(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4262888号
(P4262888)
(45)発行日　平成21年5月13日(2009.5.13)　(24)登録日　平成21年2月20日(2009.2.20)

(51)Int.Cl.　F I
*G06F 9/50 (2006.01)*　G06F 9/46 465A

請求項の数 28 (全 23 頁)

(21)出願番号　特願2000-553887(P2000-553887)
(86)(22)出願日　平成11年5月11日(1999.5.11)
(65)公表番号　特表2002-517855(P2002-517855A)
(43)公表日　平成14年6月18日(2002.6.18)
(86)国際出願番号　PCT/US1999/010273
(87)国際公開番号　WO1999/064952
(87)国際公開日　平成11年12月16日(1999.12.16)
審査請求日　平成18年4月27日(2006.4.27)
(31)優先権主張番号　09/097,169
(32)優先日　平成10年6月12日(1998.6.12)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100089705
弁理士　社本　一夫
(74)代理人　100071124
弁理士　今井　庄亮
(74)代理人　100076691
弁理士　増井　忠弐
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100096013
弁理士　富田　博行

最終頁に続く

(54)【発明の名称】処理タスクをソフトウエアからハードウエアにオフロードする方法およびコンピュータ・プログラム製品

(57)【特許請求の範囲】

【請求項１】

コンピュータと、少なくとも１つのソフトウエア・コンポーネントと、少なくとも１つの周辺ハードウエア・デバイスとを有するコンピュータ・システム環境において、パケット毎に、前記コンピュータ・システムのその時々の必要性に応じて、前記コンピュータから前記周辺ハードウエア・デバイスに動作タスクを動的にオフロードすることによって、前記コンピュータのプロセッサ資源を解放し、前記コンピュータ・システム全体の効率を向上させる方法であって、

前記コンピュータが、前記周辺ハードウエア・デバイスに問い合わせて、前記周辺ハードウエア・デバイスのタスク・オフロード能力を判定するステップと、

前記コンピュータが、前記周辺ハードウエア・デバイスの選択したタスク・オフロード能力をイネーブルして、１つ以上のパケットが必要とするイネーブルされたタスク・オフロード能力を提供するステップと、

前記コンピュータによってあるパケットに対して実行する動作タスクが前記周辺ハードウエア・デバイス上のイネーブルされたタスク・オフロード能力に対応する場合、前記コンピュータ・システムのその時々の必要性に応じて、前記コンピュータが、データ・パケットを前記周辺ハードウエア・デバイスに送り、前記周辺ハードウエア・デバイスが前記動作タスクを実行することを示すことにより、前記コンピュータから前記周辺ハードウエア・デバイスに前記動作タスクを選択的にオフロードする作用を行なうステップと、

前記周辺ハードウエア・デバイスが、前記オフロードされた動作タスクを実行するステ

ップと、
を含む方法。
【請求項２】
請求項１に記載した前記ステップおよび作用を実行するためのコンピュータ実行可能命令を有するコンピュータ読取可能媒体。
【請求項３】
請求項１記載の方法において、前記周辺ハードウエア・デバイスが、前記コンピュータ・システムに動作的に接続してあるネットワーク・インターフェース・カード（ＮＩＣ）であること、を特徴とする方法。
【請求項４】
請求項１記載の方法において、前記ソフトウエア・コンポーネントが、レイヤ状ネットワーク・モデルにおいて実行するネットワーク・ソフトウエア・アプリケーションであること、を特徴とする方法。
【請求項５】
請求項１記載の方法において、前記周辺ハードウエア・デバイスのタスク・オフロード能力を問い合わせることは、前記周辺ハードウエア・デバイスに関連する少なくとも１つのタスク・オフロード・バッファ内に格納してあるタスク・データを読み取ることにより行い、該タスク・データが前記周辺ハードウエア・デバイスの特定のタスク・オフロード能力を示すこと、を特徴とする方法。
【請求項６】
請求項５記載の方法において、前記周辺ハードウエア・デバイスには、複数のタスク・オフロード・バッファを関連付けてあり、これによって、当該周辺ハードウエア・デバイスの複数のタスク・オフロード能力を定めること、を特徴とする方法。
【請求項７】
請求項１記載の方法において、前記周辺ハードウエア・デバイスの選択したタスク・オフロード能力をイネーブルすることは、当該周辺ハードウエア・デバイスに関連するタスク・オフロード・バッファ内に、少なくとも１つのフラグ・インディケータをセットすることにより行う、を特徴とする方法。
【請求項８】
請求項１記載の方法において、前記パケットが、ネットワーク・データおよびパケット拡張データを含むネットワーク・データ・パケットであり、前記パケット拡張データが、前記周辺ハードウエア・デバイスが実行する少なくとも１つの動作タスクを示す少なくとも１つのデータ・フィールドを含むこと、を特徴とする方法。
【請求項９】
請求項８記載の方法において、前記周辺ハードウエア・デバイスが、ネットワーク・インターフェース・カード（ＮＩＣ）であること、を特徴とする方法。
【請求項１０】
請求項１記載の方法において、チェックサム動作、暗号化動作、メッセージ・ダイジェスト計算動作、ＴＣＰ区分動作、および解読動作である動作タスクのうちの１つ以上から前記動作タスクを選択すること、を特徴とする方法。
【請求項１１】
請求項１記載の方法において、前記ソフトウエア・コンポーネントがトランスポート・プロトコル・ドライバ・ソフトウエア・コンポーネントを備え、前記コンピュータ・システムがネットワーク・インターフェース・カード（ＮＩＣ）デバイス・ドライバを含み、前記周辺ハードウエア・デバイスが、前記コンピュータ・システムに接続した対応のネットワーク・インターフェース・カード（ＮＩＣ）を備え、
前記周辺ハードウエア・デバイスに問い合わせて当該周辺ハードウエア・デバイスのタスク・オフロード能力を判定する前記ステップが、
前記ＮＩＣのタスク・オフロード能力を識別するタスク・データを含むデータ構造を作成する作用と、

前記データ構造内に収容したタスク・データを問い合わせて、前記ＮＩＣのタスク・オフロード能力を識別する作用と、
を含み、
前記動作タスクを選択的にオフロードする前記作用が、タスク・オフロード・データを前記パケットに添付する作用を含み、
前記オフロードされた動作タスクを実行する前記ステップが、前記添付したタスク・オフロード・データにしたがって、前記ＮＩＣにおいて前記少なくとも１つのオフロードされたタスクを実行する作用を含むこと、を特徴とする方法。
【請求項１２】
請求項１１に記載のステップおよび作用を実行するためのコンピュータ実行可能命令を有するコンピュータ読取可能媒体。
【請求項１３】
請求項１１記載の方法において、前記データ構造が、
タスク・オフロード能力のタイプを表すデータを収容する第１データ・フィールドと、
前記第１データ・フィールドにおいて定めたタスク・オフロードを実行するために必要なデータを収容する第２データ・フィールドと、
を備えること、を特徴とする方法。
【請求項１４】
請求項１１記載の方法において、前記データ構造内に少なくとも１つのデータ・フィールドを書き込むことによって、前記ＮＩＣを選択的にイネーブルし、前記識別したタスク・オフロード能力の少なくとも１つを実行すること、を特徴とする方法。
【請求項１５】
請求項１１記載の方法において、前記パケットに添付した前記タスク・オフロード・データが、前記ＮＩＣが前記パケットを受信したときに、当該ＮＩＣが実行するオフロードされたタスクのタイプを表す第１データ・フィールドを備えること、を特徴とする方法。
【請求項１６】
請求項１５記載の方法において、前記タスク・オフロード・データが、更に、メモリ位置への少なくとも１つのポインタを収容する第２データ・フィールドを備え、前記メモリ位置が、前記タスク・オフロード・データの第１データ・フィールドにおいて指定した前記オフロードされたタスクを実行するために必要なデータを収容すること、を特徴とする方法。
【請求項１７】
コンピュータを備えたコンピュータ・システムのその時々の必要性に応じて、パケット毎に、計算タスクを動的にオフロードして、前記コンピュータに接続してあるネットワーク・インターフェース・カード（ＮＩＣ）において実行されるようにすることにより、前記コンピュータのプロセッサ資源を解放し、前記コンピュータ・システム全体の効率を向上させるコンピュータ実行可能命令を有するコンピュータ読取可能媒体であって、
前記コンピュータ上で実行するアプリケーションに、前記ＮＩＣのあらゆるタスク・オフロード処理能力について通知するステップと、
前記コンピュータ上で実行するアプリケーションから前記ＮＩＣに向けたネットワーク・データ・パケット毎に、該データ・パケットに添付するパケット拡張データ構造を検査するステップであって、前記パケット拡張データ構造が、前記ＮＩＣにオフロードする１つ以上の特定の動作タスクの識別に対する情報を収容する、ステップと、
前記ネットワーク・データ・パケットに添付する前記パケット拡張データ構造内に収容してあるデータにしたがって、前記ＮＩＣに動作タスクを実行させるステップと、
を含むステップを実行させるコンピュータ実行可能命令を更に含むコンピュータ読取可能媒体。
【請求項１８】
請求項１７記載のコンピュータ読取可能媒体において、前記通知するステップを実行する前記コンピュータ実行可能命令が、前記ＮＩＣのタスク・オフロード能力のタイプを表

すデータを収容する第１データ・フィールドを備えるタスク・オフロード・バッファ・データ構造を作成するステップを含むこと、を特徴とするコンピュータ読取可能媒体。

【請求項１９】

請求項１８記載のコンピュータ読取可能媒体において、前記タスク・オフロード・バッファ・データ構造が、更に、前記第１データ・フィールドにおいて定めた前記タスク・オフロードを実行するために必要なデータを収容する少なくとも１つの追加データ・フィールドを備えること、を特徴とするコンピュータ読取可能媒体。

