

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4611593号
(P4611593)

(45) 発行日 平成23年1月12日 (2011. 1. 12)

(24) 登録日 平成22年10月22日 (2010. 10. 22)

(51) Int. Cl.	F I
G 0 6 F 13/00 (2006. 01)	G 0 6 F 13/00 3 5 3 C
H 0 4 L 29/10 (2006. 01)	H 0 4 L 13/00 3 0 9 Z

請求項の数 15 (全 13 頁)

(21) 出願番号	特願2001-546119 (P2001-546119)	(73) 特許権者	598036300
(86) (22) 出願日	平成12年11月18日 (2000. 11. 18)		テレフオンアクチーボラゲット エル エム エリクソン (パブル)
(65) 公表番号	特表2003-517688 (P2003-517688A)		スウェーデン国 ストックホルム エスー 1 6 4 8 3
(43) 公表日	平成15年5月27日 (2003. 5. 27)	(74) 代理人	100109726
(86) 国際出願番号	PCT/EP2000/011503		弁理士 園田 吉隆
(87) 国際公開番号	W02001/045353	(74) 代理人	100101199
(87) 国際公開日	平成13年6月21日 (2001. 6. 21)		弁理士 小林 義教
審査請求日	平成19年7月31日 (2007. 7. 31)	(72) 発明者	バオケ, ステファン
(31) 優先権主張番号	99124727.1		ドイツ国 5 2 1 3 4 ヘルツォゲンラッ
(32) 優先日	平成11年12月13日 (1999. 12. 13)		, エリクソン アレー 1
(33) 優先権主張国	欧州特許庁 (EP)	(72) 発明者	ツムダ, マルガレーテ
			ドイツ国 5 2 1 3 4 ヘルツォゲンラッ
			, アン シュビーンプルッフ 1 4
			最終頁に続く

(54) 【発明の名称】 ネットワーク・オペレーションを実行する方法と装置

(57) 【特許請求の範囲】

【請求項 1】

アプリケーションレイヤのシステムコールを処理し、ネットワークレイヤのネットワークオペレーションを開始するモジュールを用いて高い遅延時間を有するネットワークを介したネットワークオペレーションを高速で実行する方法であって、

アプリケーションから前記モジュールにシステムコールを伝送する過程と、

前記モジュールにおいて、前記システムコールの種類によってブロッキング実行モードと非ブロッキング実行モードを区別することによりシステムコールの実行モードを決定する過程と、

非ブロッキング実行モードの場合にアプリケーションレイヤに論理値を返送しネットワークオペレーションを開始する過程とを有し、

ネットワークオペレーションがネットワークオペレーションを開始するユニットと通信する相手側に伝送され、

相手側で受信されたネットワークオペレーションが演算に変換され、それが実施され、その結果のデータがネットワークオペレーションを開始したユニットに戻され、

受信結果のデータは、非ブロッキング状態に言及し、かつ論理値を有するデータと、相手側で実行されたブロッキングシステムコールの結果のデータとを含むことを特徴とする方法。

【請求項 2】

演算の受信結果のデータ処理が前記モジュールにおいて実行される請求項 1 に記載の方

10

20

法。

【請求項 3】

ネットワークオペレーションの開始により、非ブロッキングシステムコールが、相手側で実行された前記システムコールの実際の結果のデータをコーリングアプリケーションの実行をブロッキングせずに待受ける状態に変換される請求項 1 又は 2 に記載の方法。

【請求項 4】

非ブロッキング実行モードで受信した結果のデータがバッファに保留される請求項 1 ないし 3 のいずれかに記載の方法。

【請求項 5】

論理値が論理的正の命題値又は論理的負の命題値のどちらかを有する請求項 1 ないし 4 のいずれかに記載の方法。

【請求項 6】

論理的負の結果のデータが論理的負の戻り値の形態で次のシステムコールの実行時にアプリケーションに報告される請求項 5 に記載の方法。

【請求項 7】

非ブロッキングシステムコールにおいて、保留されている以前のコールの負の結果のデータがない場合には常に、論理的正の値がアプリケーションに戻される請求項 1 ないし 6 のいずれかに記載の方法。

【請求項 8】

以前に実施した演算の結果の報告を返送することを保証するために、接続の最後のシステムコールがブロッキング状態に設定される請求項 1 ないし 7 のいずれかに記載の方法。

【請求項 9】

ブロッキングシステムコールは、相手側で実行されたシステムコールの結果のデータを待受けることにより周知の方法で達成される請求項 1 に記載の方法。

【請求項 10】

システムコールはソケットシステムコールである請求項 1 ないし 9 のいずれか 1 つに記載の方法。

【請求項 11】

ソケットシステムコールはオペレーティングシステムのプログラミングインターフェースを形成する請求項 10 に記載の方法。

【請求項 12】

モジュールはパイプラインモジュールである請求項 1 に記載の方法。

【請求項 13】

アプリケーションレイヤのシステムコールを処理し、ネットワークレイヤのネットワークオペレーションを開始するモジュールを用いて高い遅延時間を有するネットワークを介したネットワークオペレーションを高速で実行する装置であって、

