(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4123712号
(P4123712)
(45)発行日　平成20年7月23日(2008.7.23)　(24)登録日　平成20年5月16日(2008.5.16)

(51)Int.Cl.　F I
G06F 9/54 (2006.01)　G06F 9/46 480Z

請求項の数 4　(全 13 頁)

(21)出願番号　特願2000-364542(P2000-364542)
(22)出願日　平成12年11月27日(2000.11.27)
(65)公開番号　特開2002-163122(P2002-163122A)
(43)公開日　平成14年6月7日(2002.6.7)
審査請求日　平成18年3月3日(2006.3.3)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100100310
弁理士　井上　学
(72)発明者　亀山　伸
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内
(72)発明者　垂井　俊明
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

最終頁に続く

(54)【発明の名称】通信処理方法ならびに通信処理プログラムが記録される記録媒体

(57)【特許請求の範囲】
【請求項１】
仮想インターフェースを生成するＶＩＡモジュールと、ＴＣＰ／ＩＰソケットエミュレータモジュールと、サーバアプリケーションないしクライアントアプリケーションとが各々実装され、物理通信回線で接続された第１の情報処理装置および第２の情報処理装置上で実行される通信処理方法であって、
前記第１の情報処理装置で動作するサーバアプリケーションと前記第２の情報処理装置上で動作する前記クライアントアプリケーションとが互いに通信を行いながら、当該サーバアプリケーションおよびクライアントアプリケーションを動作させるためのプロセスを実行する通信処理方法において、
前記サーバアプリケーション上でＦＯＲＫ処理を実行することにより当該サーバアプリケーション側プロセスの子プロセスを生成する際に、
前記サーバアプリケーションは前記クライアントアプリケーションに対してデータ転送中断要求を送信し、
当該クライアントアプリケーションは前記サーバアプリケーションに対してデータ転送中断の完了報告を送信し、
前記データ転送中断要求の送信から前記完了報告の受信までに前記クライアントアプリケーションから送信されたデータを、前記ＦＯＲＫ処理により生成された子プロセスに割り当てられたアドレス空間にコピーすることにより、前記サーバアプリケーション側プロセスの処理を前記子プロセスに引き継ぐことを特徴とする通信処理方法。

【請求項２】
仮想インターフェースを生成するＶＩＡモジュールと、ＴＣＰ／ＩＰソケットエミュレータモジュールと、第１のアプリケーションおよび第２のアプリケーションとが実装された情報処理装置上で実行されるプロセス間通信方法であって、
前記第１のアプリケーションと第２のアプリケーションとが互いに通信を行いながら、当該第１のアプリケーションまたは第２のアプリケーションを動作させるためのプロセスを実行する通信処理方法において、
前記第１のアプリケーション上でＦＯＲＫ処理を実行することにより当該第１のアプリケーション側のプロセスの子プロセスを生成する際に、
前記第１のアプリケーションは第２のアプリケーションに対してデータ転送中断要求を送信し、
当該第２のアプリケーションは前記第１のアプリケーションに対してデータ転送中断の完了報告を送信し、
前記データ転送中断要求の送信から前記完了報告の受信までに前記第２のアプリケーションから送信されたデータを、前記ＦＯＲＫ処理により生成された子プロセスに割り当てられたアドレス空間にコピーすることにより、前記第１のアプリケーション側のプロセスの処理を前記子プロセスに引き継ぐことを特徴とするプロセス間通信方法。
【請求項３】
請求項１に記載の通信処理方法において、
前記サーバアプリケーションは、前記クライアントアプリケーションに対し、前記コピーの完了後に前記サーバアプリケーション側のプロセスに割り当てられた仮想インターフェースの張り替え要求を送信することを特徴とする通信処理方法。
【請求項４】
請求項２に記載のプロセス間通信方法において、
前記第１のアプリケーションは、前記第２のアプリケーションに対し、前記コピーの完了後に前記第１のアプリケーション側のプロセスに割り当てられた仮想インターフェースの張り替え要求を送信することを特徴とするプロセス間通信方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
この発明は、ＴＣＰ／ＩＰのソケット通信におけるｆｏｒｋをエミュレートする通信処理方法に係り、特にＯＳの介在なしにユーザ空間転送を行う情報処理システムにおいて、ｆｏｒｋで生成された子プロセスが親プロセスとクライアント間のネットワーク接続を引き継ぐ通信処理方法に関する。
【０００２】
【従来の技術】
サーバ・クライアントモデルの分散型情報処理システムにおいて、ＴＣＰ／ＩＰによるソケット通信はごく一般的であり、莫大なソフトウェア資産が存在する。図２にサーバ・クライアントモデルの情報処理システムの例を示す。この例ではホスト２１０とホスト２６０が通信回線３０で結合しており、サーバアプリケーション２１１とクライアントアプリケーション２６１がソケット通信を行う。
【０００３】
従来のソケット通信では、クライアントアプリケーション２６１のプロセス２６２からの要求をサーバアプリケーション２１１の親プロセス２１２が受けた場合、親プロセス２１２がＵＮＩＸのｆｏｒｋ機能によって子プロセス２１３を生成し、子プロセス２１３がプロセス２６２からの要求を処理する。しかしながらソケット通信では通信データを一旦ＯＳ空間内のバッファ（バッファ２３１やバッファ２８１）でバッファリングするために、スループットの向上に限界があった。
【０００４】
これに対して近年、ユーザ空間で直接データ転送を行いＯＳ空間へのコピーを不要とする