【請求項２０】

請求項１７記載のコンピュータ読取可能媒体において、前記ネットワーク・データ・パケットに添付する前記パケット拡張データ構造が、前記ＮＩＣが実行する動作タスクのタイプを表す第１データ・フィールドを備えること、を特徴とするコンピュータ読取可能媒体。

【請求項２１】

請求項２０記載のコンピュータ読取可能媒体において、前記パケット拡張データ構造が、更に、メモリ位置への少なくとも１つのポインタを収容する少なくとも１つの追加データ・フィールドを備え、前記メモリ位置が、前記パケット拡張データ構造の第１データ・フィールドにおいて指定した動作タスクを実行するために必要なデータを収容すること、を特徴とするコンピュータ読取可能媒体。

【請求項２２】

コンピュータと、少なくとも１つのソフトウエア・コンポーネントと、少なくとも１つの周辺ハードウエア・デバイスとを有するコンピュータ・システム環境において、パケット毎に、前記コンピュータ・システムのその時々の必要性に応じて、前記コンピュータから前記周辺ハードウエア・デバイスに動作タスクを動的にオフロードすることによって、前記コンピュータのプロセッサ資源を解放し、前記コンピュータ・システム全体の効率を向上させる方法であって、

前記周辺ハードウエア・デバイスに関連した少なくとも１つのタスク・オフロード・バッファに格納してあるタスク・データを読み取る作用であって、前記タスク・データが、前記周辺ハードウエア・デバイスの特定のタスク・オフロード能力を示す、作用と、

前記少なくとも１つのタスク・オフロード・バッファ内に収容してあるタスク・データを問い合わせて、前記周辺ハードウエア・デバイスのタスク・オフロード能力を識別する作用と、

前記周辺ハードウエア・デバイスに関連したタスク・オフロード・バッファ内に、少なくとも１つのフラグ・インディケータをセットする作用であって、前記フラグ・インディケータが１つ以上のパケットに必要な関連するタスク・オフロード能力のイネーブルを表す、作用と、

前記コンピュータによってパケットに対して実行する動作タスクが前記周辺ハードウエア・デバイス上のイネーブルされたタスク・オフロード能力に対応する場合、前記コンピュータ・システムのその時々の必要性に応じて、前記コンピュータが、タスク・オフロード・データをデータ・パケットに添付し、該データ・パケットを前記周辺ハードウエア・デバイスに送ることによって、前記コンピュータから前記周辺ハードウエア・デバイスに前記動作タスクを選択的にオフロードする作用を実行し、

前記添付したタスク・オフロード・データにしたがって、前記周辺ハードウエア・デバイスにおいて前記少なくとも１つのオフロードされたタスクを実行する作用と、

を含む方法。

【請求項２３】

請求項２２に記載の作用を実行するためのコンピュータ実行可能命令を有するコンピュータ読取可能媒体。

【請求項２４】

請求項２２記載の方法において、前記周辺ハードウエア・デバイスが、前記コンピュータ・システムに動作的に接続してあるネットワーク・インターフェース・カード（ＮＩＣ

）であること、を特徴とする方法。
【請求項２５】
請求項２２記載の方法において、前記ソフトウエア・コンポーネントが、レイヤ状ネットワーク・モデルにおいて実行するネットワーク・ソフトウエア・アプリケーションであること、を特徴とする方法。
【請求項２６】
請求項２２記載の方法において、前記周辺ハードウエア・デバイスには、複数のタスク・オフロード・バッファを関連付けてあり、これによって前記周辺ハードウエア・デバイスの複数のタスク・オフロード能力を定めること、を特徴とする方法。
【請求項２７】
請求項２２記載の方法において、前記周辺ハードウエア・デバイスがネットワーク・インターフェース・カード（ＮＩＣ）であること、を特徴とする方法。
【請求項２８】
請求項２２記載の方法において、チェックサム動作、暗号化動作、メッセージ・ダイジェスト計算動作、ＴＣＰ区分動作、および解読動作である動作タスクのうちの１つ以上から前記動作タスクを選択すること、を特徴とする方法。
【発明の詳細な説明】
【０００１】
（発明の背景）
（１．発明の分野）
本発明は、一般的に、コンピュータ・システムの効率、速度および／またはスループットを向上させる方法に関する。更に特定すれば、本発明は、典型的にホスト・プロセッサがソフトウエアで実行する計算タスクを特定のハードウエア・コンポーネントにオフロード（offload）することにより、ホスト・コンピュータ資源を開放し、コンピュータ・システムの全体的効率を高める方法に関する。
【０００２】
（２．従来技術の状態）
機能的コンピュータ・システムは、概略的に、３つの基本コンポーネントで構成されている。第１のコンポーネントは、ホスト・コンピュータおよびそれに付随する周辺ハードウエア・コンポーネントである。ホスト・コンピュータは、典型的に、中央演算装置（ＣＰＵ）を含み、バスを介して、例えば、ＲＡＭまたはＲＯＭのようなシステム・メモリと相互接続してある。また、システムは、必要な機能性に応じて、磁気または光のディスク記憶装置、キーボードまたはその他の入力デバイス、ディスプレイまたはその他の出力デバイス、ならびにモデムおよび／またはネットワーク・インターフェース・カード（ＮＩＣ）のような通信機器といった、多数の周辺ハードウエア・デバイスも含む。別の機能的コンピュータ・コンポーネントに、アプリケーション・ソフトウエアがある。このようなソフトウエアは、周知のワード・プロセッサ・アプリケーション、スプレッド・シート・アプリケーション、データベース・アプリケーション、通信およびネットワーク・アプリケーション等を含む。
【０００３】
最近の機能的コンピュータ・システムの最後のコンポーネントは、オペレーティング・システムである。コンピュータのオペレーティング・システムは、ユーザにアプリケーション・プログラムの実行を開始させるというような、多くの機能を実行する。加えて、最新のオペレーティング・システムは、アプリケーション・ソフトウエアとホスト・コンピュータおよびその周辺ハードウエアとの間のインターフェースも備えている。したがって、以前にはアプリケーション・プログラムが直接コンピュータ・システムのハードウエアにアクセスすることが当たり前であったが、最新のオペレーティング・システムは標準化された一貫性のあるインターフェースを備え、ユーザ・アプリケーションに、標準化された方法でコンピュータ・ハードウエア周辺機器とインターフェースし、これにアクセスさせるようにしている。一貫性のあるインターフェースを備えるために、オペレーティング・

システムのアーキテクチャの設計において、実際のハードウエア周辺機器およびアプリケーション・プログラム間にいくつものソフトウエア・レイヤが存在し得るようにする場合が増えている。例えば、アプリケーションは、オペレーティング・システムにコールすることができる。一方、オペレーティング・システムは、ハードウエア・デバイス・ドライバ・レイヤが与えるサービスを利用することができる。そのときには、デバイス・ドライバ・レイヤは、特定のハードウエア周辺機器と直接インターフェースする。このようなレイヤ型手法の主な利点は、他のレイヤに影響を及ぼすことなく、レイヤの追加または交換が可能なことである。

【０００４】

認識されるように、このようなオペレーティング・システム、アプリケーション・ソフトウエア、ならびにネットワークおよび通信は、増々複雑化および精巧化し続けている。勿論、その結果、高機能化し有用性が高いコンピュータ・システムが得られる。しかしながら、この機能性の増大は、コストを伴わない訳ではない。オペレーティング・システムおよびソフトウエア・アプリケーションでは、機能が更に増加するに連れて、このようなシステム機能および／またはアプリケーションを実行するときのプロセッサ／ＣＰＵが実行しなければならない動作が増加し、その結果プロセッサのオーバーヘッド増大を招く場合が多い。この現象は、特に、ネットワーク通信型ソフトウエア・アプリケーションのような、特定の種類のアプリケーションと比較すると、特に明白となる。広帯域幅媒体が増々普及するに連れて、ネットワーク速度は、ホスト・コンピュータのＣＰＵのプロセッサ速度やメモリ帯域幅に匹敵するかあるいはこれを凌駕することも多い。したがって、このようなネットワークを通じて効率的に通信するためには、ネットワーク接続するホスト・コンピュータのＣＰＵ利用およびメモリ帯域幅利用を極力抑えなければならない。

【０００５】

加えて、ネットワーク・アプリケーションは、７層ＩＳＯモデルのような殆どが用いているレイヤ状アーキテクチャ、またはＷｉｎｄｏｗｓ　ＮＴオペレーティング・システムが用いているレイヤ・モデルにより、ホスト・プロセッサに更に負担をかける。公知のように、このようなモデルは、ネットワークへの物理的接続とエンド・ユーザ・アプリケーションとの間のデータ・フローを記述するために用いられる。ネットワーク・ケーブル上にデータ・ビットを置くというような、最も基本的な機能はボトム・レイヤにおいて行われ、一方アプリケーションの詳細に関与する機能はトップ・レイヤにおいて行われる。本質的に、各レイヤの目的は、次に高いレイヤにサービスを提供し、サービスが実際にどのように実施されるかに関する詳細から、高いレイヤを遮蔽することである。レイヤは、各レイヤが、ネットワークを介して通信している別のコンピュータ上にある同じレイヤと通信していると考えるように、抽象化されている。

【０００６】

データ・パケットがレイヤ間を移動する際にこれに対して実行される種々の機能はソフトウエア集中的となり得るものであり、したがってかなりの量のＣＰＵプロセッサおよびメモリ資源を要求する可能性があることは認められよう。例えば、Ｗｉｎｄｏｗｓ　ＮＴネットワーキング・モデルでは、種々のレイヤにおいてパケットに対して行われるある機能は、パケット・チェックサムの算出および検証、データの暗号化および解読、メッセージ・ダイジェスト計算およびＴＣＰセグメント化のように、極度にＣＰＵ集中的である。これらの機能の各々を実行する際、その結果としてのＣＰＵ／メモリに対する要求は、コンピュータ・システム全体のスループットおよび性能に大きく影響する可能性がある。