アプリケーションの要求によって受信されたシステムコールの実行モードを示すデータがブロッキングシステムコールであるか非ブロッキングシステムコールであるかを決定する決定要素と、

システムコールをネットワークオペレーションに変換する変換手段と、

ネットワークオペレーションを送信する送信側と、

実施されたネットワークオペレーションの結果のデータを受信する受信側と、

受信結果のデータを格納するメモリと、

受信結果のデータを処理する処理要素とを具備し、

ネットワークオペレーションが、ネットワークオペレーションを開始するユニットと通信する相手側に伝送され、

相手側で受信されたネットワークオペレーションが演算に変換され、それが実施され、その結果のデータがネットワークオペレーションを開始したユニットに戻され、

受信結果のデータは、非ブロッキング状態に言及し、かつ論理値を有するデータと、相手側で実行されたブロッキングシステムコールの結果のデータとを含むことを特徴とする

10

20

30

40

50

する装置。

【請求項14】

一旦ネットワークオペレーションが開始されると、コーリングアプリケーションの実行をブロッキングせずに相手側で実行されたシステムコールの結果のデータを待受ける状態を達成する要素を具備する請求項13に記載の装置。

【請求項15】

非ブロッキングシステムコールの受信結果のデータを処理する手段が負の値と正の値を区別する請求項13又は14に記載の装置。

【発明の詳細な説明】

【0001】

本発明は、高い遅延時間を特徴とするネットワークを介した、高速性能のネットワークオペレーションの方法と装置に関するものである。

【0002】

高い遅延時間を有するネットワークの一例はGSM(Global System for Mobile Communication)ネットワーク（汎欧州移動体通信システム）である。GSMネットワークはチャンネル当たり9.6kビットの有効データ速度を有するデジタル移動体通信ネットワークである。データ速度が64kbit/秒にまで達するISDN（デジタル総合サービス網）といった今日の固定型ネットワークとは対照的に、GSMのデータ速度は小さい。したがって、GSM接続は不均一なネットワーク接続、つまり、少なくとも2つの異なるネットワークに広がる接続において、接続全体の絶対スループットに非常に強い影響を与える。これは特に、GSMユーザがいわゆるウェブ・サーフィン及び／又はインターネットからデータをダウンロードする目的でインターネットサービスプロバイダー（ISP）に接続をセットアップした場合に問題が生じる。前記シナリオにおいて、GSM接続は有効伝送速度に関して接続の重要な部分を構成する。

【0003】

通信相手との情報交換、この場合GSM端末とインターネットサービスプロバイダ（ISP）間の情報交換は、インターネットアプリケーション用に標準化されたプロトコルスタック、すなわち、いわゆるTCP/IP(Transmission Control Protocol/ Internet Protocol)プロトコルスタックに基づいている。前記プロトコルスタックは5つのプロトコル層から構成されている。これらの層を以後レイヤと呼ぶ。最上位レイヤのアプリケーションレイヤは、ユーザに直接使用されるアプリケーションを構成している。例えば、インターネットサーフィンはユーザが利用できるアプリケーションの1つである。アプリケーションレイヤのすぐ下にはトランスポートレイヤがあり、TCP(Transmission Control Protocol)又はUDP(Use Datagram Protocol)といった前記レイヤのタスクを履行するトランスポートプロトコルを具備する。例えば、インターネットプロトコル（IP）はネットワークレイヤを形成する。最下位の2つのレイヤである、リンクレイヤと物理レイヤは、基本ネットワークに応じて特別に規定されるので、ネットワーク志向レイヤ(Network-oriented layer)という用語で要約することができる。

【0004】

何よりも、受信データ又は送信されるべきデータを処理するのが当該レイヤのタスクである。この点において、1つのレイヤは直接隣接するレイヤとだけ通信を行う。送信用データは1つのレイヤからすぐ下のレイヤに順次送られ、そのためにデータはフォーマットされる。アプリケーションレイヤのデータは、物理接続を介して伝送可能なパケットの大きさを上回ることが多い。このため、メッセージはさらに小さなデータパケットに分割され、伝送のために連続的に配列される。データ分割はフォーマット動作の一部である。データフォーマットは各プロトコルレイヤ上で実施される。データフォーマット動作は特に、各プロトコルレイヤに固有の制御データを加えることを含む。ほとんどの場合、制御データは、ヘッダと呼ばれる形式のデータパケットの先頭部又は／及びテイルと呼ばれる形式の後尾部に加えられる。実際のデータはデータ領域に含まれる。前記制御データは各レイヤに対して特徴的に規定及び標準化される。というのは、この方法だけが、二者間の同型

