(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5982002号
(P5982002)
(45)発行日　平成28年8月31日(2016.8.31)　(24)登録日　平成28年8月5日(2016.8.5)

(51)Int.Cl.　F I
H04L 12/859 (2013.01)　H04L 12/859
H04L 12/911 (2013.01)　H04L 12/911

請求項の数 24　(全 33 頁)

(21)出願番号　特願2014-538989(P2014-538989)
(86)(22)出願日　平成24年10月25日(2012.10.25)
(65)公表番号　特表2014-531179(P2014-531179A)
(43)公表日　平成26年11月20日(2014.11.20)
(86)国際出願番号　PCT/US2012/061844
(87)国際公開番号　WO2013/063218
(87)国際公開日　平成25年5月2日(2013.5.2)
審査請求日　平成27年9月28日(2015.9.28)
(31)優先権主張番号　61/551,203
(32)優先日　平成23年10月25日(2011.10.25)
(33)優先権主張国　米国(US)

(73)特許権者　507334303
フォースウォール・メディア・インコーポレーテッド
アメリカ合衆国ヴァージニア州２０１６６，ダレス，ホースシュー・ドライブ　４５９２５，スイート　１００
(74)代理人　100065950
弁理士　土屋　勝
(72)発明者　スロサウバー・ルイス・ピー
アメリカ合衆国ヴァージニア州２０１７６，リーズバーグ，チムニー・コート，エヌイー７０２

審査官　宮島　郁美

最終頁に続く

(54)【発明の名称】トラフィックスケジューリングを使用したネットワーク帯域幅調整

(57)【特許請求の範囲】
【請求項１】
集中型のトラフィックマネージメントサーバーが少なくとも一つのクライアントノードによって送信される着信パケットのサンプルを連続的にモニターするように、前記トラフィックマネージメントサーバーを使用してネットワーク経由でデータトラフィックをモニターすることと、
前記トラフィックマネージメントサーバーが着信パケットの前記サンプルの頻度、変化率およびコンテンツに基づいて負荷情報を演算するように、ネットワーク負荷を予測することと、
前記トラフィックマネージメントサーバーと前記少なくとも一つのクライアントノードとがデータパケットの今後の送信用にスケジュールされた時間を決定するメッセージを交換するように、ネットワークトラフィックをスケジューリングすることと、
前記少なくとも一つのクライアントノードが前記スケジュールされた時間に前記ネットワーク経由で前記データパケットを送信するように、データを送信することと、
を含むコミュニケーションネットワークの帯域幅を調整する方法。
【請求項２】
必要ならば、さらに、前記データパケットを再送信することを含み、もし、前記トラフィックマネージメントサーバーからアクノリッジメントが受信されないならば、前記少なくとも一つのノードが前記データパケットを再送信することを特徴とする請求項１に記載の方法。