【０００７】

ソフトウエア・アプリケーションおよびオペレーティング・システムの機能がコンピュータ・システム資源に対して行なう要求は増大するが、同時に、ネットワーク・インターフェース・カード（ＮＩＣ）のような、多くのコンピュータ・ハードウエア周辺機器の能力、効率およびスループットも向上する。これらのコンピュータ・システム周辺機器には、多くの場合専用プロセッサおよびメモリが搭載されており、典型的に非常に精巧化され複雑なタスク、即ち、コンピュータ・システムのプロセッサがソフトウエアで実行するよう

なタスクを実行することができる。例えば、多くのＮＩＣは、チェックサム計算／検証、データ暗号化／解読、メッセージ・ダイジェスト計算、ＴＣＰセグメント化およびその他というような、適切なネットワーク・レイヤにおいてＣＰＵがソフトウエアで実行するタスクを、独立して実行することができる。したがって、このようなＣＰＵ集約タスクを周辺のハードウエア・デバイスにオフロードすることは効果的である。これが可能であれば、ホスト・コンピュータにおけるプロセッサの利用度およびメモリ帯域幅の使用度が低下することにより、システム全体の効率、速度およびスループットが向上するであろう。

【０００８】

しかしながら、異なる周辺デバイスの処理能力は大きく異なる。したがって、コンピュータ・システム／オペレーティング・システムがこのような周辺デバイスの処理能力を識別し、次いで必要に応じて特定の処理タスクをデバイスに割り当てオフロードすることを可能にする効率的な方法が必要となる。また、その時々のプロセッサの必要性に応じて動的にタスクを識別し割り当てることができれば望ましいであろう。これが可能であれば、コンピュータ・システムのプロセッサは、必要に応じてハードウエア周辺機器の能力を利用できるようになるであろう。

（発明の概要）

従来技術の現状における前述の問題は、本発明が見事に解決した。本発明は、以前はプロセッサ・ソフトウエア・レベルで行われていた機能およびタスクを、コンピュータ・システムに接続してある適切なハードウエア周辺機器にオフロードするシステムおよび方法に関する。本発明は、特に、コンピュータのＣＰＵがソフトウエアで実行するタスクの多くを実行できる場合が多い、ネットワーク・インターフェース・カード（ＮＩＣ）周辺デバイスに対する、タスクのオフロードにおいて特に有用である。

【０００９】

本発明の好適な実施形態の１つでは、例えば、オペレーティング・システム（ＯＳ）が、コンピュータ・システムに接続してあるあらゆるハードウエア周辺機器（ＮＩＣ等）のデバイス・ドライバ（「ＭＡＣ」ドライバと呼ばれる場合が多い）に「問い合わせ」することを可能にする、ソフトウエア実装方法およびプロトコルを提供する。種々のデバイス・ドライバは、各々、それぞれのハードウエア周辺機器処理能力を識別することによって応答する。この能力のことをここでは「タスク・オフロード能力」と呼ぶ。好適な実施形態では、一旦個々の周辺機器のタスク・オフロード能力を識別したなら、ＯＳは選択した周辺機器に、ＯＳによって潜在的に使用可能なあるタスクを実行させることができる。その後、ＯＳは、その時々のコンピュータ・システムの処理の必要性にしたがって、動的に、要求に応じて、先にイネーブルしたタスクまたは複数のタスクを周辺機器が実行するように要求する。

【００１０】

この概略的な発明概念は、他のアプリケーションまたはオペレーティング・システム環境にも適用可能であるが、本発明の実施形態は、ここでは、Ｗｉｎｄｏｗｓ　ＮＴのレイヤ状ネットワーキング・モデルと共に実施し利用するものとして説明する。勿論、本発明は、ネットワーク通信を管理し制御するための同様の種類のアーキテクチャであれば、本質的にいずれとでも実施可能である。即ち、本発明は、典型的にネットワーク・パケットに対して例えば種々のネットワーク・レイヤにおいて行われ、典型的に専用ＣＰＵおよびメモリ資源を必要とするタスクまたは機能をオフロードする機能を提供する。これらのオフロードしたタスクは、任意に、ネットワークに実際の物理的通信チャネルを提供するハードウエア周辺機器、即ち、ＮＩＣによって代わりに実行することができる。例えば、データ・パケットがそれぞれのネットワーク・レイヤを通過する際に、例えば、チェックサム計算／検証、暗号化／解読、メッセージ・ダイジェスト計算およびＴＣＰセグメント化というようなデータ・パケットに対するＣＰＵ集約動作の一部を実行せずに、これらのタスクをオフロードし、ＮＩＣハードウエアにおいて代わりに実行することができる。

【００１１】

本発明の好適な実施形態では、Ｗｉｎｄｏｗｓ　ＮＴレイヤ状ネットワーキング・アーキ

テクチャにおいて、トランスポート・プロトコル・ドライバ、即ち、トランスポートを適切なプログラム方法によって実現し、コンピュータに接続してある対応のＮＩＣ（複数のＮＩＣ）と関連するデバイス・ドライバ（複数のデバイス・ドライバ）の各々に問い合わせができるようにする。問い合わせされた各デバイス・ドライバも、同様に、その具体的な処理能力、即ち、「タスク・オフロード」能力を識別することによって、応答可能なように実現する。好適な実施形態の１つでは、一旦個々の周辺デバイスのタスク・オフロード能力を識別したなら、トランスポートが、これら具体的な能力のどれをイネーブルするかについて設定を行なう。これは、本質的に、周辺デバイスに、続くデータ・パケットの送信および／または受信の間にどのような種類のタスクを実行することになっているのかについて知らせることになる。その後、トランスポートは、要求に応じて、周辺デバイスの機能をイネーブルし、これを利用することができる。好ましくは、イネーブルした機能は、ネットワーク・チャネルに宛てられた実際のデータ・パケットに添付されている適切なデータによって呼び出す。このようにして、タスクを動的にオフロードすることができ、一度に１つより多いタスクをオフロードすることができる。

【００１２】

したがって、ネットワーク・パケットを特定の下位デバイス・ドライバ（例えば、Ｗｉｎｄｏｗｓ　ＮＴ環境におけるＭＡＣサブレイヤに常駐するデバイス・ドライバ）に送る前に、トランスポートは、最初に、対応するＮＩＣの能力がどれくらいなのかについて判定を行なう。特定の機能または複数の機能が可能な場合、トランスポートは所望の機能をイネーブルする。続くパケットの送信中、トランスポートが特定のタスクをハードウエアにオフロードすることを望む場合、当該所望の機能（複数の機能）をＮＩＣハードウエアにおいてそのパケットに対して実行することを意味する情報を、パケットに動的に添付することができる。例えば、トランスポートは、データ・パケット内にデータ・フラグをセットすることによって、対応するデバイス・ドライバに、ＮＩＣはチェックサムを計算し該当の発信パケットに添付することを通知する。このときには、対応するＮＩＣ上のハードウエア／ソフトウエアは、それ自体でこの特定のパケット処理に対処し、システムＣＰＵからの介入や補助は全くない。したがって、システム・プロセッサは解放され、別の処理タスクを実行するので、システム全体の効率やスループットが向上する。

【００１３】

先に注記したように、好適な実施形態の１つでは、タスクを動的にダウンロードする。即ち、その時々のコンピュータ・システムに対する必要性に応じて、ＮＩＣの能力を選択的にパケット毎に用いることができる。更に、以前にはネットワーク・スタックの種々のレベルで実行していたタスクが、今や単一点、即ち、ＮＩＣ自体で実行されるので、この手法は統合化および効率化を一層推し進め、システム全体のスループットを更に高めることになる。好ましくは、本発明の実施形態は、動作を「バッチ」する。即ち、多数のタスクを単一のＮＩＣにオフロードする機能を、トランスポートに設ける。例えば、単一のＮＩＣがチェックサム計算および暗号化双方を１つのパケットに対して実行することにより、同じ機能をそれぞれのソフトウエア・レイヤにおいてソフトウエアで実施する場合に必要となる多数のＣＰＵサイクルを不要とすることができる。

【００１４】

したがって、本発明は、コンピュータ・システム・プロセッサから、当該コンピュータ・システムに接続してあるハードウエア周辺機器に計算タスクをオフロードするシステムおよび方法を提供する。また、本発明は、個々の周辺機器の処理能力を識別するシステムおよび方法を提供する。本発明は、レイヤ状ネットワーク・アーキテクチャと共に効率的に使用することにより、ネットワーク内の種々のレイヤにおいて典型的に実行するタスクを、代わりに、適切なネットワーク・インターフェース・カード（ＮＩＣ）にオフロードすることを可能にする、タスク・オフロード・システムおよび方法を提供する。本発明は、コンピュータ・プロセッサの現処理状態にしたがって、動的に、要求に基づいて計算タスクをオフロードすることができるシステムおよび方法を提供する。更に、本発明は、多数のタスクを共にバッチし、次いでＮＩＣのような単一の周辺デバイスにオフロードするこ

とができるシステムおよび方法を提供する。

【００１５】

本発明のその他の利点は、以下の説明に明記されており、その記載から部分的に明らかとなり、あるいは本発明の実施によって習得することができよう。本発明の利点は、添付した特許請求の範囲に特定的に指摘した手段および組み合わせによって実現し、得ることができる。本発明のこれらおよびその他の特徴は、以下の説明および添付した特許請求の範囲から一層明らかとなり、以下に明記する本発明の実施によって習得することができよう。

【００１６】

（図面の簡単な説明）

先に引用した本発明の利点およびその他の利点が得られるようにするために、先に端的に説明した本発明の更に特定的な説明を、添付図面に示す、その具体的な実施形態を参照しながら行なう。これらの図面は本発明の典型的な実施形態のみを図示するのであり、したがってその範囲を限定するものとはみなされないという理解の下で、添付図面を用いて、本発明を更に具体的かつ詳細に、記載し説明する。