10

20

30

40

50

の通信を保証することができるためである。本発明に関しては、T C Pの特定の制御データが重要となる。

【0005】

トランスポートプロトコルT C Pは二通信者間のバイトフローに信頼できる通信サービスを提供する。この点に関する信頼性とは、エラーの回避、シーケンスのメンテナンス、及びデータ損失及び重複からの保護のことを言う。伝送されるべきバイトフローはアプリケーションレイヤからT C Pレイヤに順次送られ、T C Pによりセグメントに小さく分割され、それからI Pデータグラムとして伝送されることになる。I PデータグラムはI Pプロトコルの規則に則ってフォーマットされたデータパケットを示している。データグラムは、データグラムを使用することにより達成されたデータ交換は信頼性がないという属性を有する。したがって、I Pはパケットが実際に受信者に伝送されることを保証できない。同様に、I Pデータグラムは、不正な順序で又は重複して受信側に到達する可能性がある。しかし、前記構想の範囲内で、誤った伝送を検知すること及び発生したエラーを除去することがT C Pのタスクである。階層原理により、I Pデータグラムはさらにすぐ下のレイヤに伝送される。

10

【0006】

当該指定タスクを達成するために、多くの複雑なメカニズムがT C Pに統合された。ウィンドウメカニズムといった単一メカニズム、スロースタートアルゴリズムと呼ばれる蓄積型の肯定応答の体系(the accumulative acknowledgement scheme)又は輻輳回避アルゴリズムと呼ばれる輻輳を回避するアルゴリズムについては詳細な説明を省く。前記方法の正確な説明はリチャード・スティーブン氏による「図解T C P／I P第1巻」に見ることができる。

20

【0007】

本明細書にてさらに厳密に取扱われる重要なメカニズムの1つは、T C Pとの接続セットアップに関わるものである。接続がセットアップされると、T C Pは確実に定義済みのシグナリング情報のやりとりを保証するいわゆる3方向ハンドシェイクを利用する。図2でそれを理解することができる。通信者はクライアント又はサーバと呼ばれ、クライアントは通信開始ユニットであり、サーバは通信受入れユニットである。要するに、クライアントとサーバ間の接続セットアップの際には3つのメッセージのやりとりが要求されることになる。第1メッセージを送信することで、通信者間の初期シーケンス番号が宣言される。交換されるデータパケットはシーケンス番号と呼ばれる明確な数字により識別される。このため、T C Pデータパケットのヘッダは、接続がセットアップされる際にシーケンス番号の同期をとる役目を果たす領域を有しており、それはS Y Nフラグと呼ばれ、図2ではS Y Nと表記されている。接続を受入れる準備が整うと、サーバは指定のS Y Nフラグ及びクライアントA C K (S Y N)の初期シーケンス番号に対する肯定応答(Acknowledgement)を含むメッセージで応答する。最後に、クライアントはサーバA C K (S Y N)のS Y Nメッセージに対する肯定応答を含むメッセージを送信する。3つのメッセージ交換が終了すると、接続セットアップの段階は完了する。次のステップでデータが送信され、接続セットアップにより実際の通信が完了される。接続の開始には4つのメッセージのやりとりが必要である。まず、送信方向への接続を切断する、指定のF I Nフラグ(F I N)を含むメッセージがクライアントからサーバに送られる。それによりクライアントは接続を開始する。サーバはA C K (F I N)を有するメッセージの受信を確認し、同様に切断段階に入る。このため、サーバは指定のF I Nフラグ(F I N)を有するメッセージをクライアントに送り、クライアントはA C K (F I N)で応答する。このメッセージのやりとりにより接続の開始を完了する。

30

40

【0008】

要するに、セットアップ及び接続開始に際し、最低7つのメッセージ又は3.5往復のいわゆるラウンドトリップ時間(R T T)(Round Trip Time)が要求される。R T Tは要求メッセージの送信からそれに関連する応答メッセージの受信までの時間を示している。前記時間は基本ネットワークの伝送速度に応じて相応に変化する。

50

【0009】

しかし、多くのアプリケーションは通信者間の単純なやりとりに基づいている。単純なやりとりの一例には、クライアントは要求メッセージをサーバに送り、受信者は応答メッセージにおいて要求されたメッセージを送信者に送る、いわゆるトランザクションがある。前記のようなデータのやりとりは、例えば、インターネットで見受けることができ、ワールド・ワイド・ウェブ(World Wide Web)プロトコルのトランザクション志向トラフィックパターン(the transaction-oriented traffic pattern)により特徴づけられる。インターネットでのトランザクションは主にHTTPプロトコルの使用により発生する。

【0010】

現在使用されるプロトコルのHTTP/1.0版、つまり、Berners-Lee、R. Fielding及びH. Frystyk氏により1996年5月に作成されたRFC 1945の"Hypertext Transfer Protocol - HTTP/1.0"を使用して、まず、TCP接続はテキストブロック又は画像などのWWWページの所要の各対象に対してセットアップされる。そのときだけトランザクションが実行される。つまり、送信者は要求メッセージを受信者に送り、それに対し受信者は応答メッセージにおいて要求されたメッセージを送る。その後、TCP接続は再度切断される。それが小さな画像であろうとも、全ての対象に対して系統的な接続がセットアップされる必要があり、しかもその接続が3方向ハンドシェークに基づいて少なくとも3つのセットアップメッセージと4つの開始メッセージがその度に必要になるとすると、前記のような処理はセッション全体の性能にかなりの影響を持つ。