ＶＩＡ（Ｖｉｒｔｕａｌ Ｉｎｔｅｒｆａｃｅ Ａｒｃｈｉｔｅｃｔｕｒｅ）等の次世代高速ＩＯ方式が提案されている。公知の技術としては例えばＳｔｅｖｅｎ Ｈ. Ｒｏｄｒｉｇｕｅｓ他による“Ｈｉｇｈ－Ｐｅｒｆｏｒｍａｎｃｅ Ｌｏｃａｌ Ａｒｅａ Ｃｏｍｍｕｎｉｃａｔｉｏｎ Ｗｉｔｈ Ｆａｓｔ Ｓｏｃｋｅｔ”（Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ＵＳＥＮＩＸ、１９９７）がある。該技術によると高速ネットワークでサーバとクライアント間を接続することで性能向上を実現することが可能になり、かつソケット通信の一部の機能を該高速ネットワークのＡＰＩでエミュレートすることでこれまでのソフトウェア資産を有効活用することが可能になる。このエミュレートに関してはＵＮＩＸのリネーム機能によって本来の関数を新しい関数で置き換えて実現できることは周知の技術である。

【０００５】

【発明が解決しようとする課題】

ところで図２に示すように、ソケット通信における端点の一方はサーバ側（ホスト２１０）のプロセスが保持し（ソケット２２１）、他方はクライアント側（ホスト２６０）のプロセスが保持する（ソケット２７１）。

【０００６】

またサーバ側ではｆｏｒｋによって子プロセス２１３を生成する際に、親プロセスが保持するソケットの属性も子プロセスにコピーするため、親プロセスと子プロセスで同一のソケット２２１を共有して利用することが可能である。すなわちｆｏｒｋによって生成された子プロセス２１３はそのままでクライアントとの通信が可能である。

【０００７】

しかしながら図３に示すように、ＶＩＡではプロセス間の通信回線の端点（ＶＩｒｔｕａｌ Ｉｎｔｅｒｆａｃｅ：以後ＶＩと記す）はそれぞれのプロセスのローカルな資源であり、ｆｏｒｋで子プロセス３１３を生成しても親プロセス３１２のＶＩ３３１は共有できないという制限がある。

【０００８】

そのためｆｏｒｋによって子プロセスを生成してもクライアントとのＶＩの接続が確立できないのでクライアントとの通信ができないという問題が生じる。したがって、ＶＩＡでソケット通信のｆｏｒｋをエミュレートするためには生成された子プロセス３１３とホスト３６０のクライアントアプリケーション３６１との間でＶＩの接続を確立することが課題になる。

【０００９】

またアプリケーションによっては子プロセス３１３とプロセス３６２間でＶＩの接続が確立する前に親プロセス３１２とプロセス３６２間で通信が発生する場合もあり得る。この場合には親プロセス３１２とプロセス３６２間で発生した通信を確実に子プロセス３１３とプロセス３６２間で引き継ぐことが課題になる。

【００１０】

したがって、本発明の一つの目的は、ＦＯＲＫをエミュレートするようにサーバの親プロセスとクライアントのプロセス間のＶＩ接続をサーバの子プロセスとクライアントのプロセス間に張り替えて、子プロセスが親プロセスの通信を継続する方法を提供することにある。

【００１１】

【課題を解決するための手段】

本発明の一つの態様によれば、第１の情報処理装置上で実行されるプロセスと第２の情報処理装置上で実行されるプロセスが互いにユーザ空間直接転送により通信を行う機能を有する情報処理システムにおいて、前記第１の情報処理装置上の第１のプロセスと前記第２の情報処理装置上の第２のプロセスとの間の第１の接続による通信を、前記第１の情報処理装置上の第３のプロセスと前記第２のプロセスとの間の第２の接続によって引き継いで通信を継続する。その特徴は、前記第１のプロセスが第２のプロセスに対してデータ転送の中断要求を発し、第２プロセスはそれに応じてデータ転送を中断すると中断完了報告

を返し、第１プロセスはその報告を受けると、それまで第２プロセスから受信していた既受信データが有る場合にはそれを新たに生成した第３プロセスのコピーし、その後に第３プロセスと第２プロセスの間の第２の接続を確立するよう張替え要求を発し、もって前記第２の接続が前記第１の接続による通信を引き継いで通信を継続する通信処理方法にある。

【００１４】

また本発明の別の態様は、上記に加え、前記第２の機能は前記第２の接続が確立されたことを前記第２または前記第３のいずれか、または両方のプロセスが前記第１のプロセスに報告する機能を具備することを特徴とする。