【請求項３】
前記ネットワークが、インタラクティブなデジタルケーブルテレビネットワークであることを特徴とする請求項１に記載の方法。
【請求項４】
前記ケーブルテレビネットワークが、ユーザデータグラムプロトコル（ＵＤＰ）上のハイパーテキストトランスファープロトコル（ＨＴＴＰ）を使用して、複数のセットトップボックスと、アウトオブバンドチャネルを経由にてコミュニケーションするエンハンスドテレビジョンプラットフォームサーバーを含むことを特徴とする請求項３に記載の方法。
【請求項５】
前記セットトップボックスが、ＥＢＩＦユーザエージェント上で作動しているエンハンスドテレビジョンバイナリーインターチェンジフォーマット（ＥＢＩＦ）を使用して実施されるインタラクティブなアプリケーションを起動することを特徴とする請求項４に記載の方法。
【請求項６】
前記トラフィックが、インタラクティブなアプリケーションコミュニケーション、ボーティングとポーリングデータ、テレビ利用状況データ、ユーザ嗜好と統計トラフィック、および、Ｔコマース情報の内の少なくとも一つを含むことを特徴とする請求項１に記載の方法。
【請求項７】
前記スケジューリングすることと前記送信することが、
前記トラフィックマネージメントサーバー上のネットワークトラフィック状況に応じて送信・確率の値を設定することと、
前記送信・確率の値およびクロック同期情報を、前記トラフィックマネージメントサーバーから前記少なくとも一つのクライアントノードへ送信することと、
前記送信・確率の値および同期情報を、前記少なくとも一つのクライアントノードにおいて、ランダム化したデータ送信タイムを演算するために利用することと、
前記演算されるランダム化したデータ送信タイムに、前記少なくとも一つのクライアントノードにより、データパケットを送信することと、
前記トラフィックマネージメントサーバーによって前記少なくとも一つのクライアントノードへアクノリッジメントパケットを発生して送信し、そこでは、前記アクノリッジメントが前回の送信の成功を示して次回の送信用の新しい送信・確率の値を提供していることと、
もしアクノリッジメントパケットが受信されなかったならば、新しいデータ送信タイムに、前記少なくとも一つのクライアントノードよって前記データパケットを再送信することと、
を含むことを特徴とする請求項１に記載の方法。
【請求項８】
前記送信・確率の値が、タイムウィンドウに反比例しており、前記次回のパケットが送信されるべきで、前記ランダム化したデータ送信タイムが前記ウィンドウ内でランダムタイムであり、前記アクノリッジメントパケットとデータパケットの各々が、前記ウィンドウ内で、ディスクリートなタイムスロット境界上で送信されることを特徴とする請求項７に記載の方法。
【請求項９】
前記送信・確率の値が、伝送エラーおよびアクノリッジメントタイムアウトなしで、時間の経過とともに、前記少なくとも一つのクライアントノードにて、漸進的に増加されることを特徴とする請求項７に記載の方法。
【請求項１０】
前記送信確率が、送信エラーとアクノリッジメントタイムアウトの内の少なくとも一つに応じて減少されることを特徴とする請求項７に記載の方法。
【請求項１１】

前記トラフィックマネージメントサーバーから送信・確率の値およびクロック同期情報を前記送信することが、前記少なくとも一つのクライアントノードへの周期的なブロードキャストを経由して行われることを特徴とする請求項７に記載の方法。

【請求項１２】

前記トラフィックマネージメントサーバーから送信・確率の値およびクロック同期情報を前記送信することが、少なくとも一つの個別にアドレスされたクライアントノードへのナローキャストを経由して行われることを特徴とする請求項７に記載の方法。

【請求項１３】

前記トラフィックマネージメントサーバーから送信・確率の値およびクロック同期情報を前記送信することが、前記少なくとも一つのクライアントノードが前記トラフィックマネージメントサーバーへ前記データパケットを送信する権利をリクエストするプローブデータパケットを送信した後に、生じることを特徴とする請求項７に記載の方法。

【請求項１４】

前記ネットワークが、インバンドケーブルテレビ、ＤＳＬ、または、モバイルブロードバンドネットワークであることを特徴とする請求項１に記載の方法。

【請求項１５】

前記ネットワークが、ユーザデータグラムプロトコル（ＵＤＰ）、または、送信制御プロトコル（ＴＣＰ）を利用することを特徴とする請求項１に記載の方法。

【請求項１６】

前記トラフィックマネージメントサーバーが、トラフィックマネージメント情報を具備する少なくとも一つのＵＤＰパケットを少なくとも一つのＴＣＰパケットへ変換するＵＤＰ→ＴＣＰリレーサーバーとしての役割を果たすことを特徴とする請求項１に記載の方法。

【請求項１７】

前記少なくとも一つのクライアントノード上で作動しているトラフィックマネージメントクライアントが、トラフィックをスケジューリングすることとデータを送信すること、および、クライアントアプリケーションをサポートしてデータを再送信することとをマネージすることを特徴とする請求項２に記載の方法。

【請求項１８】

前記トラフィックマネージメントクライアントが、前記クライアントアプリケーション用の標準ストリームソケットアプリケーションプログラミングインターフェースを模倣して、単一データグラムソケットまたはポートを使用しながら、複数の同時仮想ストリームソケットおよび接続を提供することを特徴とする請求項１７に記載の方法。

