求項１２に記載のアダプタカード。
【請求項１４】

前記リバースプロキシアプリケーションは、セキュリティプロキシを含む、請求項１１に記載のアダプタカード。

【請求項１５】

前記リバースプロキシアプリケーションは、パススループロキシを含む、請求項１１に記載のアダプタカード。

【請求項１６】

前記リバースプロキシアプリケーションは、ＨＴＴＰプロキシを含む、請求項１１に記載のアダプタカード。

【請求項１７】

前記クライアントから受信したデータを格納するためのデータバッファをさらに備える、請求項１１に記載のアダプタカード。

【請求項１８】

前記リバースプロキシアプリケーションは、マスタープロセスモジュールであって、前記クライアントのうちの１つから受信した接続リクエストに応答し、かつ該クライアントとの接続を確立して、該確立された接続を維持するために、新たなクライアントプロセスモジュールを開始するように動作可能であるマスタープロセスモジュールを含む、請求項１１に記載のアダプタカード。

【請求項１９】

前記マスタープロセスモジュールは、さらに、前記確立された接続を前記リバースプロキシアプリケーションに通知するように動作可能である、請求項１８に記載のアダプタカード。

【発明の詳細な説明】

【０００１】

（発明の背景）

（発明の分野）

本発明は、概して、ネットワークサーバに関し、より詳細には、多数のクライアント接続をホストするサーバに関する。さらに詳細には、本発明は、多数の比較的遅いクライアント接続をホストするサーバ（例えば、インターネットウェブサーバ）に関する。

【０００２】

（背景技術の説明）

インターネットウェブサーバなどのネットワークファイルサーバが、多数の比較的遅いクライアント接続を管理することは一般的である。多数のオープン接続は、オープン接続を管理するだけで、サーバ中央処理装置（ＣＰＵ）に実質的な負担をかける。例えば、負荷が与えられたサーバ上のオープン接続の管理は、ＣＰＵの稼動能力を３０％～４０％以上消費し得る。この負担は、実質的に、サーバの一次機能（すなわち、クライアントにデータを提供する）を実行可能なＣＰＵサイクルのパーセンテージを減少させる。

【０００３】

サーバＣＰＵへの接続管理負担は、サーバソフトウェアルーチンの性能を劣化させ、同時にオープンであり得るクライアント接続の最大数を減少させる。結果として、ウェブを管理する会社は、数的に増加したクライアントを提供するために、別の冗長なサーバを提供する必要がある。別のウェブサーバを獲得および維持するためのコストは、相当な金額である。

【０００４】

プロキシサーバは、いくつかのクライアント接続管理機能を実行し、当該分野で公知である。しかし、このようなプロキシサーバは、サーバから離れて収納される必要があり、従って、サーバが管理する必要がある比較的遅くてエラーを起こしやすいネットワーク接続を介してサーバと通信する必要があるということは、当該分野で周知であり、一般的に受け入れられている。例えば、Ａｒｉ　Ｌｕｏｔｏｎｅｎの「Ｗｅｂ　Ｐｒｏｘｙ　Ｓｅｒｖｅｒｓ」（Ｐｒｅｎｔｉｃｅ　Ｈａｌｌ、１９９７）を参照のこと。本明細書中、本文献を参考として援用する。

【０００５】
従って、必要なものは、サーバＣＰＵを接続管理負担から救済するシステムおよび方法であり、従って、サーバが、数的に増加したクライアントをより効率的に管理することを可能にすることである。
【０００６】
（要旨）
本発明は、複数のクライアントとサーバとの間の接続を管理するシステムおよび方法を提供することにより、従来技術に関連する問題を克服する。本発明は、ホストＣＰＵから、ネットワークとホストバスとの間に置かれるアダプタカードに、接続管理負担を肩代わりさせることを促進する。
【０００７】
アダプタカードは、ネットワークコントローラ、メモリデバイス、処理装置、およびプロトコルアダプタを含む。メモリデバイスは、データおよびコードの格納を提供する。コードは、ネットワークコントローラを介してネットワーク上のクライアントと通信し、サーババスに直接結合されているプロトコルアダプタを介してサーバと通信するプロキシアプリケーションを含む。
【０００８】
処理装置によって実行されると、プロキシアプリケーションは、ネットワークを介してプロキシアプリケーションとクライアントとの間にネットワーク接続を確立することにより、そして、サーババスを介してプロキシアプリケーションとサーバとの間にバス接続を確立することにより、クライアント接続を管理する。さらに、メモリデバイスは、比較的少ないバス接続がサーバに対してオープンである一方で、多くのネットワーク接続がクライアントに対してオープンになることを可能にするデータバッファリングを提供する。特別な実施形態において、プロキシは、多数の遅いクライアント接続からバッファ内にクライアントデータを累積し、次いで、早いバス接続を介してサーバにクライアントデータを提示する。逆に、プロキシは、早いバス接続を介してサーバデータを受信し、サーバデータを一時的に格納し、次いで、遅いクライアント接続を介してクライアントにサーバデータを転送する。
【０００９】
より特別な実施形態において、コードは、クライアントおよびサーバと通信するためにアプリケーションプロキシによって使用される通信プロトコルスタックを含む。さらに特別な実施形態において、通信プロトコルスタックは、送信制御プロトコル／インターネットプロトコル（ＴＣＰ／ＩＰ）スタックである。
【００１０】
一実施形態において、サーバ接続は、プロキシが完全なクライアントリクエストを受信したことを判定して初めてオープンになる。サーバ接続は、次いで、プロキシが、サーバからのクライアントリクエストに対する応答を受信した後で、クローズになる。オプションとして、所定の数の永続的なサーバ接続がシステム起動時にオープンになり、プロキシは、これらの永続的な接続を使用して、サーバと通信する。
【００１１】
プロキシアプリケーションは、オプションとして、多数のアプリケーション特有のプロキシを含み、ＨＴＴＰプロキシ、安全プロキシ、および／またはパススループロキシを含むが、これらに限定されない。特定の実施形態において、プロキシのマスタープロセスモジュールは、クライアントデータからアプリケーション識別子（例えば、周知のポート数）を識別し、識別子の値に応じて、アプリケーション特有のプロキシを１つ以上呼び出す。
【００１２】
本発明は、下記の図面を参照して説明され、ここで、同様の参照番号は、実質的に同様の要素を指す。
【００１３】
（詳細な説明）

本発明は、異なる処理ユニット上で実行されるプロキシアプリケーションを用いて、サーバのメインプロセッサの接続管理に関する負担の多くを軽減することによって、従来技術に関連する問題を克服する。以下の説明において、多くの特定の詳細を記載し（例えば、特定の通信プロトコル、特定のソフトウェアおよびデータ構造など）、本発明の完全な理解を提供する。しかし、当業者であれば、これらの特定の詳細から離れて、本発明を実行し得ることを理解する。他の例において、本発明を不必要に不明瞭にしないように、周知のネットワークコンポーネントおよびプログラミングの実施の詳細（例えば、通信プロトコルスタックを介して接続を確立すること）は省略する。