【００１７】

（好適な実施形態の詳細な説明）

以下、本発明を説明するにあたって、本発明のシステムおよび方法を実現するために用いる実施形態の構造または処理を示す図を用いることにする。このように図面を用いて本発明を提示することは、その範囲を限定するものとして解釈されるべきでない。本発明は、パーソナル・コンピュータのようなホスト・コンピュータから、ネットワーク・インターフェース・カード（ＮＩＣ）のような、コンピュータに接続してあるハードウエア周辺機器に処理タスクをオフロードする方法およびシステム双方を想定している。図１および以下の論述は、本発明を実現可能な適当な計算機環境の端的な概説を与えることを意図したものである。必須ではないが、本発明の実施形態の説明は、総じて、パーソナル・コンピュータが実行するプログラム・モジュールのような、コンピュータ実行可能命令に関して行なう。一般に、プログラム・モジュールは、ルーティン、プログラム、オブジェクト、コンポーネント、データ構造等、特定のタスクを実行するか、あるいは特定の抽象データ・タイプを実施するものを含む。更に、本発明は、ハンドヘルド・デバイス、マルチプロセッサ・システム、マイクロプロセッサを用いたまたはプログラム可能な消費者用電子機器、ネットワークＰＣ、ミニコンピュータ、メインフレーム・コンピュータ等を含む、その他のコンピュータ・システム・コンフィギュレーションとも実施可能である。また、本発明の実施形態は、通信ネットワークを通じてリンクしたリモート処理デバイスによってタスクを実行する分散計算機環境においても実施可能である。分散計算機環境では、プログラム・モジュールは、ローカルおよびリモートのメモリ記憶装置双方に位置することができる。

【００１８】

図１を参照すると、本発明を実施するシステム例は、従来のパーソナル・コンピュータ２０の形態の汎用計算機を含み、演算装置２１（ＣＰＵとも呼ぶ）、システム・メモリ２２、およびシステム・メモリを含む種々のシステム・コンポーネントを演算装置２１に結合するシステム・バス２３を含む。システム・バス２３は、メモリ・バスまたはメモリ・コントローラ、周辺バス、および種々のバス・アーキテクチャのいずれかを用いたローカル・バスのいずれでもよい。システム・メモリは、リード・オンリ・メモリ（ＲＯＭ）２４およびランダム・アクセス・メモリ（ＲＡＭ）２５を含む。基本入出力システム２６（ＢＩＯＳ）は、起動中のように、パーソナル・コンピュータ２０内のエレメント間におけるデータ転送を補助する基本的なルーティンを含み、ＲＡＭ２４内に格納されている。更に、パーソナル・コンピュータ２０は、図示しないハード・ディスクの読み書きを行なうハード・ディスク・ドライブ２７、リムーバブル磁気ディスク２９の読み書きを行なう磁気ディスク・ドライブ２８、ＣＤ　ＲＯＭまたはその他の光媒体のようなリムーバブル光ディスク２９の読み書きを行なう光ディスク・ドライブ３０のような種々の周辺ハードウエ

ア・デバイスも含む。ハード・ディスク・ドライブ２７、磁気ディスク・ドライブ２８、および光ディスク・ドライブ３０は、それぞれ、ハード・ディスク・ドライブ・インターフェース３２、磁気ディスク・ドライブ・インターフェース３３、および光ドライブ・インターフェース３４を介して、システム・バス２３に接続されている。ドライブおよびそれに関連するコンピュータ読取可能媒体は、コンピュータ読取可能命令、データ構造、プログラム・モジュールおよびパーソナル・コンピュータ２０のその他のデータの不揮発性格納を行なう。ここに記載する環境の一例は、ハード・ディスク、リムーバブル磁気ディスク２９およびリムーバブル光ディスク３１を採用するが、磁気カセット、フラッシュ・メモリ・カード、ディジタル・ビデオ・ディスク、ベルヌーイ・カートリッジ、ランダム・アクセス・メモリ（ＲＡＭ）、リード・オンリ・メモリ（ＲＯＭ）等のように、コンピュータによるアクセスが可能なデータを格納することができる、別の形式のコンピュータ読取可能媒体も、動作環境例では使用可能であることは、当業者には認められよう。

【００１９】

ハード・ディスク、磁気ディスク２９、光ディスク３１、ＲＯＭ２４またはＲＡＭ２５上には、多数のプログラム・モジュールを格納可能であり、オペレーティング・システム３５、１つ以上のアプリケーション・プログラム３６、その他のプログラム・モジュール３７、およびプログラム・データ３８を含む。ユーザは、キーボード４０およびポインティング・デバイス４２のような入力デバイスによって、コマンドおよび情報をパーソナル・コンピュータ２０に入力することができる。他の入力デバイス（図示せず）は、マイクロフォン、ジョイスティック、ゲーム・パッド、衛星パラボラアンテナ、スキャナ等を含むことができる。これらおよびその他の入力デバイスは、多くの場合、システム・バスに結合するシリアル・ポート・インターフェース４６のような周辺ハードウエア・デバイスを介して、演算装置２１に接続されるが、パラレル・ポート、ゲーム・ポートまたはユニバーサル・シリアル・バス（ＵＳＢ）のようなその他のインターフェースによって接続することも可能である。また、ビデオ・アダプタ４８のような周辺ハードウエア・インターフェース・デバイスを介して、モニタ４７またはその他の種類のディスプレイ装置もシステム・バス２３に接続してある。モニタに加えて、パーソナル・コンピュータは、典型的に、スピーカおよびプリンタのような、その他の周辺出力デバイス（図示せず）を含む。

【００２０】

パーソナル・コンピュータ２０は、リモート・コンピュータ４９のような１つ以上のリモート・コンピュータへの論理接続を用いれば、ネットワーク環境においても動作可能である。リモート・コンピュータ４９は、別のパーソナル・コンピュータ、サーバ、ルータ、ネットワークＰＣ、ピア・デバイス、またはその他の共通ネットワーク・ノードとすることができ、典型的に、パーソナル・コンピュータ２０に関して先に述べたエレメントの多くまたは全てを含むが、図１にはメモリ記憶装置５０のみを図示している。図１に示す論理接続は、ローカル・エリア・ネットワーク（ＬＡＮ）５１およびワイド・エリア・ネットワーク（ＷＡＮ）５２を含む。このようなネットワーク環境は、会社全域に及ぶコンピュータ・ネットワーク、イントラネットおよびインターネットでは一般的である。

【００２１】

ＬＡＮネットワーク環境で用いる場合、パーソナル・コンピュータ２０は、ネットワーク・インターフェース・カード（ＮＩＣ）またはアダプタ５３と多くの場合呼ばれている、周辺ハードウエア・デバイスを介してローカル・ネットワーク５１に接続する。ＷＡＮネットワーク環境で用いる場合、パーソナル・コンピュータ２０は、典型的に、モデム５４、またはインターネットのようなワイド・エリア・ネットワーク５２を通じて通信を確立するその他の手段を含む。モデムは、内蔵でも外付けでもよく、典型的に、シリアル・ポート・インターフェース４６を介してシステム・バス２３に接続する。ネットワーク環境では、パーソナル・コンピュータ２０に関して図示したプログラム・モジュールまたはその一部は、リモート・メモリ記憶装置５０に格納することも可能である。尚、図示のネットワーク接続は一例であり、コンピュータ間に通信リンクを確立する別の手段も使用可能であることは認められよう。

【００２２】
また、本発明の範囲内の実施形態は、実行可能な命令を有するコンピュータ読取可能媒体も含む。このようなコンピュータ読取可能媒体は、汎用コンピュータまたは特殊目的コンピュータがアクセスすることができる、入手可能なあらゆる媒体とすることができる。限定ではなく、一例として、このようなコンピュータ読取可能媒体は、ＲＡＭ、ＲＯＭ、ＥＰＲＯＭ、ＣＤ－ＲＯＭまたはその他の光ディスク・ストレージ、磁気ディスク・ストレージまたはその他の磁気記憶ストレージ、あるいは所望の実行可能命令を格納するために使用でき、汎用コンピュータまたは特殊目的コンピュータがアクセスすることができるその他のあらゆる媒体とすることができる。前述の組み合わせも、コンピュータ読取可能媒体の範囲に含まれて当然である。実行可能命令は、例えば、汎用コンピュータ、特殊目的コンピュータ、または特殊目的処理デバイスに、ある機能または機能群を実行させる命令およびデータから成る。最後に、本発明の範囲内の実施形態は、データ構造を表す複数のデータ・フィールドが格納されているコンピュータ読取可能媒体を備えている。

【００２３】
本発明の実施形態は、演算装置２１の処理オーバーヘッドおよびメモリ使用量を削減する機能を備えることを目的とする。これを達成するには、例えば、オペレーティング・システム、アプリケーション・プログラムおよび／または演算装置／ＣＰＵ２１上で実行するその他のプログラム・モジュールによって実行する特定の計算タスクを、コンピュータ・システム２０に接続してある適切な周辺ハードウエア・デバイスにオフロードする。このような周辺デバイスの多くは、増々専用のプロセッサやメモリが搭載されるようになっており、典型的にＣＰＵ２１のみが実行する同じタスクの多くを完全に実行することができる。このようなデバイスの例は、例えば、ネットワーク・インターフェース・カード（図１における５３）、ディスク・ドライブ・インターフェース・カード（例えば、図１における３２、３３、３４）、小型コンピュータ・システム・インターフェース（ＳＣＳＩ）デバイス、インテリジェント・シリアル・インターフェース・カード、またはデータの暗号化／解読のためのデバイスというような特定用途周辺機器を含む。