【請求項１９】

前記クライアントアプリケーションが前記データを送信する前の前記スケジュールされたデータ送信時間を待つように、データの前記送信および再送信が、同期方式で実施されることを特徴とする請求項１７に記載の方法。

【請求項２０】

前記ネットワークがデータ送信の準備ができた時に、コールバックが前記トラフィックマネージメントクライアントによって前記クライアントアプリケーションへ発行されるように、データの前記送信および再送信が、非同期方式で実施されることを特徴とする請求項１７に記載の方法。

【請求項２１】

前記スケジューリングし、送信し、再送信するステップが、前記少なくとも一つのクライアントノード上で作動している少なくとも一つのアプリケーションにさらされており、ユーザ側の動作を調整するために使用されることを特徴とする請求項２に記載の方法。

【請求項２２】

データの前記送信および再送信が、さらに、静的テーブルを使用して前記データを圧縮することを含み、そこでは、ストリング値を有する複数の標準ＨＴＴＰメッセージエレメントに、それぞれ、テーブルインデックスが割り当てられ、前記インデックスが前記スト

リング値の代わりに送信され、前記ストリング値が前記静的テーブルを利用することで受信時に再構築されることを特徴とする請求項２に記載の方法。
【請求項２３】
さらに、外部イベントに積極的に応答する帯域幅調整サーバーを含み、前記外部イベントが、時刻帯域幅負荷、時季帯域幅負荷および緊急ニュース速報の中から選択されることを特徴とする請求項１に記載の方法。
【請求項２４】
ネットワークと、
少なくとも一つのクライアントノードと、
集中型のトラフィックマネージメントサーバーと、
を含み、
前記トラフィックマネージメントサーバーは、
前記トラフィックマネージメントサーバーが前記少なくとも一つのクライアントノードによって送信される着信パケットのサンプルを連続的にモニターするように、前記ネットワーク経由でデータトラフィックをモニターし、
前記トラフィックマネージメントサーバーが着信パケットの前記サンプルの頻度、変化率およびコンテンツに基づいて負荷情報を演算するように、ネットワーク負荷を予測し、
前記トラフィックマネージメントサーバーと前記少なくとも一つのクライアントノードとがデータパケットの今後の送信用にスケジュールされた時間を決定するメッセージを交換するように、ネットワークトラフィックをスケジューリングする、
ように構成されている、
コミュニケーションネットワークの帯域幅を調整するシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般的には、ネットワーク帯域幅を調整するシステムおよび方法に関する。より具体的には、さまざまな実施形態が、トラフィックレポーティングおよび帯域幅予約メカニズムに関する。
【背景技術】
【０００２】
インターネットからケーブルＴＶ、ＩＰＴＶ、モバイルブロードバンド、ワイファイ、ホームネットワークまでのコミュニケーションネットワークの継続的な普及およびこれらのネットワークを経由して送信されるデータについてのますます高まる要求に伴い、ネットワークの混雑に関する懸念が、より浸透してきている。より大きなパイプを提供できるのと同じくらい速く、ユーザ要求は、それらを満たすと脅かしている。インターネットを支えるこれらのような、データ配信のための最善努力メカニズムは、しばしば、不十分であると分かります。
【０００３】
ネットワーク混雑の懸念は、ケーブルＴＶネットワークのアウトオブバンド（ＯＯＢ）チャネルのような狭くて非対称なチャネルにとって、特に重大であろう。ＯＯＢケーブルチャネルは、エンハンスドＴＶバイナリーインターチェンジフォーマット（ＥＢＩＦ）またはオープンケーブルアプリケーションプラットフォーム（ＯＣＡＰ）規格を使用して実施されるもののようなエンハンスドＴＶアプリケーションにより発生されるトラフィックを処理するための要求によって、ますます緊張している。そのようなトラフィックは、ボーティングとポーリングデータ、ユーザ嗜好と統計トラフィック、Ｔコマース情報およびその他のデータを含むだろう。ＤＳＬとモバイルブロードバンドのネットワーク上でのデータ配信は、同様な課題に直面している。
【０００４】
帯域制限されたネットワーク上でのネットワーク混雑の課題に対処するために、さまざまなアプローチが記述されてきた。いくつかのアプローチは次のものを含む。すなわち、