【００２０】

また本発明の別の態様は、ソケット通信の機能をエミュレートするようにプログラムされたエミュレーションライブラリで実現され、前記第１の情報処理装置と前記第２の情報処理装置でそれぞれ前記エミュレーションライブラリを実行することにより、前記第１の情報処理装置と前記第２の情報処理装置のそれぞれで実行するソケット通信のためのユーザプログラムを何ら変更することなく通信が可能であることを特徴とする。

【００２１】

【発明の実施の形態】

図１に本発明の実施形態のサーバ・クライアントシステムを示す。

【００２２】

図１においてホスト１１０のメモリ（図示しない）上にはサーバアプリケーション１１１、ＴＣＰ／ＩＰソケットエミュレータモジュール１２０、ＶＩＡモジュール１３０が格納されている。またホスト１６０のメモリ（図示しない）上には同様にクライアントアプリケーション１６１、ＴＣＰ／ＩＰソケットエミュレータモジュール１７０、ＶＩＡモジュール１８０が格納されている。

【００２３】

ホスト１１０とホスト１６０はネットワーク１で接続されており、サーバアプリケーション１１１とクライアントアプリケーション１６１は通信回線（１０、２０）を介してパケットの送受で通信を行う。

【００２４】

ＴＣＰ／ＩＰソケットエミュレータモジュール１２０および１７０は、前記公知例と同様にＶＩＡのＡＰＩによってＴＣＰ／ＩＰのソケット通信の機能をエミュレートした関数群を含み、特に図１０に示した関数群に関しては、関数本来の機能に加え、本発明を実現するための機能を追加して該関数をエミュレートしている。

【００２５】

ＶＩＡモジュール１３０および１８０はＶＩＡドライバ（図示せず）を含んでいる。またＶＩＡによる通信回線の端点であるＶＩ（１３１、１３２、１８１、１８２）を構成する。

【００２６】

サーバアプリケーション１１１において親プロセス１１２はＦＯＲＫ（）をエミュレートした図１０の参照符号Ｆ１示すＥＭＵ＿ＦＯＲＫ（）Ｆ１によって子プロセス１１５を生成する。ＥＭＵ＿ＦＯＲＫ（）Ｆ１では子プロセスを生成する機能に、親プロセス１１２および子プロセス１１５がプロセス間通信を行うために共有メモリ１１７を生成し、共有メモリ１１７に張替え要求フラグＰ１１８、張替え要求フラグＣ１１９およびコピー発生フラグ１４０を生成する機能が追加されている。また親プロセス１１２内にデータ既受信フラグ１１３、張替え完了フラグ１２３および張替え要求監視スレッド１１４を生成する。さらに、子プロセス１１５内に張替え要求監視スレッド１１６およびコピー要求監視スレッド１４１を生成する。張替え要求フラグＰ１１８は親プロセス１１２によって書き込まれ、その書き込みを張替え要求監視スレッド１１６が監視する。コピー発生フラグ１４０は親プロセス１１２によって書き込まれ、その書き込みをコピー要求監視スレッド１４１が監視する。張替え要求フラグＣ１１９は子プロセス１１５によって書き込まれ、その

書き込みを張替え要求監視スレッド１１４が監視する。
【００２７】
クライアントアプリケーション１６１においては、プロセス１６２がＣＯＮＮＥＣＴ（）をエミュレートした図１０の参照符号Ｆ２に示すＥＭＵ＿ＣＯＮＮＥＣＴ（）でもってサーバアプリケーション１１１に対して通信回線の確立を要求する。ＥＭＵ＿ＣＯＮＮＥＣＴ()Ｆ２には、接続が許可されるとプロセス１６２内に中断要求フラグ１６３と中断要求監視スレッド１６４を生成する機能が追加してある。中断要求フラグ１６３はサーバアプリケーション１１１から送られてくるパケットによって書き込まれ、その書き込みを中断要求監視スレッド１６４が監視する。
【００２８】
図４、図５および図６は、本実施態様における親プロセス１１２、子プロセス１１５、およびプロセス１６２の処理手順をそれぞれ示すフローチャートである。以降、図を用いてＶＩの張替え処理の詳細を説明する。
【００２９】
まず、図４に示すフローチャートによって親プロセス１１２の処理の流れ説明する。
【００３０】
親プロセス１１２は、ステップＳ２ではクライアントアプリケーション１６１上のプロセス１６２からの接続要求を聴取する。次に、ステップＳ３ではプロセス１６２からの接続要求に対して接続許可を出す。次に、ステップＳ４ではＥＭＵ＿ＦＯＲＫ（）Ｆ１によって子プロセスを生成する。ステップＳ５ではＶＩの張替えが既に行われているかを張替え完了フラグ１２３を見てチェックする。張替え完了フラグ１２３はステップＳ１３において書き込まれる。
【００３１】
既に張替えが行われている場合（張替え完了フラグ１２３が１の場合）はステップＳ１４に進む。張替えが行われていない場合（張替え完了フラグ１２３が０の場合）はステップＳ６へ進む。
【００３２】
ステップＳ６では親プロセス１１２は、自プロセスでのＶＩ張替え要求トリガの発生の有無をチェックする。子プロセスはステップＳ４で生成されているが、その時点（ステップＳ４）ではまだＶＩの張替えは行わない。なぜならばＶＩの張替えは多大なオーバヘッドを伴う高価な処理であり、また子プロセスを生成しても必ずしも該子プロセスがクライアントと通信を行うわけではないからである。そのため図７に示すように、実際に子プロセス１１５とプロセス１６２の通信を発生させるようなイベントをＶＩ張替え処理開始のトリガとする。
【００３３】
図７において、トリガＴ１は親プロセス１１２による通信回線１０のＣＬＯＳＥである。該トリガは自発的なトリガである。通信回線１０のＣＬＯＳＥ後はプロセス１６２の通信相手は子プロセス１１５になる。
【００３４】
トリガＴ２は親プロセス１１２の消滅である。該トリガは自発的な場合と突発的な場合がある。消滅後は子プロセス１１５が通信可能の状態でなければならない。
【００３５】
トリガＴ３は子プロセス１１５の受信動作（ＲＥＣＶ）の発生である。該トリガは自発的なトリガである。子プロセス１１５はプロセス１６２からの通信を期待しているので通信可能の状態でなければならない。トリガＴ４は子プロセス１１５の送信動作（ＳＥＮＤ）の発生である。該トリガは自発的なトリガである。子プロセス１１５がプロセス１６２へデータを送信するため通信可能の状態でなければならない。
【００３６】
ステップＳ６では、親プロセス１１２は該トリガの発生がなければステップＳ１４に進んで、サーバプログラムを継続して実行する。ステップＳ６において該トリガの発生がある