【００２４】
ここで説明する概略的な発明の概念は、前述の周辺ハードウエア・デバイスのいずれとでも、計算タスクをオフロードするために使用可能であるが、本発明の説明は、現時点において好適な一実施形態の例に関して行なう。ここでは、計算タスクを、図１に示すＮＩＣ５３のようなネットワーク通信デバイスにオフロードする。即ち、例示の実施形態は、Ｍｉｃｒｏｓｏｆｔ　Ｃｏｒｐｏｒａｔｉｏｎ（マイクロソフト社）から入手可能なＷｉｎｄｏｗｓ　ＮＴオペレーティング・システムのネットワーキング環境およびアーキテクチャにおいて実施したものとして説明する。とは言え、Ｗｉｎｄｏｗｓ　ＮＴの概念および用語を具体的に参照するものの、殆どではなくとも多くのオペレーティング・システムおよびネットワーキング・アーキテクチャが、本発明の環境に関連する類似性を共有することを当業者は認めよう。

【００２５】
本発明の内容をより良く理解するために、次に図２を参照する。図２は、Ｗｉｎｄｏｗｓ　ＮＴネットワーキング・モデルを構成するコンポーネントの一部の簡略図を示す。例示の目的のために、種々のＷｉｎｄｏｗｓ　ＮＴコンポーネントに対応するＯＳＩレイヤも示す。ＯＳＩモデルにおける物理レイヤに対応するボトム・レイヤに、実際のＮＩＣ（ネットワーク・カード、またはネットワーク・アダプタと呼ぶこともある）１００～１０４が常駐している。ＮＩＣは、物理媒体（ネットワーク・ケーブル）との物理的相互接続、および特定のネットワーク・トポロジにしたがって、上位レイヤ全てが発生するデータを搬送する信号の伝送を行なうハードウエア・デバイスである。先に注記したように、多くのＮＩＣは専用プロセッサおよびメモリを搭載しており、精巧な計算タスクも実行可能である。そのタスクには、ホスト・プロセッサＣＰＵが通常であれば処理するようなタスクが含まれる。ＮＩＣは、物理的に、コンピュータ内のスロットに配置するプリント回路ボード・カードとして、マザー・ボード上のコンピュータ・シャーシ内に配置した専用チッ

プとして、またはその他のいずれかの適当な方法で実施することができる。
【００２６】
各ＮＩＣは、双方向ライン１０８～１１２によって概略的に表すように、対応するネットワーク・ドライバ１１６～１２０を介して論理的にＷｉｎｄｏｗｓ　ＮＴ　ネットワーキング・モデルと相互接続している。ネットワーク・ドライバは、ネットワーク・モデルのＭＡＣサブレイヤ内に常駐し、対応するＮＩＣを介してＷｉｎｄｏｗｓ　ＮＴを物理ネットワーク・チャネルにリンクする。各ドライバは、典型的に対応するＮＩＣのベンダが供給するソフトウエア・コンポーネントとして実施され、その対応するネットワーク接続を通じてパケットを送受信する役割、およびオペレーティング・システムの代わりにＮＩＣを管理する役割を果たす。また、各ドライバは、対応するＮＩＣ上でＩ／Ｏを開始し、これらから割込を受け取り、上流側のプロトコル・ドライバにコールし、アウトバウンド・データ転送の完了をこれらに通知する。また、デバイス・ドライバは、対応するＮＩＣの追加処理能力全てに対する呼び出し、制御および／または監視を行なう役割も担う。
【００２７】
環境によっては、ＴＣＰ／ＩＰまたはＸＮＳのような単一のネットワーク特定プロトコルを実施するように、ドライバ・コンポーネントに書き込む。ここに記述し特許を請求する本発明の基本的発明は、このような環境に適用可能である。しかしながら、例示の目的のために、本発明は、Ｗｉｎｄｏｗｓ　ＮＴネットワーク・アーキテクチャに関連付けて説明することにする。この場合、ネットワーク・ドライバ・インターフェース仕様（ＮＤＩＳ）と呼ばれるインターフェースおよび環境が与えられる。ＮＤＩＳインターフェースは、機能的には、図２の１２６に示す通りである。ＮＤＩＳは、ネットワーク・ドライバ１１６～１２０の各々を、種々のトランスポート・プロトコル（その例を１２８～１３４で示す）の詳細から、およびその逆に、遮蔽する。即ち、ＮＤＩＳは、１つ以上のＮＩＣドライバ（１１６～１２０）が１つまたは多数の下位ＮＩＣ（１００～１０４）、１つまたは多数の上位トランスポート・プロトコル・ドライバ、あるいはトランスポート（図２の１２８～１３４で表す）、およびオペレーティング・システムと通信するためのインターフェースを記述する。
【００２８】
本質的に、ＮＤＩＳはＮＩＣドライバの開発のために、完全に抽象化した環境を定義する。したがって、ＮＩＣドライバが実行すべき各外部機能毎に、ＮＩＣハードウエア割込の登録および代行受信（intercept）からトランスポート・プロトコル・ドライバとの通信、登録操作およびポートＩ／Ｏを介した下位のＮＩＣとの通信まで、機能を実行するためにＮＤＩＳ　ＡＰＩに頼ることができる。このレベルの抽象化およびその結果としての移植性を得るために、ＮＤＩＳは、ＮＤＩＳインターフェース・ライブラリ・ラッパ（NDIS interface Library Wrapper）（図示せず）と呼ぶ、エクスポート・ライブラリを用いる。ＮＩＣドライバおよびプロトコル・ドライバ間、ＮＩＣドライバおよびオペレーティング・システム間、ならびにＮＩＣドライバおよびＮＩＣ間の相互作用は全て、ラッパ関数へのコールによって実行する。したがって、Ｗｉｎｄｏｗｓ　ＮＴのトランスポート特定ドライバに書き込む代わりに、ネットワーク・ベンダは、ＮＤＩＳインターフェースを、単一ネットワーク・ドライバの最上位レイヤとして与える。こうすることにより、いずれのプロトコル・ドライバも、このインターフェースをコールすることにより、そのネットワーク要求をネットワーク・カードに宛てて送ることが可能となる。したがって、ユーザは、ＴＣＰ／ＩＰネットワークおよびＤＬＣ（または、ＮＷＬＩＮＫ、またはＤＥＣｎｅｔ、ＶＩＮＥＳ、ＮｅｔＢＥＵＩ等）ネットワークを通じて、１つのネットワーク・カードおよび単一のネットワーク・ドライバを用いて通信することができる。
【００２９】
ネットワークおよびデータ・リンク・レイヤは、一例として図２の１２８～１３４で示す、トランスポート、プロトコルおよび関連するドライバである。Ｗｉｎｄｏｗｓ　ＮＴでは、トランスポート・プロトコル・ドライバは、トランスポート・ドライバ・インターフェース（ＴＤＩ）、または恐らくその上縁における別の特定用途インターフェースを実施

し、ネットワークのユーザにサービスを提供するソフトウエア・コンポーネントである。Ｗｉｎｄｏｗｓ　ＮＴでは、ＴＤＩは、機能ブロック１３８に示すリディレクタおよびサーバ機能のような、セッション・レイヤにおいて通信するネットワーキング・コンポーネントのために共通インターフェースを備える。公知のように、トランスポート・プロトコルは、ネットワークのためのデータ・オーガナイザとして作用し、本質的にどのようにデータを次の受信レイヤに提示し、それに応じてデータをパッケージ化するかを定義する。これらは、パケット（Windows NT関連ではＮＤＩＳパケットとも呼ぶ）を割り当て、データを送出側アプリケーションからパケットにコピーし、ＮＤＩＳをコールすることによってパケットを下位デバイス・ドライバに送り、対応するＮＩＣを介してデータをネットワーク上に送出できるようにする。

【００３０】

尚、データ・パケットが種々のネットワーク・レイヤ、典型的に、ネットワーク・モデルのレイヤ３および４を通過する際に、追加の機能またはタスクもデータ・パケットに対して実行可能であることは認められよう。例えば、トランスポート・プロトコル・ドライバは、チェックサム値を計算し、これをパケットに添付する。一般に、この動作には、ネットワーク・パケットの送出側に対応するトランスポート・プロトコルが、パケットを構成するデータ・エレメントを全て加算することによって計算した数値を、それに添付する必要がある。次いで、パケットの受信側が、添付されているチェックサム数値をデータと比較することによって、データが移動中に変化しなかったことを確認する。

【００３１】

ＮＩＣハードウエアにおいて最適に実行可能な別の関連タスクに、データ・パケットに対するメッセージ・ダイジェストの計算がある。チェックサムと同様、メッセージ・ダイジェストは、パケット内のデータの完全性を保証するために用いられる。加えて、メッセージ・ダイジェストは、メッセージを送った側が意図した者であることを確かめることによって、データの認証を保証するために用いることもできる。メッセージ・ダイジェストの計算は非常にＣＰＵ集中的であり、ソフトウエアで実施するには不経済な機能である。

【００３２】

別の望ましい機能に、パケット内のデータの暗号化がある。暗号化とは、パッケージ内のメッセージを変換することにより、パケットの無許可の読み手が、暗号化キーを予め知らなければ、メッセージの内容を実際に見ることができないようにする暗号化プロセスのことを言う。勿論、暗号アルゴリズムも非常にＣＰＵおよびメモリ集中的となりがちであり、ソフトウエアで実行すると、法外な程不経済となる可能性がある。

【００３３】

データ・パケット上で実行可能な別のタスクに、ＴＣＰセグメント化がある。公知のように、ＴＣＰは大きなデータ・パケットを、下位ネットワークが許可する最大データ・サイズに合ったセグメントに区分する。例えば、イーサネットはネットワーク上で最大１５１４バイトのパケットを許可する。したがって、ＴＣＰが例えば６４Ｋバイトを送らなければならない場合、データを１５１４バイトのセグメントに分解しなければならない。