【0011】

TCPが行わねばならないタスクを検討したことにより、小規模な通信不履行と高速データ通信とを特徴としたネットワーク状況、つまりはワイヤバウンドデータネットワークに適應するようにTCPが設計されたことが既に明らかである。しかし、こういった前提はGSMでは満たされておらず、この種のネットワークにおいてTCPの機能性が不適切であるという一因となる。小規模な伝送容量を有するネットワークに対するTCPの最適化に対処する構想が、多くの文献に示されている。以下に、解決法の一例を紹介する。

【0012】

R. Braden氏によるRFC 1644の"T/TCP - TCP Extensions for Transactions Functional Specification"は、特に接続セットアップ段階における最適化の構想を紹介している。この目的のため、特定のサーバへの接続に関する情報がバッファに保留される。したがって、完全な3方向ハンドシェークはサーバへの最初の接続セットアップの際にだけ実行され、前記接続セットアップの間に宣言される接続設定に関する情報がバッファに保留される。同じサーバへのその後のアクセスにおいて、接続セットアップ完了までの行程は、格納された情報から設定を取込むことにより、一段階飛ばされる。

【0013】

しかし、この解決法の短所は、アプリケーションに対する透過性が存在していないことにある。この構想を実現するためには、アプリケーションにおいてシステムコールを変更する必要がある。さらに高速の接続セットアップは、例えば、システムコールの"connect"と"send"を"sendto"に置換えることにより達成される。前記システムコールはいわゆるソケットプログラミングインターフェースの一部であり、そのさらに正確な機能性を以下に説明する。

【0014】

上述に従い、本発明の目的は、高い遅延時間を特徴とした伝送リンクを経由する2つの通信ユニット間に能率的なトランザクションを履行する方法と装置を提供することである。

【0015】

本発明によれば、この目的は特許請求項1と特許請求項16を教示することにより達成される。前記請求項により、トランザクションを高速で履行することにより遅延に影響されやすいアプリケーションのさらに安定性の高い動作が保証されるので、トランザクションの履行の最適化がアプリケーションの能率性に直接影響するという長所が示された。

【0016】

別の長所は、本発明の利用はアプリケーションレイヤ及びトランスポートレイヤに対して透過性を維持することが保証されるので、本発明の実現が両レイヤに依存しないことである。

【0017】

本発明のさらに有利な実施形態は、特許請求項2から15及び請求項17及び18に開示される。

【0018】

以下に実施形態及び添付図を引用して本発明をさらに詳細に説明する。

【0019】

添付図1と特許請求項1を引用して本発明を以下に説明する。

10

【0020】

図1によれば、アプリケーション1はシステムコール2を要求する。システムコール2はステップ3でパイプラインモジュールと呼ばれるモジュールに伝送される。パイプラインモジュールの第1タスクは、システムコールの実行モードを決定すること、つまり、ブロッキングシステムコールであるか非ブロッキングシステムコールであるかを決定すること（ステップ4）にある。パイプラインモジュールがブロッキングコールとしてシステムコールを認識すれば、周知の方法で処理を継続する。すなわち、ネットワークオペレーションが開始され（ステップ5）、コーリングアプリケーションには開始されたネットワークオペレーションの結果が返送される（ステップ7）までブロッキングされたままである（ステップ6）。アプリケーションのブロッキング状態を図1に示す。ブロッキング実行モードからアプリケーションに戻る矢印は無く、ネットワークオペレーションだけが開始されている（ステップ5）。非ブロッキングシステムコールの場合は、論理値が直接アプリケーションに戻される（ステップ8）。このステップにより、相手側で実際に実行されたシステムコールの結果を待受けるためにアプリケーションがブロッキングされたままでいることはなく、プログラムのステップを引き続き継続することが可能になる。「直接」("direct")という用語は、本明細書では、ネットワークオペレーションを開始するユニットで論理値を決定するために必要な時間を言う。これは前記ユニットにおいて局所的に行われるので、相手側でシステムコールを実行し、当該結果を送受信するためにネットワークを介してネットワークオペレーションを通信するのに要する時間よりも、前記時間はかなり短い。

20

30

【0021】

非ブロッキング実行モードの場合に論理値が直接戻ることによりネットワークオペレーションも同様に開始される。これは論理値のリターンの直前かその後すぐに行われる。非ブロッキングシステムコールの場合も、開始されたネットワークオペレーションの結果を同様にパイプラインモジュールにおいて待受ける（ステップ6）が、実行中のアプリケーションとは独立して行われる。結果を受信すると、前記結果を実行モード及び返送された値に応じて検討し処理する（ステップ9）。