【００１４】

図１は、物理ネットワーク媒体１０４を介して、インターネットワーク１０２に接続されているシステム１００を示すブロック図である。特定の実施において、システム１００は、インターネットウェブサーバであり、インターネットワーク１０２は、インターネットであるが、当業者であれば、本発明を、任意の種類のネットワークサーバで実施し得ることを理解する。

【００１５】

システム１００は、ファイルサーバ（例えば、ＨＴＴＰウェブサーバ）１０６およびアダプタカード１０８を含む。ファイルサーバ１０６は、アダプタカード１０８を介して、インターネットワーク１０２上のクライアント１０９（１－ｎ）に、データを提供し、かつ、このクライアント１０９（１－ｎ）からデータを受信する。アダプタカード１０８は、クライアント１０９（１－ｎ）とアダプタカード１０８との間にネットワーク接続を確立、かつこれを維持し、サーバ１０６とアダプタカード１０８との間にバス接続を確立する。したがって、このバスが接続された後、アダプタカード１０８は、サーバ１０６の代わりに、クライアント１０９（１－ｎ）から通信を受信し、通信をサーバ１０６に転送し、クライアント１０９の代わりに、サーバ１０６から応答を受信し、この応答をクライアント１０９に転送する。

【００１６】

サーバ１０６は、不揮発性メモリ１１０、作業メモリ１１２、サーバ大容量データ記憶装置１１４、処理ユニット１１６および１つ以上のユーザ入力／出力（Ｉ／Ｏ）デバイス１１８を含み、これらすべては、サーババス１２０（例えば、ＰＣＩバス）を介して相互通信する。不揮発性メモリ１１０（例えば、リードオンリーメモリおよび／または１つ以上のハードディスクドライブ）は、サーバ１０６がパワーダウンした場合でさえも保持されるデータおよびコード用の記憶装置を提供する。作業メモリ１１２（例えば、ランダムアクセスメモリ）は、サーバ１０６用の動作メモリを提供し、かつ、起動の間、作業メモリ１１２内にロードされる実行可能なコード（例えば、オペレーティングシステム）を含む。数あるプログラムの中でも、作業メモリ１１２は、サーバアプリケーション１２１および通信プロトコルスタック１２２を含む。サーバアプリケーション１２１は、サーバ１０６がネットワークサーバとして機能することを可能にする、ネットワークソフトウェアアプリケーション（例えば、ＦＴＰ、ＨＴＴＰなど）を含む。通信プロトコルスタック１２２は、インターネットワーク上において他のマシーンとの通信を容易にする標準のプロトコルスタック（例えば、ＴＣＰ／ＩＰ）である。標準のプロトコルスタックは、当該分野において周知である。例えば、Ｗ．Ｒｉｃｈａｒｄ　Ｓｔｅｖｅｎｓの「ＴＣＰ／ＩＰ　Ｉｌｌｕｓｔｒａｔｅｄ、Ｖｏｌ．１」（Ａｄｄｉｓｏｎ－Ｗｅｓｌｅｙ、１９９４年）を参照されたい。同文献は、本明細書中に参考として援用する。サーバ大容量データ記憶装置１１４は、サーバが、インターネットワーク１０２に取り付けられたクライアント１０９（１－ｎ）に提供するデータ（例えば、ＨＴＭＬページ、グラフィクスファイルなど）用のデータ記憶装置（例えば、１つ以上のハードディスクドライブ）を提供する。処理ユニット１１６は、作業メモリ１１２内で命令を実行し、これにより、サーバ１０６は、その主な機能（例えば、クライアントにデータを提供し、クライアントからデータを受信する機能）を実行する。Ｉ／Ｏデバイス１１８は通常、キーボード、モニタ、および／またはサーバ１０６とのユーザ双方向通信を容易にする他のデバイスを含む。上述のコンポ

ーネントのそれぞれは通常、インターネットウェブサーバなどのネットワークサーバ内に見られる。

【００１７】

アダプタカード１０８は、不揮発性メモリ１２３、作業メモリ１２４、処理ユニット１２６、バスプロトコルブリッジ１２８、ネットワークコントローラ１２９を含み、これらすべては、アダプタバス１３０を介して相互通信する。不発揮性メモリ１２３は、アダプタ１０８がパワーダウンした場合でさえも保持されるデータおよびコード（例えば、起動コード）用の記憶装置を提供する。処理ユニット１２６は、作業メモリ１２４内にあるコードを実行することによって、アダプタカード１０８に機能性を付与する。バスプロトコルブリッジ１２８は、アダプタバス１３０とサーババス１２０との間にインターフェースを提供し、ネットワークコントローラ１２９は、アダプタバス１３０とネットワーク媒体１０４との間にインターフェースを提供する。

【００１８】

作業メモリ１２４は、アダプタ１０８用の動作メモリを提供し、プロキシアプリケーション１３２および通信プロトコルスタック１３４を含む。プロキシ１３２およびプロトコルスタック１３４は、起動時に、不揮発性メモリ１２３から作業メモリ１２４内にロードされる。必要に応じて、プロキシ１３２およびプロトコルスタック１３４を、サーバ１０６の不揮発性メモリ１１０またはサーバ大容量データ記憶装置１１４を含む（ただし、これらに限定されない）１つ以上の別のソースからロードしてもよい。プロキシ１３２が処理ユニット１２６によって実行された場合、プロキシ１３２は、アダプタ１０８とサーバ１０６との間、およびアダプタ１０８とクライアント１０９との間に上述の接続を確立、かつ、管理する。

【００１９】

本発明のこの特定の実施形態において、プロトコルスタック１２２および１３４は、標準（例えば、ＴＣＰ／ＩＰ）のプロトコルスタックである。アダプタ１０８内に標準の通信プロトコルスタックを採用すると、広範囲にわたった、大多数のネットワークサーバにおいてすでに存在する標準の通信ソフトウェア（例えば、プロトコルスタック１２２）を用いることが容易になる。しかし、当業者であれば、この特定の要素（他の記載した用途（明示的に記述しない場合も含む））が本発明に必須要素ではないことを理解する。例えば、本発明は、サーバ１０６およびアダプタ１０８の両方で、カスタム通信ソフトウェア（例えば、サーバアプリケーション１２１と、プロトコルスタック１３４またはプロキシ１３２のいずれかとの間の直接通信）を用いて、実行し得る。さらに、本発明の特定の実施形態において、サーバ１０６のリソース（例えば、サーバ大容量データ記憶装置１１４）への直接アクセスをプロキシ１３２に提供することによって、この要素を省略することが可能である。