場合は親プロセス１１２が実行している通常の処理を一旦休止し、ＶＩ張替え処理のルーチンであるステップＳ７に進む。

【００３７】

ステップＳ７では親プロセス１１２はプロセス１６２に対してデータ転送の中断を要求する。その後、ステップＳ８でプロセス１６２からのデータ転送中断完了報告を待つ。データ転送中断完了報告があると親プロセスはステップＳ９に進み、ＶＩ張替え前に親プロセス１１２とプロセス１６２の間で通信が発生し親プロセス１１２が既にデータを受け取っているか否かをチェックする。チェックはデータ既受信フラグ１１３を見て行う。データ既受信フラグ１１３はステップＳ１６で書き込まれる。既にデータを受け取っている場合（データ既受信フラグ１１３が１の場合）はステップＳ１０に進む。データを受け取っていない場合（データ既受信フラグ１１３が０の場合）はステップＳ１１に進む。

【００３８】

ステップＳ１０では、親プロセス１１２はコピー発生フラグ１４０を立てて子プロセス１１５にデータコピーの発生を通知する。続いて親プロセス１１２と子プロセス１１５が協調して親プロセス１１２のアドレス空間を子プロセス１１５にコピーし、親プロセス１１２が既に受信しているデータを子プロセス１１５に漏れなく引き継がせる。

【００３９】

ステップＳ１１では子プロセス１１５およびプロセス１６２に対して互いにＶＩを張替えるよう要求する。親プロセス１１２と子プロセス１１５は共有メモリ１１７を介して通信し、張替え要求フラグＰ１１８を立てることで子プロセス１１５に対してＶＩの張替えを要求する。

【００４０】

親プロセス１１２とプロセス１６２は通信回線１０を介して通信し、親プロセス１１２がプロセス１６２に対してＶＩ張替え要求用パケットを送信し張替え要求フラグ１６３を立てることでＶＩの張替えを要求する。ステップＳ１２ではプロセス１６２からの張替え完了報告を待つ。

【００４１】

ステップＳ１３では張替え完了フラグ１２３を立てる。張替え完了フラグ１２３を立てることにより、以後張替え要求トリガを発行させるイベントが発生した場合でも、二重に張替え要求が発行されること防ぐ。ステップＳ１４ではＶＩ張替え処理のルーチンから復帰する場合は、休止していた親プロセス１１２の処理を再開し、そうでなければ通常の処理の実行する。

【００４２】

ステップＳ１５では親プロセス１１２の通常の処理において、プロセス１６２からのデータを受信したか否かをチェックする。この処理はＲＥＣＶ（）をエミュレートした関数ＥＭＵ＿ＲＥＤＶ（）において実行される。ステップＳ１５においてデータ受信が発生した場合はステップＳ１６に進む。データ受信が発生していない場合はステップＳ１７へ進む。

【００４３】

ステップＳ１６ではデータ既受信フラグ１１３を立てる。データ既受信フラグ１１３を立てることにより、子プロセス１１５にコピーすべきデータの受信が発生したことを親プロセス１１２は知る。

【００４４】

ステップＳ１７では親プロセス１１２の通常の処理において該プロセスが終了するか否かをチェックする。終了する場合はステップＳ１８へ進み該プロセスは終了する。終了しない場合は再度ステップＳ５へ進み、これまでのステップを繰り返す。