【００３４】

これらおよびその他の機能は、典型的に、種々のネットワーク・レイヤに常駐するソフトウエア・コンポーネントにおいて、コンピュータのＣＰＵ２０が実行するので、かなりのコンピュータ資源を利用し、コンピュータ・システムの性能の全体的な低下を招く可能性がある。したがって、これらまたはその他の同様のタスクをオフロードし、対応するＮＩＣにおいてこれらを代わりに実行するようにすれば、コンピュータ・システムの全体的な速度および効率を大幅に向上させることが可能となる。

【００３５】

先に注記したように、Ｗｉｎｄｏｗｓ　ＮＴまたは同様のレイヤ状ネットワーキング・モデルにおけるデータ伝送の基本単位はデータ・パケットである。Ｗｉｎｄｏｗｓ　ＮＴ環境では、データ・パケットのことをＮＤＩＳパケットと呼ぶ。各パケットは、スタックの最上位（即ち、ＩＳＯスタックにおけるレイヤ５）から最下位のソフトウエア・レイヤ（

即ち、ＩＳＯスタックにおけるレイヤ２）まで移動する。したがって、パケットは、データの送信および受信の間にレイヤを通過する際、各レベルを通じて共通のデータ構造を定義する。一例として、図３は、パケットが各レイヤを通って、イーサネットＮＩＣとして１００で示すＮＩＣまで達する際に、パケットが辿る経路を示す。先に注記したように、トランスポート・ドライバ１２８は、送出側アプリケーションからデータを受け取り、下位プロトコルと一貫性のあるパケットにこれをパッケージ化し、次いでＮＤＩＳインターフェース１２６を介して下位のデバイス・ドライバ１１６にパケットを転送する。加えて、トランスポート・プロトコルは、パケットに対して他の機能（例えば、チェックサムの計算等）も行なうことができる。あるいは、他の機能コンポーネントが、図３に示すＩＰセキュリティ機能１４４（例えば、暗号化および／またはパッケージ・ダイジェストの計算）のように、パケットに追加の機能を実行するネットワーク・レイヤまたはデータ・リンク・レイヤに常駐することも可能である。

【００３６】

本発明の好適な実施形態の１つでは、データ・パケット１４２は、計算タスクを、ＮＩＣハードウエア１００のような周辺デバイスにオフロードするための手段である。例えば、図３において、アプリケーション・データ１４０は、ネットワーク・モデルの上位レイヤから、ＴＣＰ／ＩＰ１２８のような適切なトランスポート・プロトコル・ドライバに渡される。ドライバはデータを適切なデータ・パケット１４２にパッケージ化し直す。次いで、この特定のデータ・パケット１４２にどの追加機能を実行するかに応じて、パケット拡張部と呼ぶ、既定のデータ構造をデータ・パケットに添付する機能コンポーネントを含ませる。以下で更に詳細に説明するが、パケット拡張部の内容は、データ・パケットがＮＩＣ１００に到達したときに、どのタスクまたは複数のタスクをこのデータ・パケットに対して実行するかを示す。データ・パケット１４２がネットワーク・ドライバ１１６に到達すると、ネットワーク・ドライバ１１６がこのパケット拡張部の内容を問い合わせ、どのタスク（複数のタスク）をＮＩＣ１００が実行するのかを確認する。次いで、ドライバ１１６は、ＮＩＣ上のハードウエアを制御／操作し、特許拡張部の内容を通じて要求されたいかなる機能タスクをも実行する。

【００３７】

例えば、図３において、データ・パケット１４２をソフトウエア・コンポーネント１４４に渡す。ソフトウエア・コンポーネント１４４は、別個に実施することも、あるいはトランスポート・プロトコル・ドライバ自体の一部として実施することも可能であり、パケット拡張部をパケット１４２に添付する。オフロードする特定のタスクに応じて、パケット拡張部にデータを含ませる。例えば、ＩＰセキュリティ機能を実施する場合、ＮＩＣが指定された暗号化キーにしたがってデータ・パケットを暗号化することを示すデータを含ませる。勿論、ソフトウエア・コンポーネント１４４は、既定のデータを添付し、先に説明したような多数の機能のいずれか１つを、ネットワーク・レイヤ内に常駐するソフトウエア・コンポーネントによって実行する代わりに、ハードウエア・レベルで実行するようにすることができる。デバイス・ドライバ１１６は、パケット拡張部から情報を抽出し、次いでＮＩＣ１００において指定されたタスク（複数のタスク）を呼び出す。

【００３８】

図４は、パケット１４２の概略構造の現時点における好適な実施形態の１つを示す。パケット１４２は、Ｗｉｎｄｏｗｓ　ＮＴ環境において用いられる正確なネットワーク環境に応じて、いかなるフォーマットでも可能であるが、パケットはＮＤＩＳにしたがってフォーマット化してあり、パケット記述子、協働するデバイス・ドライバ（ドライバ群）およびプロトコル・ドライバ（ドライバ群）が意味を規定するフラグ、当該パケットに関連する帯域外データ（ＯＯＢ）の記憶領域、パケット長に関する情報、ならびにパケットのデータ内容に関係するメモリ位置に対するポインタというような情報を含む。

【００３９】

図４は、更に、タスクのオフロードを識別するためにＮＤＩＳデータ・パケットに添付する、追加のデータ構造フィールド、即ち、パケット拡張部１５０も示す。先に説明したよ

うに、宛先のＮＩＣにオフロードしている特定のタスクまたは複数のタスクの識別に必要な情報を収容するデータ構造を定義するのは、このパケット拡張部１５０である。好適な実施形態では、各タスク・オフロード・タイプ（例えば、チェックサム、暗号化／解読、等）毎に、既定のデータ・フィールドをパケット拡張部１５０内に含ませる。このデータ・フィールドは、単に、制御フラグまたは複数のフラグの形式とすることができる。制御フラグは、単に、特定の機能を実行する（チェックサム等）ことを示すに過ぎない。または、情報は、タスクをどのように遂行すべきかについて更に定義するデータ構造へのポインタの形式とすることも可能である。例えば、図４に示す例では、パケット拡張部１５０は、ＮＩＣは１５２で示すチェックサム動作を実行することを示すフラグを収容する。このタイプのタスクでは、パケット拡張部は、受信局におけるＮＩＣが、送出側のＮＩＣが計算したチェックサムの有効性をチェックするようにも設定する。

【００４０】

限定ではなく一例として、ＮＩＣがチェックサム動作を実行することを意味するパケット拡張部データ構造の好適な実施形態は、以下の構造を有する。

【００４１】

【表１】

```
typedef struct _NDIS_TCP_IP_CHECKSUM_PACKET_INFO
        {
        union
        {
                struct
                {
                        ULONG   NdisPacketChecksumV4:1;
                        ULONG   NdisPacketChecksumV6:1;
                }
                        Transmit;
                struct
                {
                        ULONG   NdisPacketTcpChecksumFailed:1;
                        ULONG   NdisPacketUdpChecksumFailed:1;
                        ULONG   NdisPacketIpChecksumFailed:1;
                        ULONG   NdisPacketTcpChecksumSucceeded:1;
                        ULONG   NdisPacketUdpChecksumSucceeded:1;
                        ULONG   NdisPacketIpChecksumSucceeded:1;
                        ULONG   NdisPacketLoopback:1;
                }
                        Receive;
                ULONG       Value;
           };
}
        NDIS_TCP_IP_CHECKSUM_PACKET_INFO,
        *PNDIS_TCP_IP_CHECKSUM_PACKET_INFO;
```

この特定のパケット拡張部データ構造では、変数NdisPacketChecksumV4およびNdisPacketChecksumV6双方をセットしない場合、デバイス・ドライバは、データ・パケットに対してチェックサムを全く行なうことなく、これを送る。

【００４２】

また、図４は、パケット・データの暗号化および／またはメッセージ・ダイジェストの計算に関連して実行する、セキュリティ機能１５４も、送出側ＮＩＣが実行すべきことを指定する。このタイプのタスクでは、フィールド１５４は、データ構造を収容するメモリ位置に対するポインタを収容することが好ましい。一方、このデータ構造は、暗号化および／またはメッセージ・ダイジェスト機能の実行に関連する情報を収容する。状況によって

は、関連データを有するメモリ位置に対するポインタを含ませることは、パケット拡張部自体の中に実際のデータを格納するよりも有効な場合がある。

【００４３】

このような利点の１つを図５に示す。これは、好適なプログラム・シーケンスの一例を示し、多数の連続するデータ・パケットに同じタイプの暗号化またはダイジェスト計算動作を実行しなければならない状況に対応する。プログラム・ステップ３０２において開始した後、プログラム・ステップ３０４は、現パケットがパケット・シーケンス内の最初のパケットであるか否かについて判定を行なう。そうである場合、この最初のパケットに、続くパケットに対する動作にも同様に用いる情報即ちコンテクストを与える。例えば、最初のパケットは、用いる特定の暗号化キーを明記するパケット拡張部を有する。この値即ちコンテクストは、別個のメモリ位置即ちハンドルに格納する。連続するパケットは、この情報を含む必要がなく、例えば、プログラム・ステップ３０８に示すように、暗号化キー（またはその他のコンテクスト情報）が格納されているメモリ位置に対するポインタのみを有する。この手法は、パケットのシーケンスにおける後続のデータ・パケット全体のサイズを縮小し、更にタスク・オフロード方法の効率および移植性を高める。

【００４４】

好適な実施形態では、パケット拡張部１５０内に収容する情報は、パケット１４２を送る特定のデバイス・ドライバによる問い合わせを受ける。例示の実施形態において記載しているＷｉｎｄｏｗｓ　ＮＴ環境では、このタイプの機能は、適切なＮＤＩＳ関数をコールすることによって実行することが好ましい。例えば、パケットに対するパケット拡張部１５０のメモリ位置へのポインタを戻す既定のＮＤＩＳ関数を実行することができる。このときには、デバイス・ドライバ・ソフトウエアは、どのタスクを実行するのか識別し、オフロードするタスク（複数のタスク）に応じて、適切にドライバの対応するＮＩＣハードウエアを動作させ／操作する。