【００２０】

バス接続１３６を介して、バスプロトコルブリッジ１２６とサーババス１２０との間のサーバ１０６に、アダプタカード１０８を接続する。この特定の実施形態において、バス接続１３６は、通常のバス拡張スロット（例えば、ＰＣＩスロット）である。しかし、当業者であれば、ＩＳＡスロット、ＵＳＢポート、シリアルポートまたはパラレルポートを含む（ただし、これらに限定されない）他の種類のバス接続によって、本発明を実施し得ることを理解する。バス接続１３６によって、高速、大きなパケットサイズ、プロキシ１３２とサーバアプリケーション１２１との間の比較的にエラーのない（ネットワーク接続と比較した場合）通信が促進され、サーバ１０６の処理ユニット１１６にかかる接続管理の負担が軽減される。要するに、プロキシ１３２（処理ユニット１１６上で実行される）は、遅く、エラーを生じがちなネットワーク接続上でクライアント１０９と通信して、次いで、高速バス接続１３６上で、クライアント１０９の代わりにサーバアプリケーション１２１と通信する。

【００２１】

図２は、プロキシ１３２およびプロトコルスタック１３４を詳細に示す作業メモリ１２４

のブロック図である。当業者であれば、プロキシ１３２の種々のソフトウェアモジュールが、相互接続された機能ブロックとして図示され、一方、ソフトウェアモジュールは実際、ソフトウェアモジュールが処理ユニット１２６（図１）によって実行された場合に、相互に通信し得る作業メモリ１２４内に格納される、実行可能なコードのブロックであることを理解する。

【００２２】

プロキシ１３２は、マスタープロセスモジュール２０２、複数のクライアントプロセスモジュール２０４（１－ｎ）、データバッファ２０６およびアプリケーションプロキシモジュール２０８を含む。マスタープロセスモジュールは、プロキシ１３２の種々のモジュールの全体制御および調整を提供する。インターネットワーク１０２（図１）上のクライアント１０９からの接続リクエストに応答して、マスタープロセス２０２は、接続リクエストを受け取り、データバッファ２０６内のこのクライアント接続用にデータ構造を初期化し、新規の別々のクライアント接続プロセス２０４を開始して接続を処理し、次いで、特定のクライアント接続が確立されたことをアプリケーションプロキシ２０８に通知する。各クライアントプロセス２０４は、このようなクライアント接続を処理する。アプリケーションプロキシ２０８は、サーバ１０６とのバス接続を確立、かつ管理する。データバッファ２０６は、クライアント１０９から受信され、サーバ１０６向けのデータ、サーバ１０６から受信され、クライアント１０９向けのデータ、確立されたクライアントおよび／またはサーバ接続に関する接続データ用の記憶装置を提供する。

【００２３】

通信プロトコルスタック１３４は、ソケット層２１０、ＴＣＰ層２１２、ＩＰ層２１４、ネットワークドライバ２１６およびサーババスドライバ２１８含むデバイス層を含むＴＣＰ／ＩＰスタックである。プロトコルスタック１３４の個々の層のそれぞれの機能性は、当該技術において周知である。したがって、本明細書において、詳細には説明しない。プロキシ１３２の種々のモジュールとサーバアプリケーション１２１との間の接続は、ソケット層２１０、ＴＣＰ層２１２、ＩＰ層２１４およびサーババスドライバ２１８を介して確立される。プロキシ１３２の種々のモジュール間の接続は、ソケット層２１０、ＴＣＰ層２１２、ＩＰ層２１４およびネットワークドライバ２１６を介して、クライアント１０９によって接続が確立される。

【００２４】

図３は、ハイパーテキスト転送プロトコル（ＨＴＴＰ）プロキシ２０８（１）、パスースループロキシ２０８（２）、セキュリティプロキシ２０８（３）および「他の」のプロキシ２０８（ｆ）を含む複数のアプリケーション指定プロキシ２０８（１～ｆ）を含む、アプリケーションプロキシモジュール２０８を示すブロック図である。マスタープロセス２０２は、クライアント接続を機能させる、１つ以上のアプリケーション指定プロキシ２０８（１～ｆ）を構築することによって、アプリケーションプロキシ２０８に確立されたクライアント接続を通知する。アプリケーション指定プロキシ（例えば、ＨＴＴＰプロキシ２０８（１））を構築する一つの手段は、アプリケーション指定プロキシのキューを処理する際に、クライアントプロセス識別子を入力することである。

【００２５】

マスタープロセス２０２は、クライアント接続リクエストに含まれるポート番号から、特定のクライアントプロセス用にいずれのアプリケーション指定プロキシを実行するかを判定する。特定のネットワークアプリケーションおよび／またはプロトコル（例えば、ファイル転送プロトコル（ＦＴＰ）、ＨＴＴＰなど）を識別するために、周知のポート番号を用いることが標準的な実施である。例えば、ポート番号８０は、ＨＴＴＰ接続リクエストに対応する。したがって、マスタープロセス２０２は、接続リクエストがポート８０を示したことに応答して開始される、すべてのクライアントプロセス２０４を、ＨＴＴＰプロキシ２０８（１）に通知する。

【００２６】

ＨＴＴＰプロキシ２０８（１）は、通知されたクライアントプロセスのそれぞれをモニタ

リングする。クライアントプロセス（例えば、２０４（ｎ））によって、完全なＨＴＴＰリクエストがデータバッファ２０６内に受信および格納されたことを、ＨＴＴＰプロキシ２０８（１）が判定すると、ＨＴＴＰプロキシ２０８（１）は、サーバへの接続を開始し、リクエストをサーバに伝送し、サーバから応答を受信し、データバッファ２０６内に応答を格納し、次いでサーバ接続を終了する。次いで、サーバ応答を、関連するクライアントプロセス２０４（ｎ）によって、クライアント１０９（ｎ）に伝送する。

【００２７】

マスタープロセス２０２が、他のアプリケーション指定プロキシのいずれとも対応しないポート番号を有する接続リクエストを受信すると、マスタープロセス２０２は、パス－スループロキシ２０８（２）に通知する。パス－スループロキシ２０８（２）は単に、サーバ接続を開始し、データバッファ２０６からサーバ１０６に、関連するクライアントプロセス２０４から受信されたデータを転送し、次いでサーバ接続を終了する。

【００２８】

マスタープロセス２０２は、関連するポート番号にかかわらず、すべてのクライアント接続を、いくつかのアプリケーション指定プロキシに通知し得る。例えば、セキュリティプロキシ２０８（３）は、例えば、他のアプリケーション指定プロキシのうちの一つを実施する前に、認定表示のない接続リクエストに応答して開始される任意のクライアントプロセスを終了することによって、すべてのクライアント接続リクエストをスクリーニングするために動作する。