【００４５】

以上がＶＩ張替えの際の親プロセス１１２の動作である。

【００４６】

次に、図５に示すフローチャートによって子プロセス１１５の処理の流れを説明する。

【００４７】
ステップＳ２１ではＥＭＵ＿ＦＯＲＫ（）Ｆ１によって子プロセス１１５が生成されて処理が開始される。
【００４８】
ステップＳ２２では子プロセス１１５は自身におけるＶＩ張替え要求トリガの発生の有無をチェックする。ステップＳ２２においてトリガＴ３またはトリガＴ４が発生していればステップＳ２３に進む。該トリガが発生していなければステップＳ２６に進む。
【００４９】
ステップＳ２３では子プロセス１１５は張替え要求フラグＣ１１９を立て、親プロセス１１２に対して子プロセス１１５において張替え要求トリガが発生したことを報告する。親プロセス１１２は張替え要求フラグＣ１１９をスレッド１１４で監視して該トリガの発生を知る。
【００５０】
ステップＳ２４では子プロセス１１５は親プロセス１１２からのデータコピー要求の有無をチェックする。該チェックはコピー要求監視スレッド１４１によってコピー発生フラグ１４０を監視することで行われる。ステップＳ２４においてデータコピー要求がある場合（コピー発生フラグ１４０が１の場合）は、子プロセス１１５が実行している通常の処理を一旦休止し、ステップＳ２５に進む。ステップＳ２４においてデータコピー要求が無い場合（コピー発生フラグ１４０が０の場合）はステップＳ２６に進む。
【００５１】
ステップＳ２５では子プロセス１１５は親プロセス１１２のアドレス空間が子プロセス１１５へコピーされる。
【００５２】
ステップＳ２６では子プロセス１１５は親プロセス１１２からのＶＩ張替え要求の有無をチェックする。該チェックはスレッド１１６によって張替え要求フラグＰ１１８を監視することで行われる。ステップＳ２６において張替え要求がある場合（張替え要求フラグＰ１１８が１の場合）は、ＶＩ張替え処理のルーチンであるステップＳ２５に進む。ステップＳ２６において張替え要求が無い場合（張替え要求フラグＰ１１８が０の場合）はステップＳ２９に進む。
【００５３】
ステップＳ２７では子プロセス１１５はクライアントアプリケーション１６１上のプロセス１６２からの接続要求を聴取する。
【００５４】
ステップＳ２８では子プロセス１１５はプロセス１６２からの接続要求に対して接続許可を出す。
【００５５】
ステップＳ２９ではＶＩ張替え処理のルーチンから復帰する場合は、休止していた子プロセス１１５の処理を再開し、そうでなければ通常の処理の実行する。
【００５６】
ステップＳ３０では子プロセス１１５の通常の処理において該プロセスが終了するか否かをチェックする。終了する場合はステップＳ３１へ進み該プロセスは終了する。終了しない場合は再度ステップＳ２２へ進み、これまでのステップを繰り返す。
【００５７】
以上がＶＩ張替えの際の子プロセス１１５の動作である。
【００５８】
最後に、図６に示すフローチャートでプロセス１６２の処理の流れを説明する。
【００５９】
ステップＳ４１で開始されたプロセス１６２はステップＳ４２において親プロセス１１２に対してＶＩの接続要求を行う。
【００６０】

プロセス１６２はステップＳ４３では接続が許可されるのを待つ。
【００６１】
ステップＳ４４では親プロセス１１２からのデータ転送中断要求の有無をチェックする。該チェックは中断要求監視スレッド１６４によって中断要求フラグ１６３を監視することで行われる。ステップＳ４４において中断要求がある場合（中断要求フラグ１６３が１の場合）は、プロセス１６２が実行している通常の処理を一旦休止し、ステップＳ４５へ進む。ステップＳ４４において中断要求が無い場合（中断要求フラグ１６３が０の場合）はステップＳ５２へ進む。
【００６２】
ステップＳ４５ではデータ転送中断のための処理を行う。
【００６３】
ステップＳ４６ではデータ転送中断が完了したか否かをチェックする。終了していなければステップＳ４５に戻る。終了していればステップＳ４７に進む。
【００６４】
ステップＳ４７では親プロセス１１２に対してパケット通信によりデータ転送中断完了の報告を行う。
【００６５】
ステップＳ４８ではプロセス１６２は親プロセス１１２からのＶＩ張替え要求の有無をチェックする。該チェックはスレッド１６４によって張替え要求フラグ１６３を監視することで行われる。ステップＳ４８において張替え要求がある場合（張替え要求フラグ１６３が１の場合）は、プロセス１６２が実行している通常の処理を一旦休止し、ＶＩ張替え処理のルーチンであるステップＳ４９へ進む。ステップＳ４８において張替え要求が無い場合（張替え要求フラグ１６３が０の場合）はステップＳ５２へ進む。
【００６６】
ステップＳ４９ではプロセス１６２は子プロセス１１５に対してＶＩの接続要求を行う。次にステップＳ５０では接続が許可されるのを待つ。
【００６７】
接続が許可されると、プロセス１６２はステップＳ５１で親プロセス１１２に対してＶＩの張替えが完了したことを報告する。該報告は親プロセス１１２に対して張替え完了報告用パケットを送信し張替え完了フラグ１２３を立てることにより行う。
【００６８】
次にステップＳ５２ではプロセス１６２はＶＩ張替え処理のルーチンから復帰する場合は、休止していたプロセス１６２の処理を再開し、そうでなければ通常の処理の実行する。
【００６９】
ステップＳ５３ではプロセス１６２の通常の処理において該プロセスが終了するか否かをチェックする。終了する場合はステップＳ５４へ進み該プロセスは終了する。終了しない場合は再度ステップＳ４４へ進み、これまでのステップを繰り返す。
【００７０】
以上がＶＩ張替えの際のプロセス１６２の動作である。
【００７１】
以上で説明した親プロセス１１２および子プロセス１１５およびプロセス１６２におけるそれぞれの処理が全体として時系列でどのように動作するかを図９に示すタイムチャートを用いて説明する。
【００７２】
まず時点ａにおいてクライアントからサーバ上の親プロセスに対して接続要求９１０を発行し、該親プロセスは時点Ａでそれを受け取る。なお図中、処理名称の始めに表記した括弧内の字句は該クライアントに対する処理の対象がサーバ上の親プロセスであるか子プロセスであるかを示す。以下同様である。続いて該親プロセスは時点Ｂで該クライアントに対して接続許可９１１を発行し、該クライアントは時点ｂでそれを受け取る。時点Ｘにおいて該親プロセスはＥＭＵ＿ＦＯＲＫ（）９１８によって子プロセスを生成する。時点Ｃ