【0022】

以下に受信結果処理の一例を説明する。

【0023】

40

まず、受信結果は実行モードについて検討される。結果が非ブロッキングシステムコールからのものである場合は、アプリケーションはその時点で実行中であり、非ブロッキング状態で結果を待受けしていることは明白である。このために受信結果は格納される。次のステップでは、当該結果はその値について、つまり、その値が負であるか又は正であるかが検討される。その結果が負であれば、アプリケーションに対するエラーメッセージが必要であることを表示するために、例えば、特定の符号を利用して示される。結果が負であるということは、接続の切断などのエラーが発生していることを示し、実行中のアプリケーションにこの事実を報告する必要がある。エラーメッセージの観点から本発明を実施する上で最も安全な方法は、エラーメッセージが出来るだけ早く発効される場合に保証される。このため、まず、現時点では報告されていない以前のコールの結果及び負の結果が格

50

納されているか否かをまず各システムコールに照会することが最適である。保留中の負の結果が存在していることが前記照会により認められれば、前記結果は論理的な負値形式で現在実行中のシステムコールの戻り値としてアプリケーションに戻される。

【0024】

戻り値に関する同様の手続がブロッキング状態にあるアプリケーションへの応答を含む結果で行われる。この場合、まず、今までに報告されなかった負の結果が格納されているか否かが同様に照会される。保留中の負の結果が存在していることが前記照会により認められれば、前記結果は論理的な負値形式で戻り値としてアプリケーションに戻されることになる。しかし、負の結果が存在しなければ、現在までの受信値はアプリケーションに返される。

10

理論的に、一つは例えば接続をセットアップするために送信されるデータパケットの形式のメッセージについて述べている。実際に、これは、例えば、ネットワークレイヤのネットワークオペレーションへのアプリケーションレイヤのプログラミング命令の変換を実行するプログラミングインターフェースを用いて実現することができる。前記プログラミング命令は、ユーザがネットワークサービスにアクセスすることを可能にするシステムコールとして実現することができる。

【0025】

ネットワークサービスは複数のサービス要素又はネットワークオペレーションによって正式に特徴づけられ、それらを介してユーザはサービスを利用することができる。こういったサービス要素は、例えば、そのサービスが特定の機能(activity)を実施することや特定の
20
の方法で相手側の機能に応じることなどを取計う。TCP/IP又はOSI (Open System Interconnection)といったプロトコルスタックの範囲内で、様々なサービス要素が存在する。前記サービス要素はサービスが特定の機能を実施するように取計う。例えば、接続
30
セットアップはサービス要素の「request」及び機能の「CONNECT」により開始される。OSI形式において、それに対応するメッセージはCONNECT.request形式を有する。前記リクエストが満たされた場合は、相手側は機能を示す表示手段を介して信号を受信する。例えば、あて先に指定された通信相手は、CONNECT.requestの次にCONNECT.indicatorを受信する。これは接続セットアップが要求されていることを意味するものである。相手側は応答メッセージ、この場合は、CONNECT.responseを使用し、それにより接続要求の受入れ又は拒絶を望んでいることを伝達する。最初のCONNECT.requestを発効した通信者は、CONNECT.confirmと呼ばれる応答確認により、相手側の準備が完了していることを確認する。既述の例はCONNECTにより開始された接続セットアップに言及している。類似のサービス要素はデータ送信(SEND)などの他の操作にも規定されるが、ここでは詳細な説明を省く。しかし、既述の例は、ネットワークオペレーションの規格が非常に抽象的であることを示している。例えば、オペレーティングシステムUnix（登録商標）4.3 BSDの場合、ソケット
40
インターフェース(socket interface)と呼ばれる、ネットワークサービスへのシステム独立型アクセス用の統一プログラミングインターフェースが導入される。これによりサービス要素のさらに明確な規格が保証される。それによるメッセージの開始は、アプリケーションレイヤ上のプログラミング命令の規格に具体的に言及する。そのとき、前記ソケットインターフェースは、以後システムコール又はコールと呼ぶプログラミング命令のサービス要素への変換を実行する。以下に、本発明を説明する目的で、一例として前記プログラミングインターフェースを利用する。

【0026】

ソケットプログラミングインターフェースは、それぞれがいわゆるソケットに言及している一群のシステムコールを含む。当該構想の観点から、ソケットは通信の終端点を構成し、一方はネットワーク側に、他方はアプリケーション側に接続される。最も重要なシステムコールについて以下でさらに詳述する。新しいソケットがシステムコールソケットと共に発生し、それによりアドレス形式、ソケット型及びプロトコルがパラメータとして特定される。TCP接続に対しては、例えば、インターネットアドレス形式、接続志向ソケット型及びTCP/IPプロトコルが選択される。システムコールコネクトを利用して、ア

50

アプリケーションはネットワークの遠隔ソケットへの接続を要求する。ローカルソケットと遠隔ソケットのアドレスはここではパラメータとして含まれる。各接続は一对のソケットにより明確に特徴づけられる。実際のデータ送信はcall sendにより行われ、受信データの読み込みはcall recvにより行われる。本明細書で述べた例に加えてまだ多くのシステムコールが存在するが、説明を明瞭にするためにさらに詳細には取扱わない。