【００２９】

図３に「他」のプロキシ２０８（ｆ）を含み、アプリケーションプロキシ２０８が、キャッシュ（ｃａｃｈｉｎｇ）ＨＴＴＰプロキシアプリケーション、電子メールアプリケーションおよびファイル転送アプリケーションを含む（ただし、これらに限定されない）特定のアプリケーションに所望である、任意の現在公知のプロキシ、または将来開発されるプロキシを含み得ることを示す。

【００３０】

図４は、データスループロキシ１３２の伝送に影響を与えるように、データバッファ２０６内に実施される、クライアントデータ構造４０２（１－ｎ）およびプロキシデータ構造４０４（１－ｆ）の例を示す。マスタープロセス２０２は、アプリケーションプロキシ２０８内で、各クライアントプロセス２０４用に一つのクライアントデータ構造４０２および各アプリケーション指定プロキシ用に一つのプロキシデータ構造４０４を作成および初期化する。

【００３１】

各クライアントデータ構造４０２は、クライアントソケット４０６、サーバソケット４０８、接続状態４１０、入力キュー４１２、出力キュー４１４およびアプリケーションプロキシデータ４１６を含む。各クライアント接続（例えば、接続（ｎ））に関して、クライアントソケット４０６（ｎ）およびサーバソケット４０８（ｎ）はそれぞれ、クライアント１０９（ｎ）およびサーバ１０６のＩＰアドレスおよびポート番号を含み、したがって、クライアントプロセス２０４（ｎ）のうちの一つと、各クライアントデータ構造４０２（ｎ）が一意的に関連付けられる。接続状態４１０（ｎ）は、接続（ｎ）の現在の状態（例えば、受信された完全なリクエスト、受信された応答など）を示す。特定のデータ構造４０２（ｎ）と関連付けられたクライアントプロセス２０４（ｎ）によって、クライアント１０９（ｎ）から受信されたデータを格納および蓄積するために、入力キュー４１２（ｎ）を用いる。クライアントプロセス２０４（ｎ）によって、クライアント１０９（ｎ）に転送されるべきアプリケーションプロキシ２０８からデータを格納するために、出力キュー４１４（ｎ）を用いる。特定のアプリケーションプロキシ（例えば、フラグなど）に特定の任意の情報を格納するために、アプリケーションプロキシデータ４１６（ｎ）を提供する。

【００３２】

各プロキシデータ構造（例えば、４０４（ｆ））は、クライアントキュー４１８（ｆ）、

クライアント準備キュー４２０（ｆ）および読み出し保留キュー４２２（ｆ）を含む。クライアントキュー４１８（ｆ）は、プロキシデータ構造４０４（ｆ）が対応する、特定のアプリケーションプロキシ（ｆ）と関連する各クライアントプロセス２０４用のクライアントプロセス記述子（例えば、関連するクライアントデータ構造４０２のポインタ）を含む。クライアント準備キュー４２０（ｆ）は、関連するアプリケーションプロキシ（ｆ）によって処理する（例えば、サーバ１０６に伝送する）準備が整った入力キュー４１２内にデータを有する、各クライアントデータ構造４０２用のクライアントプロセス記述子を含む。

【００３３】

当業者であれば、上述のクライアントデータ構造４０２およびプロキシデータ構造４０４は本質的には例示であり、本発明によって、他のデータ構造を採用し得ることを理解する。このような別のデータ構造の構成は必然的に、採用される特定のアプリケーション指定プロキシの機能および構造に依存する。

【００３４】

図５は、本発明による、クライアントとサーバとの間の接続を管理する特定の方法５００をまとめたフローチャートである。プロキシ１３２は、第１の工程５０２において、クライアント１０９とのネットワーク接続を確立し、次いで第２の工程５０４において、ネットワーク接続を介して、クライアント１０９から通信（例えば、ＨＴＴＰリクエスト）を受信する。次に、プロキシ１３２は、第３の工程５０６において、サーバ１０６とのバス接続を確立し、次いで第４の工程５０８において、バス接続を介して、サーバ１０６に、受信したクライアント通信を転送する。次いで、プロキシ１３２は、第５の工程５１０において、サーバ１０６から、クライアント通信への応答（例えば、ＨＴＭＬデータ）を受信し、第６の工程５１２において、クライアントネットワーク接続を介してクライアント１０９に応答を伝送する。最後に、プロキシ１３２は、第７の工程５１４において、終了すべき（例えば、シャットダウンすべき）信号があるか否かを判定し、終了すべき信号がある場合には、方法５００は終了する。第７の工程５１４において、終了すべき信号がない場合には、方法５００は、第１の工程５０２に戻り、別のクライアント１０９とのネットワーク接続を確立する。

【００３５】

図６は、方法５００の第１の工程５０２を実行する（クライアントとのネットワーク接続を確立する）一つの特定の方法６００をまとめたフローチャートである。第１の工程において、マスタープロセス２０２は、インターネットワーク１０２に接続する。次いで、第２の工程６０４において、マスタープロセス２０２は、インターネットワーク１０２上のトラフィックを聞き、クライアント１０９から接続リクエストがあるか否かを判定する。クライアントの接続リクエストがない場合には、方法６００は終了する。クライアント１０９（ｎ）から接続リクエストがある場合には、第３の工程６０６において、マスタープロセス２０２は、クライアント１０９（ｎ）から接続リクエストを受け取り、クライアントプロセス２０４（ｎ）を開始し、接続を処理し、データバッファ２０６内のクライアントデータ構造４０２（ｎ）を初期化する。次に、第４の工程６０８において、マスタープロセス２０２は、クライアント接続リクエストからプロキシアプリケーション識別子（例えば、ポート番号）を区別し、識別子の値に応じて、各プロキシデータ構造４０４のクライアントキュー４１８内に、クライアントプロセス記述子（例えば、クライアントデータ構造４０２（ｎ）のポインタ）を書き込むことによって、一つ以上のアプリケーションプロキシ２０８（１～ｆ）に通知する。最後に、第５の工程６１０において、マスタープロセス２０２は、最大許容数のクライアント接続がオープンであるか否かを判定する。最大数のクライアント接続がオープンである場合には、方法６００は終了する。最大数のクライアント接続がオープンでない場合には、方法６００は、第２の工程６０４に戻り、別の接続リクエストを聞く。