【００４５】

実際のデータ・パケットを利用して計算タスクをコンピュータ・プロセッサからハードウエア周辺機器にオフロードすることは、多くの理由から有利である。例えば、トランスポート・ドライバは、パケット毎に、周辺機器の能力を利用することができる。これによって、タスクを動的にダウンロードすることができ、周辺機器の能力を必要に応じて用いることができる。したがって、特定の時点において、コンピュータ・システムの処理オーバーヘッドが低い場合、あるタスクを従来通りコンピュータ・プロセッサ上で実行することが望ましいこともある。あるいは、ＣＰＵが他の計算タスクを多く負担する場合、単にデータ・パケットに必要なパケット拡張部を添付することによって、タスクを周辺デバイスにオフロードすることができる。

【００４６】

他の利点は、単一のパケットによって多数のタスクをオフロードできることであり、本質的に、多数の動作を一度に「バッチ処理」する。例えば、コンピュータ・プロセッサがチェックサム動作または暗号化動作を実行する場合、その動作、即ち、チェックサムの計算またはパケット・データの暗号化を完了することができるようにするには、その前にデータ・フィールド全体をメモリ位置にロードしなければならない。更に、レイヤ状ネットワーキング・モデルのため、一度に実行できる動作は１つのみであり、データをメモリに多数回コピーする必要がある。しかしながら、パケット単位の手法では、多数のタスクを１つのパケットにオフロードすることができる。したがって、ハードウエア周辺機器は、ハードウエアの能力に応じて、データ上での単一のパスにおいて２つ以上の動作を実行することができ、コンピュータ・システムのスループットおよび効率を大幅に向上させることができる。

【００４７】

前述の方式は、特定のＮＩＣにオフロードするタスクを指定する機能を用いる場合特に有用であるが、パケット単位の情報転送は他の方法でも同様に使用可能であることは認められよう。例えば、特定のＮＩＣが所定の時点にパケットの配信をスケジュールすることが

できる場合、パケット拡張データ構造は、ＮＩＣハードウエアがパケットをどのようにおよび／またはいつ送るかを識別する情報を渡すためにも使用することができる。

【００４８】

本発明の好適な実施形態では、トランスポート・プロトコル・ドライバがパケット拡張部をデータ・パケットに添付し、特定のタスクをＮＩＣにオフロードする前に、２つの追加の機能を最初に実行する。異なるタイプのハードウエア周辺機器が数多くあり、各々が異なる処理能力を有する限りにおいて、本発明の実施形態は、トランスポート・ドライバが最初にコンピュータ・システムに接続してある周辺機器のタスク・オフロード能力を問い合わせることができるようにする手段を備えることが好ましい。一旦これらの機能を確認し終えれば、トランスポート・プロトコル・ドライバは、対象のタスクを設定またはイネーブルすることができる。一旦イネーブルすれば、前述のようにパケット毎に指定のタスクを続いて利用することができる。

【００４９】

図６は、ＮＩＣのような周辺機器のタスク・オフロード機能を問い合わせるこの機能を実施し、次いで必要と思われるタスクを設定／イネーブルする現時点において好適な１つのプログラム・ステップ・セットを示す。好ましくは、図示のプログラム・ステップは、トランスポート・プロトコル・ドライバ内に統合化したソフトウエア・コンポーネントまたはモジュールとして実施する。更に、Ｗｉｎｄｏｗｓ　ＮＴ環境において実施する場合、機能的動作およびドライブ間通信の多くは、ＮＤＩＳインターフェースを介して実行することが好ましい。勿論、異なるオペレーティング・システム環境において実施する場合、または非ネットワーク・タイプの周辺機器と共に実施する場合、図示のプログラム・ステップはそれに応じて変更しなければならない。

【００５０】

実行可能なプログラム命令は、コンピュータ・システムのＣＰＵ（例えば、図１の演算装置２１）が実行する。プログラム・ステップ２０２において開始し、当該コンピュータ・システムに接続してある周辺機器（複数の周辺機器）のタスク・オフロード能力について問い合わせを行なう。例示の環境では、周辺機器（複数の周辺機器）は、システムに付属するＮＩＣ（複数のＮＩＣ）である。好ましくは、各ＮＩＣデバイス・ドライバ（図２の１１６～１２０）には、既定のタスク・オフロード・バッファ位置（複数の位置）が関連付けられており、その各々が当該デバイス・ドライバおよびその対応するＮＩＣのタスク・オフロード能力を収容する。少なくとも、タスク・オフロード・バッファは、ＮＩＣおよびそのデバイス・ドライバが対応する個々のタスク（複数のタスク）を識別し、更に、対応する個々のタスク毎に特定的なあらゆる情報を含む。好適な実施形態では、デバイス・ドライバのタスク・オフロード・バッファの内容は、ＮＤＩＳ関数コールによって検索する。

【００５１】

限定ではなく一例として、ドライバのタスク・オフロード・バッファの好適な実施形態は以下の構造を有する。

【００５２】

【表２】

```
typedef                                 struct
_NDIS_TASK_OFFLOAD
{
ULONG                   Size;
NDIS_TASK               Task;
ULONG                   OffsetNextTask;
ULONG                   TaskBufferLength;
UCHAR                   TaskBuffer[1];
}
NDIS_TASK_OFFLOAD,
*PNDIS_TASK_OFFLOAD;
```

ここで、

【００５３】

【表３】

| | |
|---|---|
| Size | この変数は、この構造のサイズにセットし、使用する構造のバージョンを判定するために用いる。 |
| Task | これは、構造において定義したタスク・オフロードのタイプを示す。 |
| OffsetNextTask | 次のタスク・オフロード・バッファに対するオフセット。これが、このドライバ／ＮＩＣが対応する最後のタスクである場合、値は０である。 |
| TaskBufferLength | この構造に従うタスク・オフロード・バッファの長さ。 |
| TaskBuffer | これは、この構造で定義した特定のタスク・オフロードを記述するために必要な情報／データを収容するタスク特定バッファである。 |

一旦各周辺機器の個々のタスク・オフロード能力を確認したなら、プロセッサ２１は、図６のプログラム・ステップ２０４に示す機能に対応するコンピュータ実行可能命令に進む。ここで、各周辺ＮＩＣが対応する個々のタスク・オフロード能力のいずれかが、トランスポートに該当するものであるか否かについて判定を行なう。当該周辺機器に対応するタスクがない場合、または対応するタスクが当該トランスポートには有用ではない場合、プロセッサは直接プログラム・ステップ２０８に進み、この処理の特定のラインを停止し、トランスポートはその通常動作に進む。あるいは、識別したタスクの１つ以上が該当する場合、プログラム・ステップ２０６を実行する。このステップにおいて、実行可能命令を実行し、プロトコル・ドライバが望む特定のタスク・オフロード能力をイネーブルにセッ

トする。これは、デバイス・ドライバおよびその対応するＮＩＣに、どのタイプのタスク・オフロードをパケット毎に予期する可能性があるのかについて知らせる。好適な実施形態では、タスク・オフロード能力をセットするには、データ構造内に適切なデータ値をセットし、次いで、ＮＤＩＳインターフェース関数コールによって、対応するデバイス・ドライバに受け渡す。例えば、個々のタスクについて、当該タスクのタスク・オフロード・バッファ構造内の適切なビットをトランスポートによってセットすることにより、当該ドライバ／ＮＩＣのタスクをイネーブルすることができる。こうして、トランスポートは、後に使用したくなる可能性がある各ＮＩＣ／ＮＩＣドライバのタスク・オフロード能力をいくつでもイネーブルすることができる。

【００５４】

一旦所望のオフロード能力を各デバイス・ドライバおよびその対応するＮＩＣ毎にイネーブルしたなら、コンピュータ・システム・プロセッサ２１はプログラム・ステップ２０８に進み、この特定の機能に対する処理が終了する。この時点において、トランスポート・ドライバは、前述のように各パケット毎に、イネーブルしたタスク・オフロード能力の各々を利用することができる。

【００５５】

要約すれば、本発明の実施形態は、現在従来技術において利用可能なものと比較して、格別な利点を与える。コンピュータ・システムのプロセッサおよびメモリにおいて本来実行すべき具体的な処理タスクを、代わりに、当該コンピュータに接続してある特定の周辺デバイスまたは複数の周辺デバイスにダウンロードする。次いで、周辺機器によって計算タスクを実行することにより、コンピュータ・システムの資源を他の計算タスクのために保存する。特に、オフロードした処理タスクがＣＰＵおよび／またはメモリ集中的な場合、このタスク・オフロード方式は、コンピュータ・システム全体の計算効率を劇的に向上させる。開示したコンピュータ・タスク・オフロード方法は、動的に要求に応じてタスクをオフロードすることを可能にする手段を備えるという利点がある。したがって、プロセッサは、他の計算タスクの処理に忙しく、プロセッサのオーバーヘッドが高いときに、タスクをオフロードすることができる。逆に、コンピュータ・システムの処理資源に対する要求が低くなった場合には、代わりにプロセッサはタスクをそれ自体で実行すればよい。加えて、特定の周辺機器に多数のタスクをバッチでオフロードすることができる。多くの場合、周辺機器は、このような多数のタスクを、コンピュータ・プロセッサよりははるかに効率的に実行するように最適化されているので、システム全体の効率が更に改善する結果となる。最後に、本発明の方法は、周辺機器の特定のタスク・オフロード能力について問い合わせ、その後に選択的にイネーブルすることを可能にする処理方式を提供する。このように、コンピュータ・システムは、その種々の周辺デバイスの能力を容易に識別し、後に必要となる処理能力またはその可能性がある処理能力のみを利用することができる。