において該親プロセスは該クライアントに対してデータ転送中断要求９１２を発行し、該クライアントは時点ｃでそれを受け取る。時点ｃ以降該クライアントはデータ転送中断処理（図示せず）を行い、中断処理が完了した時点ｄにおいて該親プロセスに対して中断完了報告９１３を発行する。該親プロセスは時点Ｄで該中断完了報告９１３を受け取り、その後時点Ｙにおいて該子プロセスへの既受信データのコピー９２２を行う。続いて時点Ｅにおいて該親プロセスは該クライアントに対して張替え要求９１４を発行し、該クライアントは時点ｅにおいてそれを受け取る。時点ｆにおいて該クライアントは該子プロセスに対して接続要求９１５を発行し、該子プロセスは時点Ｆにおいてそれを受け取る。続いて時点Ｇにおいて該子プロセスは該クライアントに対して接続許可９１６を発行し、該クライアントは時点ｇにおいてそれを受け取る。最後に時点ｈにおいて該クライアントは該親プロセスに対して張替え完了報告９１７を発行し、該親プロセスは時点Ｈでそれを受け取り、一連のＶＩ張替え処理が完了する。

【００７３】

ここで該親プロセスは時点Ｂ以降データ受信可能状態であり、該子プロセスは時点Ｇ以降データ受信可能状態である。また時点ｂｃ間および時点ｈ以降が該クライアントがデータ送信可能時間であり、時点ｄｇ間が送信中断時間である。もし該クライアントが時点ｂでデータの送信を開始した場合、時点Ｂ´において該親プロセスが該データを受信可能になる。しかし該親プロセスは中断完了報告９１３を受け取ることにより、該親プロセスは時点Ｄ以降データの受信が無いことを知る。したがってＶＩの張替えの前に該親プロセスがデータを受け取る可能性のある時間は時点Ｂ´Ｄ間であり、その間にデータを受け取った場合に該データを該子プロセスにコピーする。その結果、ＶＩ張替え完了後に該子プロセスが正しく該親プロセスの通信を引き継ぐことが可能になる。

【００７４】

一方図８は、データ転送の中断を行わなかった場合のタイムチャートである。図８を用いて本発明の有効性を示す。

【００７５】

まず図９と同様に該親プロセスは時点Ｂ以降データ受信可能状態であり、該子プロセスは時点Ｇ以降データ受信可能状態である。また時点ｂｃ間および時点ｈ以降が該クライアントがデータ送信可能時間であり、時点ｄｇ間が送信中断時間である。しかしながら該親プロセスは該クライアントが送信を中断し、それがサーバ側に反映される時点Ｅ´を知ることはできない。すなわち該親プロセスは該クライアントからのデータがいつまで送信されるかを知ることができない。したがって該親プロセスはどの時点で該子プロセスに対して既受信データのコピーを行えばよいかわからず、コピーが正しく行われる保証ができない。

【００７６】

これに対し、本実施態様では、先に図９で説明した通り、サーバ上の親プロセスはクライアント上のプロセスにデータ転送中断要求を出し、クライアント側から中断完了が返送されることによって、それまでに受信したデータを子プロセスにコピーすれば良いことが分かる。よってＶＩ張替えを行った場合に正しく通信を継続することを保証される。

【００７７】

以上述べた実施形態においては、プロセス間の通信は共有メモリを用いて実行しているが、その他のPIPE等の手段を使用してもよい。またホスト間の通信はパケット通信によって実行しているが、その他の手段を用いても良い。さらに第１のホストと第２のホストは物理的に分離されている必要はなく、同一情報処理装置上で実行されているプロセスにおいてもこの発明が適用されることは自明である。