【００３６】

図７は、方法５００の（クライアント１０９からの通信を受ける）第２の工程５０４を実

行する方法７００を概説するフローチャートである。第１の工程７０２で、マスタープロセス２０２は、データを受信するために処理されるべきクライアントプロセス２０４が存在するどうかを判断する。マスタープロセス２０２がクライアントプロセス２０４（１－ｎ）の全てをすでに処理している場合、方法７００は終了する。処理していない場合、第２の工程７０４で、マスタープロセス２０２が第１のクライアントプロセス２０４（１）を呼ぶ。次いで、第３の工程７０６で、クライアントプロセス２０４（１）は、クライアント１０９（１）からデータが入来しているかどうかを判断するために、クライアント接続（例えば、ＴＣＰバッファ）をチェックする。第１のクライアントプロセス２０４（１）に対する入来データがない場合、方法７００は、残りのクライアントプロセス２０４（２－ｎ）を処理するために、第１の工程７０２に戻る。第３の工程７０６で、クライアントプロセス２０４（１）が、クライアント１０９（１）からの入来データが存在すると判断した場合、第４の工程７０８で、クライアントプロセス２０４（１）は、入力キュー４１２（１）がクライアントデータを受信するために利用可能であるかを判断するために、クライアントデータ構造４０２（１）をチェックする。入力キュー４１２（１）が利用可能でない場合、方法７００は、残りのクライアントプロセス２０４（２－ｎ）を処理するために、第１の工程７０２に戻る。第４の工程７０８で、クライアントプロセス２０４（１）は、入力キュー４１２（１）がデータの受信のために利用可能であると判断した場合、次いで、第５の工程７１０で、クライアントプロセス２０４（１）は、入来するクライアントデータを入力キュー４１２（１）に転送する。次いで、第６の工程７１２で、クライアントプロセス２０４（１）は、入力キューに蓄積されたデータが完全なリクエスト（すなわち、サーバ１０６への転送準備が完了したデータ、例えば、完全なＨＴＴＰリクエスト）を構成しているかどうかを判断する。データが完全なリクエストを構成していない場合、方法７００は、残りのクライアントプロセス２０４（２－ｎ）を処理するために、第１の工程７０２に戻る。しかしながら、クライアントプロセス２０４（１）は、第６の工程７１２で、入力キュー４１２（１）内のデータが完全なリクエストを構成していると判断した場合、第７の工程７１４で、クライアントプロセスは、例えば、完全なリクエストが存在することを示すように接続状態４１０（１）を設定することにより、完全なリクエストが存在することをプロキシアプリケーション２０８に通知する。次いで、方法７００は、処理すべきクライアントプロセス２０４（２－ｎ）がさらに存在するかどうかを判断するために、第１の工程７０２に戻る。

【００３７】

図８は、方法５００の（サーバ１０６とのバス接続を確立する）第３の工程５０６を実行する方法８００を概説するフローチャートである。第１の工程８０２で、１つ目のアプリケーションプロキシ２０８（１）が、プロキシデータ構造４０４（１）のクライアントキュー４１８（１）から、第１のクライアント記述子を引き出す。次いで、第２の工程８０４で、プロキシ２０８（１）は、第１のクライアントがその入力キュー４１２内に完全なリクエストを有するかどうかを判断するために、第１のクライアント記述子により特定されるクライアントデータ構造４０２の接続状態４１２をチェックする。接続状態４１２が完全なリクエストを示す場合、第３の工程８０６で、プロキシ２０８（１）は、クライアント準備完了キュー４２０（１）にクライアント記述子を加える。次に、第４の工程８０８で、プロキシ２０８（１）は、サーバ１０６に対して、最大数の接続が開かれているかどうかを判断する。最大数のサーバ接続がすでに開かれている場合、方法８００は終了する。最大数のサーバ接続がまだ開かれていない場合、第５の工程８１０で、プロキシ２０８（１）は、サーバ１０６とのバス接続を開き、関連づけられたクライアントデータ構造４０２のサーバソケット４０８に接続情報を書き込む。次に、第６の工程８１２で、プロキシ２０８（１）は、クライアントキュー４１８（１）内の最後の記述子をチェックしたかどうかを判断する。最後の記述子がチェックされている場合、方法８００は終了する。最後の記述子がチェックされていなければ、方法８００は、クライアントキュー４１８（１）内の次の記述子を引き出すために、第１の工程８０２に戻る。第２の工程８０４で、プロキシ２０８（１）が、完全なクライアントリクエストが受信されていないと判断した

場合、方法８００は、直接、第６の工程８１２に進む。一旦、プロキシデータ構造４０４（１）のクライアントキュー４１８（１）内の記述子の全てが処理されると、方法８００または同様の方法が、他のアプリケーションプロキシ２０８（２－ｆ）の各々に対して繰り返される。

【００３８】

図９は、方法５００の（サーバ１０６にクライアント通信を転送する）第４の工程５０８を実行する方法９００を概説するフローチャートである。第１の工程９０２で、プロキシ２０８（１）は、プロキシデータ構造４０４（１）のクライアント準備完了キュー４２０（１）から、第１のクライアント記述子を引き出す。次いで、第２の工程９０４で、プロキシ２０８（１）は、サーバ接続が開いているかどうかを判断するために、第１のクライアントのデータ構造４０２のサーバソケット４０８をチェックする。サーバ接続が開いている場合、第３の工程９０６で、プロキシ２０８（１）は、開いたサーバ接続を介して、クライアント入力キュー４１２からサーバ１０６にクライアントデータ（例えば、ＨＴＴＰリクエスト）を転送する。次に、第４の工程９０８で、プロキシ２０８（１）は、クライアント準備完了キュー４２０（１）から読み出し保留キュー４２２（１）にクライアント記述子を移動する。次いで、第５の工程９１０で、プロキシ２０８（１）は、クライアント準備完了キュー４２０（１）内の最後のクライアントがチェックされたかどうかを判断する。チェックされていない場合、方法９００は、クライアント準備完了キュー４２０（１）内の次のクライアントをチェックするために、第１の工程９０２に戻る。最後のクライアントがチェックされている場合、方法９００は終了する。第２の工程９０４で、プロキシ２０８（１）が、特定のクライアントのために開いたサーバ接続が存在しないと判断した場合、方法９００は、クライアント準備完了キュー４２０（１）内の最後のクライアントがチェックされたかどうかを判断するために、直接、第５の工程９１０に進む。一旦、プロキシデータ構造４０４（１）のクライアント準備完了キュー４２０（１）内の記述子の全てが処理されると、方法９００または同様の方法が、他のアプリケーションプロキシ２０８（２－ｆ）の各々に対して繰り返される。