【００５６】

本発明は、その精神または本質的な特徴から逸脱することなく、他の特定形態でも具体化が可能である。前述の実施形態は、あらゆる観点においても、例示であり限定として解釈すべきではない。したがって、本発明の範囲は、前述の説明ではなく、添付した特許請求の範囲によって示すこととする。特許請求の範囲の均等の意味および範囲に該当するあらゆる変更は、その範囲に含まれるものとする。

【図面の簡単な説明】

【図１】　本発明と共に使用可能な典型的なコンピュータ・システムおよび付属周辺デバイスの一例を示す図である。

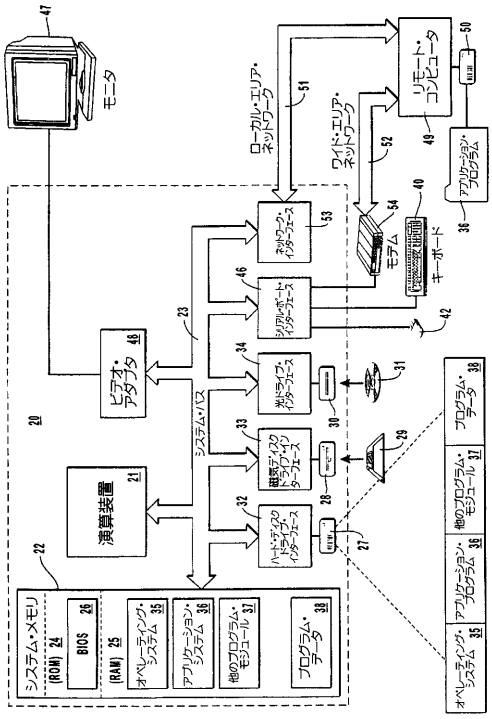
【図２】　レイヤ状ネットワーク・アーキテクチャ内にある機能コンポーネントの一部を示す図である。

【図３】　本発明の現時点における好適な実施形態の１つに応じたプログラム・コンポーネントによるデータ・パケットのフローを示す機能ブロック図である。

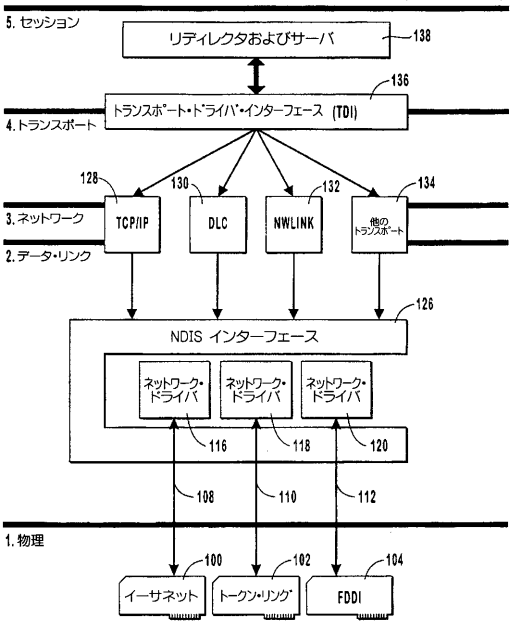
【図４】　データ・パケットおよびパケット拡張の現時点における好適な実施形態の１つを示す図である。

【図５】　パケット毎にタスクをオフロードする際に用いるプログラム・ステップの現時点における好適な実施形態の１つを示すフロー・チャートである。
【図６】　周辺デバイスのタスク・オフロード能力を問い合わせそして設定するために用いるプログラム・ステップの現時点における好適な実施形態の１つを示すフロー・チャートである。

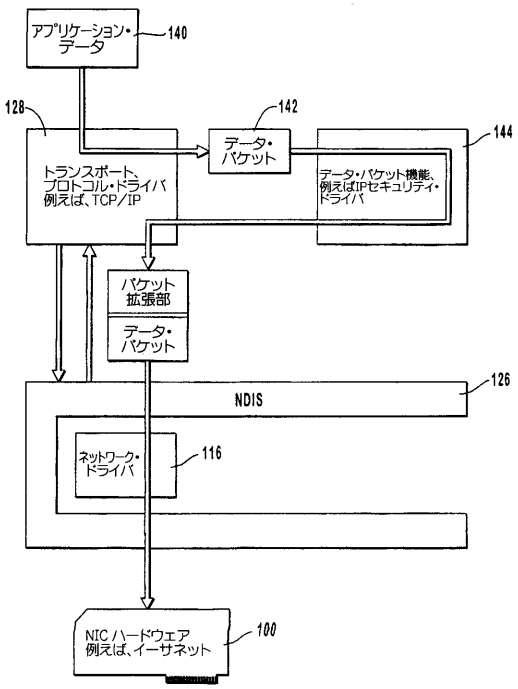
【図１】



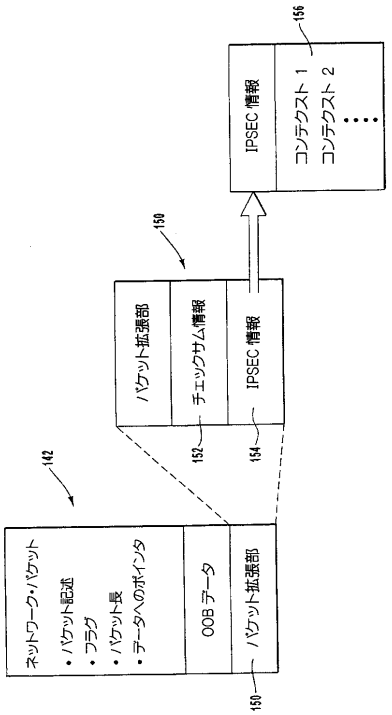

【図２】



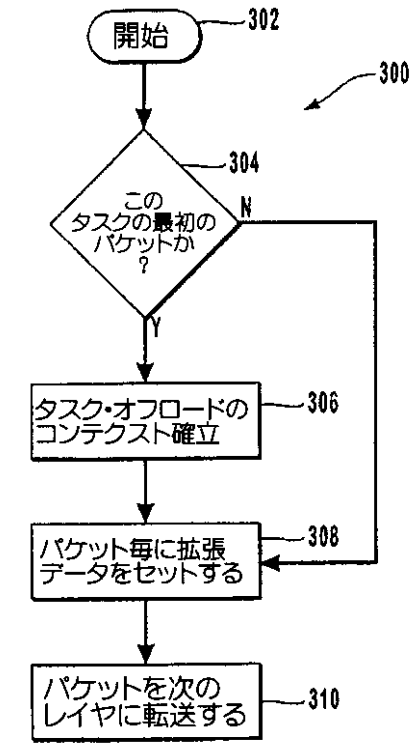
【図３】


アプリケーション・データ
140
128
142
データ・パケット
144
トランスポート、プロトコル・ドライバ 例えば、TCP/IP
データ・パケット機能、例えばIPセキュリティ・ドライバ
パケット拡張部
データ・パケット
NDIS
126
ネットワーク・ドライバ
116
NIC ハードウェア 例えば、イーサネット
100

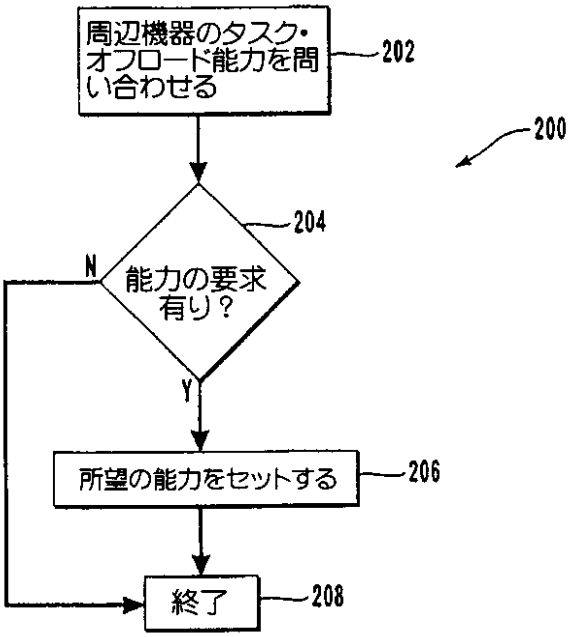
【図４】


156
IPSEC 情報
コンテクスト 1
コンテクスト 2
....
150
パケット拡張部
チェックサム情報
IPSEC 情報
152
154
142
ネットワーク・パケット
• パケット記述
• フラグ
• パケット長
• データへのポインタ
OOB データ
パケット拡張部
150

【図５】


開始
302
300
304
この タスクの最初の パケットか ?
N
Y
タスク・オフロードの コンテクスト確立
306
パケット毎に拡張 データをセットする
308
パケットを次の レイヤに転送する
310

【図６】


周辺機器のタスク・オフロード能力を問い合わせる
202
200
204
N
能力の要求 有り?
Y
所望の能力をセットする
206
終了
208

フロントページの続き

(72)発明者　アナンド，サンジェイ
アメリカ合衆国ワシントン州９８０５２，レッドモンド，オールド・レッドモンド・ロード　７００１，アパートメント　エヌ１５５
(72)発明者　ブランドン，カイル
アメリカ合衆国ウィスコンシン州５３７０３，マディソン，ウィリアムソン・ストリート　７５４
(72)発明者　スリニヴァス，ンク
アメリカ合衆国ワシントン州９８０２９，イサクア，サウス・イースト・フォーティセカンド・ストリート　２５０４１
(72)発明者　ハイダー，ジャミール
アメリカ合衆国ワシントン州９８０５３，レッドモンド，ノース・イースト・シックスティーンス・プレイス　２３２９２

審査官　鈴木　修治

(56)参考文献　米国特許第０５６３４０７０（ＵＳ，Ａ）
特開平０９－２３４８５３（ＪＰ，Ａ）
特許第２９５０３６６（ＪＰ，Ｂ２）
特開平１０－１２４４６８（ＪＰ，Ａ）
特開平０６－０１９８５５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/46-9/54
G06F 15/16-15/177
G06F 13/10-13/14