【００７８】

【発明の効果】

上述のように、本発明によれば、ユーザ空間直接転送によりOSの介在なしに通信を行っているクライアント側のプロセスとサーバ側のプロセス（親プロセス）との間の通信回線を、自発的にサーバ側の別のプロセス（子プロセス）との間の通信回線に張替えて通信を継

続させ、ＵＮＩＸのＦＯＲＫ（）と同様の機能を実現することが出来る。

【図面の簡単な説明】

【図１】本発明の一実施形態のサーバ・クライアントシステムのブロック図である。

【図２】一般的なＴＣＰ／ＩＰにおけるソケット通信の概念を示す概念図である。

【図３】ユーザ空間直接転送で通信するサーバ・クライアントシステムシステムにける通信の状態を示すブロック図である。

【図４】上記実施態様における通信路を張替える際のサーバ側の親プロセスの処理を示すフローチャート。

【図５】上記実施態様における通信路を張替える際のサーバ側の子プロセスの処理を示すフローチャート。

【図６】上記実施態様における通信路を張替える際のクライアント側のプロセスの処理を示すフローチャート。

【図７】上記実施態様にて通信路を張替えるトリガとなるイベントを示す図。

【図８】実施態様と比較のための通信路を張替える際にサーバ側の親プロセスがクライアント側のプロセスに対してデータ転送の中断を要求しない場合の処理の進行を示すタイムチャート。

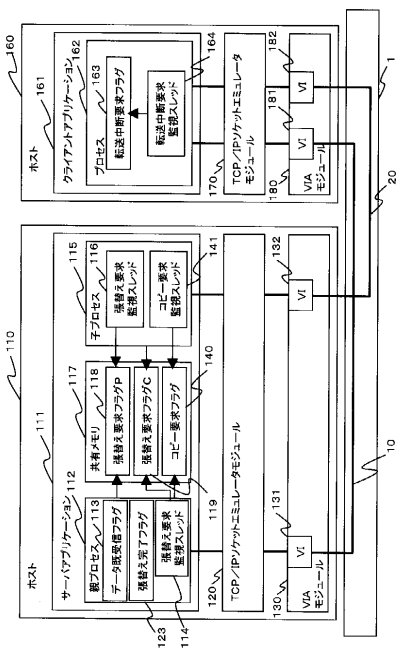
【図９】実施態様の通信路を張替える際の処理の進行を示すタイムチャート。

【符号の説明】

１…ネットワーク、１０，２０…通信回線、１１０，１６０…ホスト、１１１…サーバアプリケーション、１１２…親プロセス、１１６…子プロセス、１２０，１７０…ＴＣＰ／ＩＰソケット通信エミュレータブロック、１３０，１８０…ＶＩＡモジュール、１３１，１３２，１８１，１８２…ＶＩ、１６１…クライアントアプリケーション、１６２…プロセス。