【００３９】

図１０は、方法５００の（サーバ１０６から応答を受信する）第５の工程５１０を実行する方法１０００を概説するフローチャートである。第１の工程１００２で、プロキシ２０８（１）は、読み出し保留キュー４２２（１）が空であるかどうかを判断し、空である場合、方法１０００が終了する。読み出し保留キュー４２２（１）が空でない場合、第２の工程１００４で、プロキシ２０８（１）が、読み出し保留キューから第１のクライアント記述子を引き出す。次に、第３の工程１００６で、プロキシ２０８（１）が、第１のクライアント記述子により特定されるクライアントデータ構造４０２のサーバソケット４０８内で特定された、開いたサーバ接続上に入来サーバデータ（すなわち、クライアントリクエストへの応答）が存在するかどうかを判断するために、その接続をチェックする。その接続上に入来サーバデータが存在しない場合、方法１０００は、読み出し保留キュー内の次のクライアントをチェックするために、第２の工程１００４に戻る。入来サーバデータが存在する場合、第４の工程１００８で、プロキシ２０８（１）は、クライアント記述子により特定されるクライアントデータ構造４０２の出力キュー４１４が利用可能であるかどうかを判断するためのチェックを行う。出力キュー４１４が利用可能でない場合、方法１０００は、読み出し保留キュー４２２（１）内の次のクライアント記述子をチェックするために、第２の工程１００４に戻る。出力キュー４１４が利用可能である場合、第５の工程１０１０で、プロキシ２０８（１）は、クライアントデータ構造４０２の出力キュー４１４に、入来サーバデータを移動する。次に、第６の工程１０１２で、プロキシ２０８（１）は、サーバデータが「ファイルの終わり」インジケータを含むかどうかを判断する。含んでいない場合、第７の工程１０１４で、プロキシ２０８（１）は、読み出し保留キュー４２２（１）内の最後のクライアント記述子が処理されたかどうかを判断するためのチェックを行う。処理されている場合、方法１０００は終了する。処理されていない場合、方法１０００は、読み出し保留キュー４２２（１）から次のクライアント記述子を読み

出すために、工程１００４に戻る。

【００４０】

第６の工程１０１２で、プロキシ２０８（１）が、サーバデータが「ファイルの終わり」インジケータを含むと判断した場合、方法１０００は、プロキシ２０８（１）が読み出し保留キューからクライアント記述子を取り除く、第８の工程１０１６に進み、次いで、第９の工程１０１８で、サーバ接続を閉じる。第９の工程１０１８の後、方法１０００は、第７の工程１０１４に戻る。一旦、プロキシデータ構造４０４（１）の読み出し保留キュー４２２（１）内の記述子の全てが処理されると、方法１０００または同様の方法が、他のアプリケーションプロキシ２０８（２－ｆ）の各々に対して繰り返される。

【００４１】

図１１は、方法５００の（クライアント１０９にデータを送信する）第６の工程５１２を実行する方法１１００を概説するフローチャートである。第１の工程１１０２で、マスタープロセス２０２は、第１のクライアントプロセス２０４（１）を呼ぶ。次いで、第２の工程１１０４で、第１のクライアントプロセス２０４（１）は、クライアントデータ構造４０２（１）の出力キュー４１４（１）内にデータが存在するかどうかを判断する。出力キュー４１４（１）内にデータが存在しない場合、方法１１００は、マスタープロセス２０２が残りのクライアントプロセス２０４（２－ｎ）のうちの次のクライアントプロセスを呼ぶ、第１の工程１１０２に戻る。しかしながら、第２の工程１１０４で、クライアントプロセス２０４（１）が、出力キュー４１４（１）内にデータが存在すると判断した場合、第３の工程１１０６で、クライアントプロセス２０４（１）は、クライアントネットワーク接続がデータを受信する準備が完了しているかどうかを判断する。クライアントネットワーク接続が準備完了していない場合、方法１１００は第１の工程１１０２に戻る。クライアントネットワーク接続が準備完了している場合、第４の工程１１０８で、クライアントプロセス２０４（１）は、クライアント接続（例えば、ＴＣＰ出力バッファ）に、出力キュー４１４（１）内のデータの少なくとも一部を移動する。次に、第５の工程１１１０で、マスタープロセス２０２は、最後のクライアントプロセスが呼ばれたかどうかを判断する。呼ばれている場合、方法１１００は終了する。呼ばれていない場合、方法１１００は、残りのクライアントプロセス２０３（２－ｎ）のうちの次のクライアントプロセスを呼ぶために、第１の工程１１０２に戻る。

【００４２】

これで、本発明の特定の実施形態の説明を終える。説明した特徴の多くは、本発明の範囲を逸脱することなく、置換、変更、または省略され得る。例えば、アダプタ１０８の機能素子（例えば、処理ユニット１２６およびプロキシ１３２）は、着脱可能アダプタカードに設けられる代わりに、サーバに直接組み入れられ得る。さらに、代替的なデータ構造が、提供された例示的なデータ構造と置き換えられ得る。また、本明細書中において開示された方法およびルーチンの特定の順序は、本発明の必要不可欠な要素であるとは見なされない。さらに別の例として、マスタープロセス２０２は、起動時に、サーバ１０６との好適な数の持続的バス接続を開き、アプリケーションプロキシ２０８（１－ｆ）によるこれらの接続の使用を管理するように構成され得るため、サーバ１０６がバス接続を繰り返し開閉する必要がなくなる。示された特定の実施形態からのこれらおよび他の逸脱は、特に、前述の開示より当業者には明らかである。

【図面の簡単な説明】

【図１】　図１は、本発明によるサーバおよびアダプタカードのブロック図である。

【図２】　図２は、図１のアダプタカードの作業メモリのブロック図であり、プロキシモジュールをより詳細に示す。

【図３】　図３は、図２のアプリケーションプロキシモジュールをより詳細に示すブロック図である。

【図４】　図４は、図２のデータバッファ内に格納されているデータの少なくともいくつかに関する例示的なデータ構造を示すブロック図である。

【図５】　図５は、本発明によるクライアントとサーバとの間の接続を管理する１つの

方法を要約するフローチャートである。

【図６】　図６は、図５の方法の第１のステップを実行する１つの方法を要約するフローチャートである。

【図７】　図７は、図５の方法の第２のステップを実行する１つの方法を要約するフローチャートである。

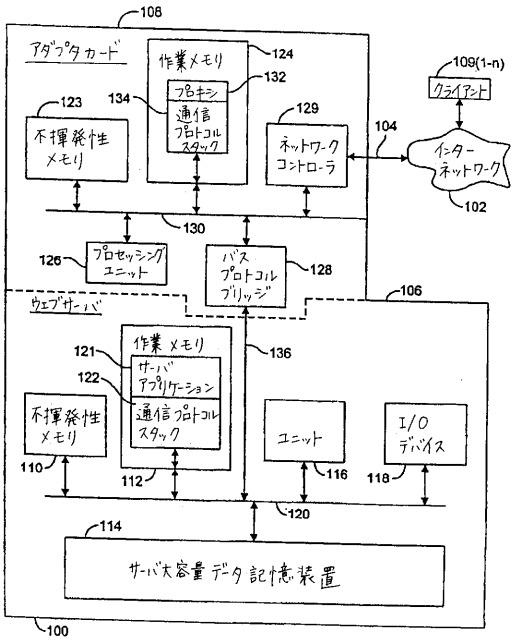
【図８】　図８は、図５の方法の第３のステップを実行する１つの方法を要約するフローチャートである。