システムコールはプログラムインターフェースによりネットワークオペレーションに変換される。これは、例えば、call connectの履行により3方向ハンドシェークが開始されることを意味する。ネットワークオペレーションの実行に対し、ユーザには直接的な影響力を持たない。しかし、ユーザはシステムコールによりネットワークオペレーションへのアクセスを制御することができる。現在のアプリケーションの変更を回避するために、本発明の目的は、システムコールとネットワークオペレーションの間の追加レイヤを統合し、それによりアプリケーションの実施を変更することなくどちらの操作についても透過性のある変換を可能にする。以下、パイプラインモジュールと呼ぶ前記追加レイヤは、アプリケーションから受信したシステムコールを解釈し、相応に処理するタスクを有する。

【0027】

しかし、パイプラインモジュールの主なタスクはシステムコールの実行モードの決定、つまり、システムコールの結果を直接アプリケーションに渡すことができるか、或いは、アプリケーションがサーバにより実行される実際の演算値を待受ける必要があるかの決断にある。第1番目の場合は非ブロッキングシステムコールに関係し、アプリケーションが実行した演算の実際の値を求める第2番目の場合はブロッキングシステムコールに関係する。これは、パイプラインモジュールが2つの型のシステムコールを区別する必要があることを意味し、それは以下の方法で達成することができる。戻り値が論理的な命題値を有する場合、つまり、実行した演算が成功した場合に真である1が返され、実行した演算が失敗した場合に誤である0が返されれば、アプリケーションは非ブロッキングコールに関わっていることを認識し、これにより戻り値を直接アプリケーションに渡すことが可能であるということを意味する。前記値の受信により、アプリケーションは続いて次の演算に進む。係属中の演算の進行状況、つまり、サーバに送信されたがまだその実値を受信していない演算は、パイプラインモジュールのバッファに保留される。前記進行状況を格納することは、パイプラインモジュールが受信した実値をまだ係属状態にある演算に割当ての必要である。

【0028】

データ及び非論理値がシステムコールの戻り値として待受けされることを特徴とするブロッキングシステムコールの場合、パイプラインモジュールは値を直接アプリケーションには戻さず、実際のデータが受信されるまで待受けされる。この時間中はアプリケーションはブロッキングされたままである。データ以外の論理的な戻り値を返送しないそれらシステムコールを除いて、当該システムは接続の最後のシステムコールも加えてブロッキング状態に設定されることを確認する必要がある。というのは、このようにしてのみ既に実行された演算結果の返送が保証されるからである。ソケットプログラミングインターフェースの記号論によれば、その使用の完了時にソケットを閉鎖することが要求され、それはシステムコールのclose又はshutdown、つまりブロッキングモードで達成されるシステムコールを使用することにより実行される。

【0029】

本発明により保証された最適化をする上での問題点は、非ブロッキングソケットコールの実際の履行は待受けされず、非ブロッキングソケットコールにおける戻り値の意味が変化することにある。このため、パイプラインモジュールが、アプリケーションへの応答メッセージを有する実行済みのシステムコール全体の制御を保証することが必要である。例えば、コールコネクトが失敗すれば、アドレス指定済みのIPアドレスにはネットワークの停止により接続することができないので、パイプラインモジュールは、適切な戻り値を受信した後で、且つ該アプリケーションが首尾よく完了されたものとして前記コールを判断した後に、やっとその応答エラーを認識する。しかし、戻り値は受信時にパイプラインモ

ジュールに格納され、次のシステムコールでアプリケーションに報告される。好ましい実施例ではその通信が次のシステムコールで直接行われ、それによりできるだけ素早いエラー報告が行われることが保証される。係属中のエラー報告の照会にかかる計算機の照会時間はGSMの伝送時間よりも短い。したがって、パイプラインモジュールに存在する戻り値の照会はさほど労せず達成することができる。

【0030】

次のシステムコールで発生したエラーを報告することも可能であり、これはブロッキング状態で実行される。

【0031】

こういった実施形態はその実施にかかっている。しかし、どちらの場合でも、最後のシステムコールをブロッキング状態で実行することが必要である。

10

【0032】

TCPによる接続においても、何時でもエラーが発生する可能性に対処する準備が整っているので、「遅延した」エラー報告は周知のアプリケーションにおいて何ら問題を伴わない。

【0033】

本発明の実施例を実施形態及び図3aと図3bを引用して以下でさらに詳細に説明する。図の左側はアプリケーションにより開始された実行済みのシステムコールを図示しており、図の右側はそれに対応するシステムコールの実行を開始するネットワークオペレーションを図示している。

20

【0034】

図3aによる実施形態は、従来技術のRFC 1945、プロトコルHTTP 1.0で実施される接続セットアップを図示している。WWWページに接続するために使用されるこのプロトコルにおいて、各対象に対して独立した接続がセットアップされる。接続は第1のコールconnectにより図3aに従って達成される。結果として、例えば1つの対象などのデータは、第1及び第2コールのsend及びrecvにより要求され、伝送される。その後の接続は別のコールcloseにより開始される。