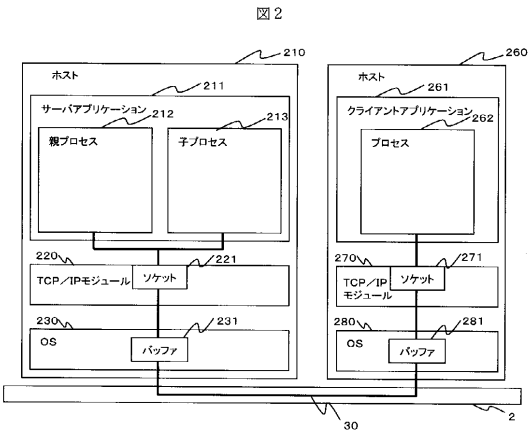
【図１】

図1



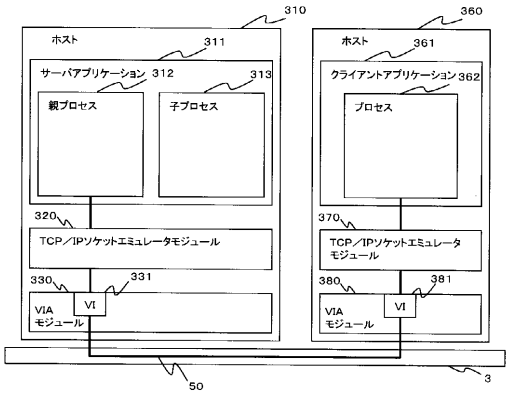
【図２】

図2



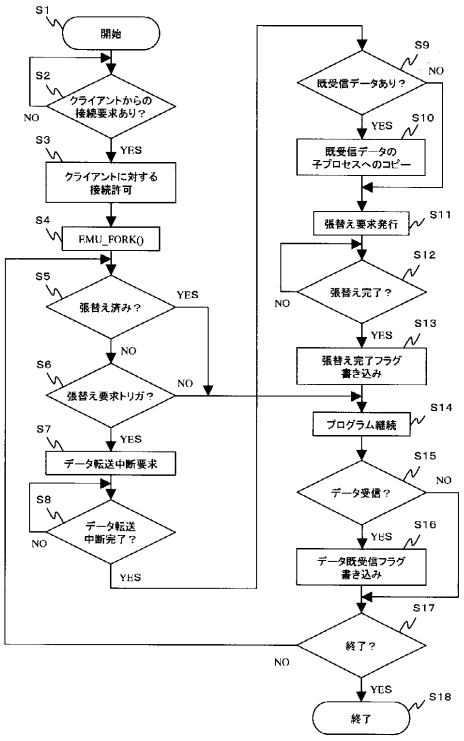
【図３】

図3



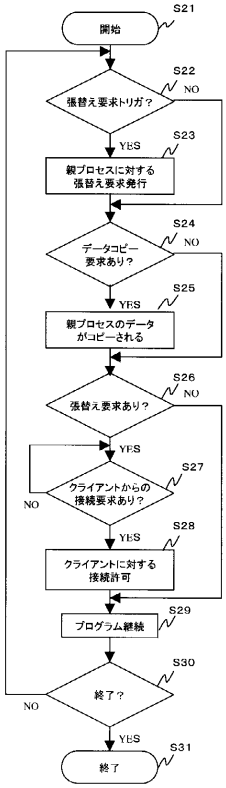
【図４】

図4



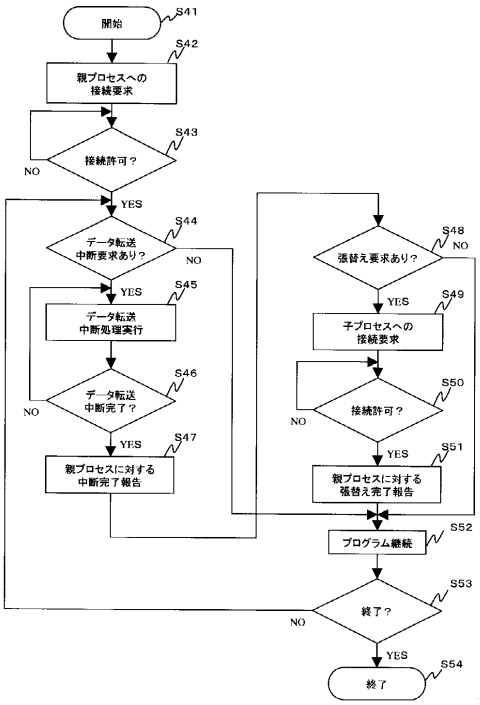
【図５】

図5



【図６】

図6



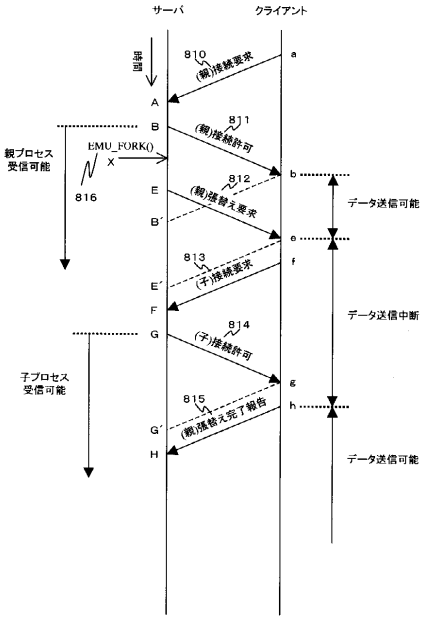
【図７】

図７

イベント	親プロセス	子プロセス
RECV	—	○ T3
SEND	— T1	○ T4
CLOSE	○ T2	—
プロセス消滅	○	—

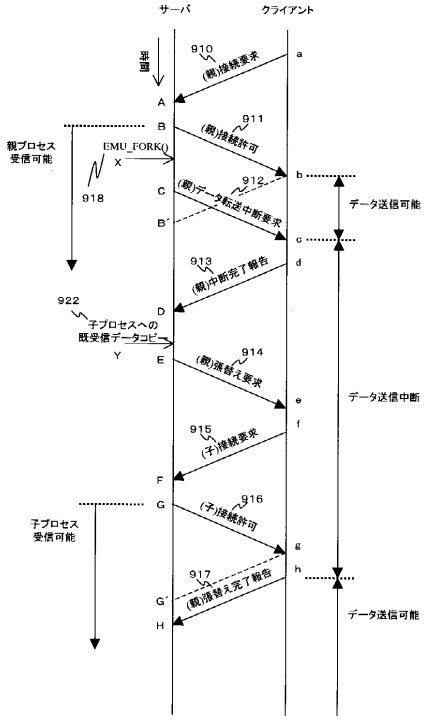
【図８】

図８



【図９】

図９



フロントページの続き

(72)発明者　今木　常之
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
(72)発明者　川本　真一
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内

審査官　鈴木　修治

(56)参考文献　特開２０００－０５７０９５（ＪＰ，Ａ）
特開平０９－１３８７５４（ＪＰ，Ａ）
特開平０８－０７７１２１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/46-9/54
G06F 15/16-15/177
G06F 15/00