【図９】　図９は、図５の方法の第４のステップを実行する１つの方法を要約するフローチャートである。

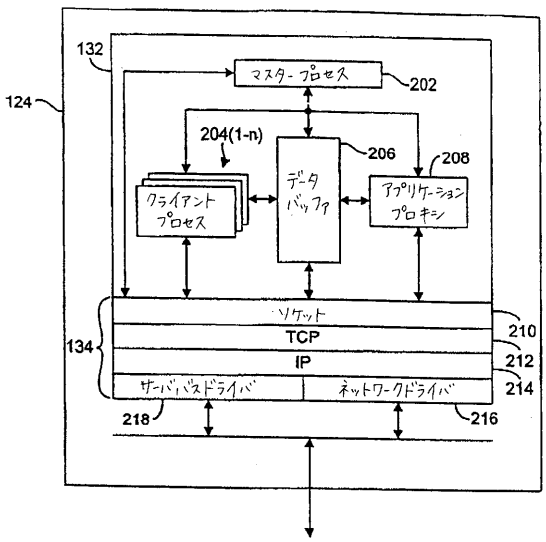
【図１０】　図１０は、図５の方法の第５のステップを実行する１つの方法を要約するフローチャートである。

【図１１】　図１１は、図５の方法の第６のステップを実行する１つの方法を要約するフローチャートである。

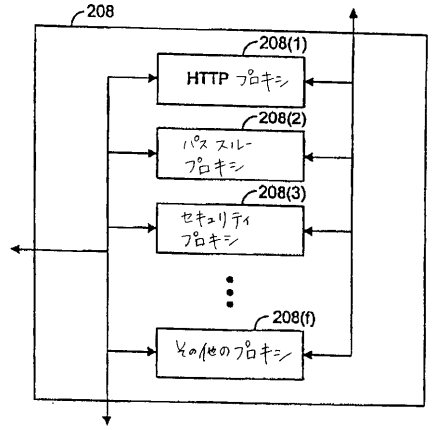
【図１】



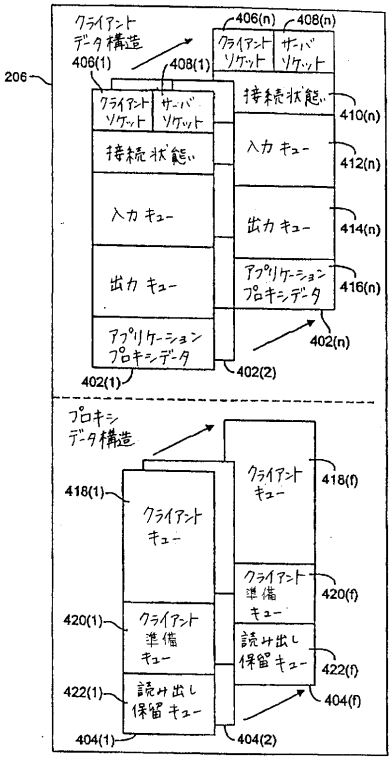
【図２】



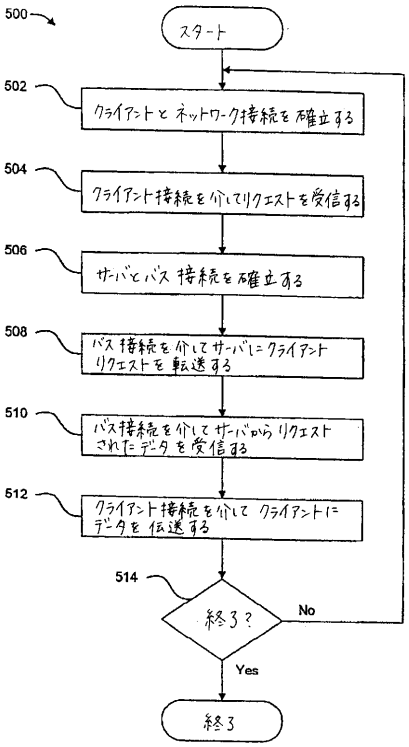
【図３】


208
208(1)
HTTP プロキシ
208(2)
パス スルー プロキシ
208(3)
セキュリティ プロキシ
208(f)
その他のプロキシ

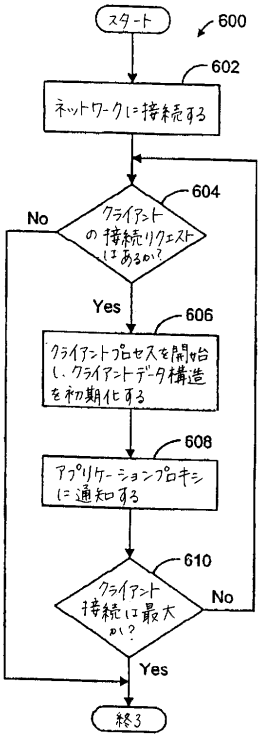
【図４】


クライアント データ構造
406(n)
408(n)
クライアント ソケット
サーバ ソケット
406(1)
408(1)
206
接続状態
410(n)
入力 キュー
412(n)
出力 キュー
414(n)
アプリケーション プロキシデータ
416(n)
402(n)
402(2)
402(1)
プロキシ データ構造
418(1)
クライアント キュー
418(f)
クライアント 準備 キュー
420(f)
420(1)
読み出し 保留 キュー
422(f)
422(1)
404(f)
404(1)
404(2)

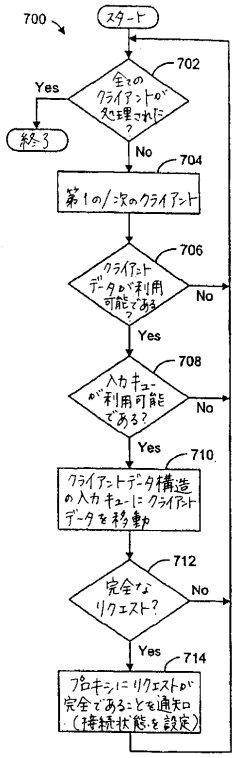
【図５】


500
スタート
502
クライアントと ネットワーク接続を 確立する
504
クライアント接続を 介してリクエストを受信する
506
サーバと パス 接続を確立する
508
パス 接続を介して サーバに クライアント リクエストを 転送する
510
パス接続を介して サーバから リクエスト されたデータ を受信する
512
クライアント接続を介して クライアントに データを 伝送する
514
終了？
No
Yes
終了