【0035】

本発明に従うと、変換済みのシステムコールは伝送手段を介して直接伝送されることはなく、まずパイプラインモジュールに渡される。パイプラインモジュールはシステムコールを処理する責任を負う。図3bによれば、第1システムコールconnectは第1ステップで開始され、パイプラインモジュールにより非ブロッキングシステムコールとして解釈される。これは、一方ではパイプラインモジュールがその対応するネットワークオペレーションをサーバに送り、同時に、サーバにおいて実施された実際の演算結果を待受けせずにシステムコールに照らした結果をアプリケーションに直接送るといった方法で図示されている。前記結果の直接的な送信は直接戻り矢印により表されている。

30

【0036】

この方法は、アプリケーションを全RTT(Round trip Time)の間継続されるブロッキング状態に設定する必要が無く、結果の受信により次のステップが直接的に継続して実行されるという効果を有する。図3bによれば、別のシステムコールsendが第1システムコールconnectと同様に対処される。次のシステムコールはrecvである。前記システムコールもまずはパイプラインモジュールに送信され、データは戻り値として求められるため、そこでブロッキングコールとして識別される。このため、パイプラインモジュールは結果をアプリケーションに直接戻すことはできず、サーバで実際に実施されたシステムコールの結果を待受けする。計算機の動作時間が特にGSM接続などネットワーク接続の伝送時間と対比して短いという事実により、ネットワークオペレーションは連続して直接的に送信される。この理由により、システムコールconnect及びシステムコールsendの結果が待受けされないため、本事例では2つのRTTが節約される。前記結果はシステムコールrecvの結果と統合して返送される。ネットワークオペレーションの連続した直接的な送信は、送信順序に従って対応したネットワークオペレーションを含む幅広い矢印により図3bに

40

50

示される。

【0037】

T/TCP以外のトランスポートプロトコルが使用される場合にその機能性も保証されるので、本発明の特徴はトランスポートプラットフォームから独立していることである。T/TCPの場合、システムコールconnect及びシステムコールsendのどちらもいわゆるsendtoシステムコールとして解釈される。しかし、前記他のプロトコルを使用しても、所要のRTTの全体的な節約という点から見て何ら変わらない。前記RTTの回数は図例において3回から1回に減少する。

【0038】

以下に、本発明は特許請求項16（添付図なし）に準じてさらに詳細に説明される。長い遅延時間を有するネットワークを介したネットワークオペレーションを高速で実行するための本発明による装置においては、システムコールの実行モードは決定要素により決定される。この点に関し、ブロッキング状態と非ブロッキング状態を区別する。システムコールは変換要素によりネットワークオペレーションに変換される。送信側はネットワークオペレーションを通信ユニットに送信する。通信ユニットで実行されたネットワークオペレーションの結果は、受信機によりネットワークオペレーションを開始するユニットに受信される。受信結果はメモリに格納され、それに含まれる値に従って処理要素により処理される。特許請求項18によれば、負の値と正の値が区別される。

【0039】

非ブロッキングシステムコールの場合、対応するネットワークオペレーションが一旦開始されると、アプリケーションの実行をブロッキングせずに相手側で実行されたシステムコールの結果が待受けされる。これは、このような状態を実現する要素を用いて達成される。

【0040】

本発明の実施例は、追加の上記パイプラインモジュールがソケットプログラミングインターフェースの下部に実装されることに基づいている。このため、本発明の実施には、相違するオペレーティングシステムの相違する設定を考慮することによりオペレーティングシステムのレイヤ上で達成される必要がある。しかし、ソケットプログラミングインターフェースの基本的な概念はプロトコルからの独立性にあり、そのため、ソケットプログラミングインターフェースにプロトコルモジュールを追加する目的で、内部ソフトウェアインターフェースがほとんどのオペレーティングシステムに実装されており、これにより本発明が容易に実現できる。

【0041】

本発明の実施は、オペレーティングシステムWindows（登録商標）を一例として図4に基づいて以下でさらに詳細に説明される。

【0042】

図4は関連するレイヤを含むプロトコルスタックを示している。直接隣接し合うアプリケーション、Winsock2、パイプラインモジュール及びトランスポートプロトコルは相互に通信し合っており、矢印で表記されている。当該隣接レイヤ間の通信は規定のソフトウェアインターフェースを用いて行われる。例えば、オペレーティングシステムWindows（登録商標）においてWinsock 2 APIと呼ばれるソケットプログラミングインターフェースを用いて、アプリケーションとWinsock 2間の通信が行われる。1997年8月のマイクロソフト改訂版2.2.2の"Windows Sockets 2 Service Provider Interface"に詳細に説明される、ソフトウェアインターフェースWinsock SPI (Service Provider Interface)が、Winsock 2レイヤとそのすぐ下で実施されるプロトコル（本事例の場合はパイプラインモジュール）間の通信に使用される。