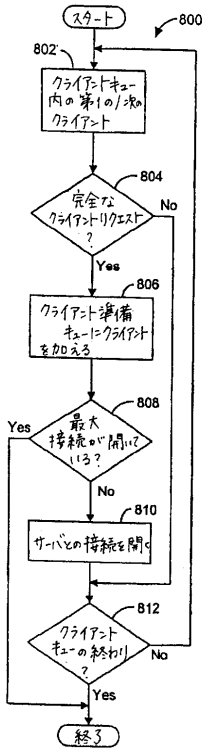
【図６】


スタート
600
602
ネットワークに接続する
604
クライアント の 接続リクエスト はあるか？
No
Yes
606
クライアントプロセスを開始 し、クライアントデータ構造 を初期化する
608
アプリケーションプロキシ に通知する
610
クライアント 接続は最大 か？
No
Yes
終了

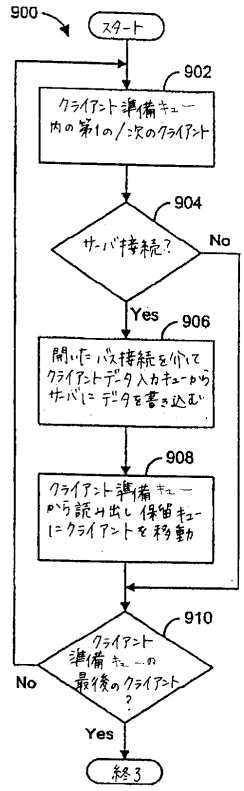
【図７】


700
スタート
702
全てのクライアントが処理された？
Yes
終了
No
704
第1の/次のクライアント
706
クライアントデータが利用可能である？
No
Yes
708
入力キューが利用可能である？
No
Yes
710
クライアントデータ構造の入力キューにクライアントデータを移動
712
完全なリクエスト？
No
Yes
714
プロキシにリクエストが完全であることを通知（接続状態を設定）

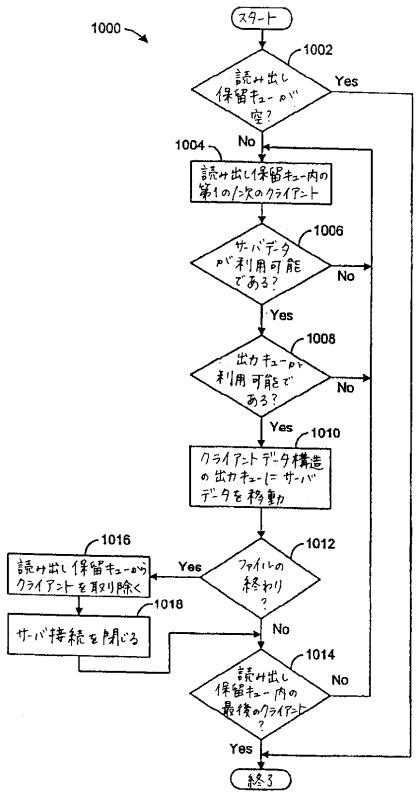
【図８】


スタート
800
802
クライアントキュー内の第1の/次のクライアント
804
完全なクライアントリクエスト？
No
Yes
806
クライアント準備キューにクライアントを加える
808
最大接続が開いている？
Yes
No
810
サーバとの接続を開く
812
クライアントキューの終わり？
No
Yes
終了

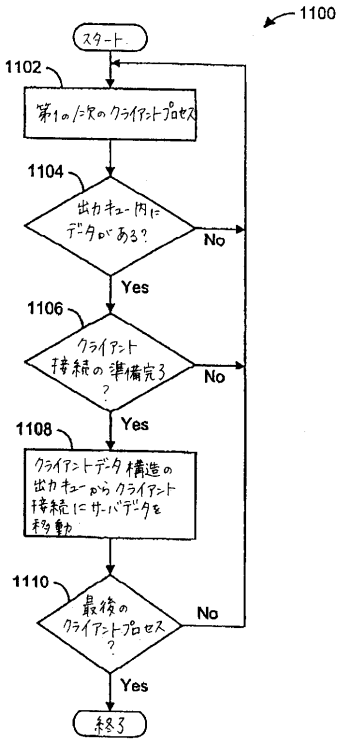
【図９】


900
スタート
902
クライアント準備キュー内の第1の/次のクライアント
904
サーバ接続？
No
Yes
906
開いたパス接続を介してクライアントデータ入力キューからサーバにデータを書き込む
908
クライアント準備キューから読み出し保留キューにクライアントを移動
910
クライアント準備キューの最後のクライアント？
No
Yes
終了

【図１０】


1000
スタート
1002
読み出し保留キューが空？
Yes
No
1004
読み出し保留キュー内の第1の/次のクライアント
1006
サーバデータが利用可能である？
No
Yes
1008
出力キューが利用可能である？
No
Yes
1010
クライアントデータ構造の出力キューにサーバデータを移動
1012
ファイルの終わり？
Yes
No
1016
読み出し保留キューからクライアントを取り除く
1018
サーバ接続を閉じる
1014
読み出し保留キュー内の最後のクライアント？
No
Yes
終了

【図１１】


1100
スタート
1102
第1の/次のクライアントプロセス
1104
出力キュー内にデータがある？
No
Yes
1106
クライアント接続の準備完了？
No
Yes
1108
クライアントデータ構造の出力キューからクライアント接続にサーバデータを移動
1110
最後のクライアントプロセス？
No
Yes
終了

フロントページの続き

(72)発明者 ウォーリー, ダブリュー. スペンサー ザ サード
アメリカ合衆国 カリフォルニア ９４０１９, ハーフ ムーン ベイ, コリアス アベニュー ３１１
(72)発明者 マクドネル, エオイン ビー.
アメリカ合衆国 カリフォルニア ９４０８７, サニーベイル, ヨークタウン ドライブ １０５７
(72)発明者 ヴァスタノ, ジョン エイ.
アメリカ合衆国 カリフォルニア ９５０５１, サンタ クララ, カブリロ アベニュー ３４９７
(72)発明者 ウェザーフォード, ウィリアム ティー.
アメリカ合衆国 カリフォルニア ９４４０２, サン マテオ, フォージ ロード １５５６

審査官 木村 雅也

(56)参考文献 特開平１１－０６５９６９（ＪＰ, Ａ）
特開平１１－１８７０６１（ＪＰ, Ａ）
特開平１１－２４２６４４（ＪＰ, Ａ）
特表平０７－５０４２８６（ＪＰ, Ａ）
米国特許第０５９４１９８８（ＵＳ, Ａ）
米国特許第０５８９８８３０（ＵＳ, Ａ）
米国特許第０５８７２９１９（ＵＳ, Ａ）
米国特許第０６１８２１４１（ＵＳ, Ｂ１）
特開平０１－２２６０５９（ＪＰ, Ａ）

(58)調査した分野(Int.Cl., ＤＢ名)
G06F 13/00
G06F 15/00