【図面の簡単な説明】

【図1】 図1は本発明による方法のフローチャートを示している。

【図2】 図2は3方向ハンドシェークの概要図を示している。

【図3】 図3aは従来技術による接続セットアップを図示しており、図3bは本発明に

10

20

30

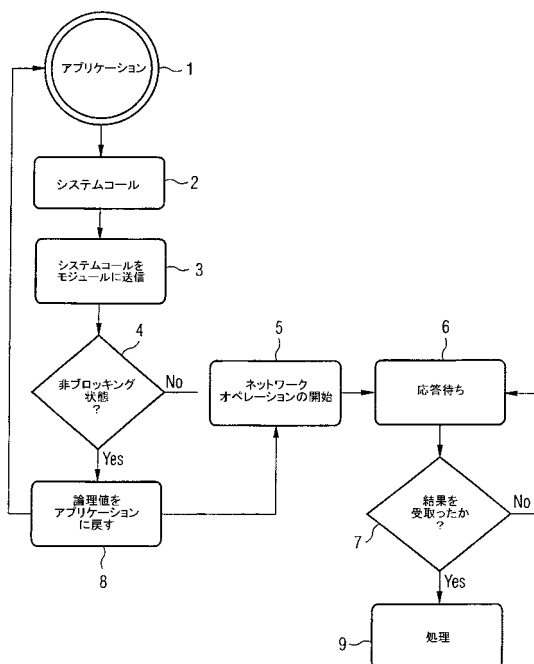
40

50

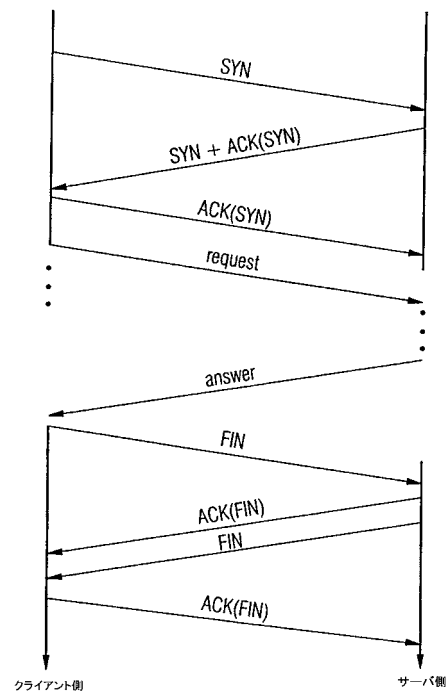
による接続セットアップを図示している。

【図 4】 図 4 は 1 実施例を示している。

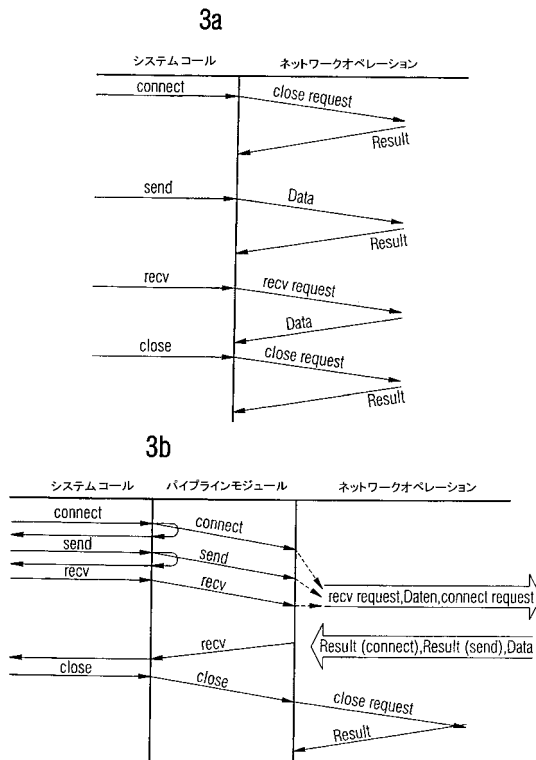
【図 1】



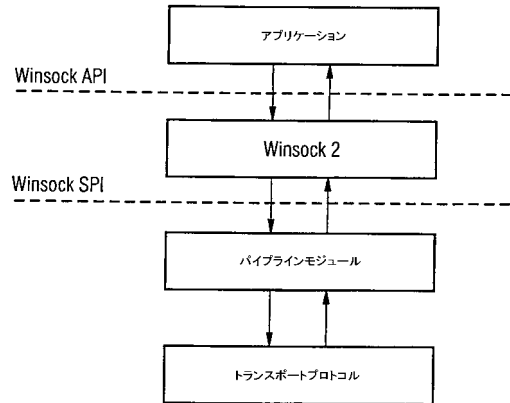
【図 2】



【図 3】



【図 4】



フロントページの続き

審査官 寺谷 大亮

(56)参考文献 特開平11-004259 (JP, A)

特開平11-045220 (JP, A)

特開平10-334015 (JP, A)

新井正広, Delphiの神託―第28回, Inside Windows 6月号, 日本, ソフトバンクパブリッシング
株式会社, 1999年 6月 1日, 第5巻, 第6号, p.96~103

(58)調査した分野(Int.Cl., DB名)

G06F 13/00

H04L 29/10