（「テールドロップ」または活発な待ち行列管理を経由した）パケットドロッピング、ＴＣＰ混雑回避、明示的混雑通知、ウィンドウシェーピング、トラフィックシェーピング（すなわち、パケット遅延）、サービス（ＱｏＳ）スキームの品質、および、関連した帯域幅予約技術。

【０００５】

必要とされるものは、データを送信および受信するために待機しながら、長いメッセージ、過剰なハンドシェイクまたはその他のＱｏＳタイプのオーバヘッドを要求しないで、ノードがプロセシングし続けるのを許容する方法にて混雑を減らすために、ネットワーク帯域幅を調整するシステムおよび方法である。

【発明の開示】

【０００６】

さまざまな実施形態において、トラフィックレポーティングおよび帯域幅予約メカニズムを利用して、ネットワークトラフィックをモニタリングし、ネットワーク負荷を予測し、トラフィックをスケジューリングすることを含むネットワーク帯域幅を調整するために、システムおよび方法が提供されるだろう。トラフィックレポーティングは、メッセージを送信および受信する適切な回数を指示するネットワークノードへ制御メッセージをブロードキャストすることを含むだろう。ネットワークノードは、ネットワークでのその使用を積極的に調整するために、トラフィックレポート（例えば、制御メッセージ）を使用するだろう。予約は、同期または非同期方法にて実施されるだろう。前記予約メカニズムは、従来のストリームソケットＡＰＩをエミュレートするだろう。

【０００７】

別の実施形態では、前記トラフィックレポートブロードキャストに加えて、クライアントノードが、メッセージまたはその他のデータを送信または受信する前に、帯域幅予約を行うだろう。

【０００８】

いくつかの実施形態では、システムおよび方法が、デジタルケーブルＴＶアウトオブバンド（ＯＯＢ）ネットワークにおいて、ユーザデータグラムプロトコル（ＵＤＰ）上で帯域幅調整および軽量の送信制御プロトコル（ＴＣＰ）機能性を提供するだろうし、そこでは、エンハンスドテレビジョン（ＥＴＶ）プラットフォームサーバーが、複数のセットトップボックス（ＳＴＢ）が動作するエンハンスドＴＶバイナリーインターチェンジフォーマット（ＥＢＩＦ）ユーザエージェントを有するＵＤＰを使用してインターチェンジされるハイパーテキストトランスファープロトコル（ＨＴＴＰ）ペイロードを経由して、コミュニケーションしている。

【０００９】

他の実施例は、ＴＣＰやその他のプロトコルを利用するネットワーク、インバンドケーブルＴＶ、ＤＳＬおよびセルラー／モバイルネットワークのようなネットワーク、および、その他のアーキテクチャーやコンポーネントに基づいたネットワークを有している。

【００１０】

別の実施形態では、帯域幅調整が、ＥＢＩＦアプリケーションにとって明白であろう方法にて、ＥＢＩＦユーザエージェントにより実施されるだろう。他の実施例では、前記トラフィックスケジュールが、ネットワークアプリケーションにさらされるだろうし、それが、それから、前記開示されたトラフィックスケジューリングメカニズムを使用でき、ユーザ側の動作を適切に調整できる。

【００１１】

別の実施形態では、帯域幅調整サーバーが、適切にマネージされるために、外部イベント（例えば、時刻または時季帯域幅予測、おそらくトラフィックストームを示す緊急ニュース速報）に応答しているだろう。

【００１２】

上述の一般的な記述および以下の詳細な記述の双方が、例示的で説明のみであり、特許請求される本発明を制限するものではないことが理解されるべきである。添付の図面は、

この明細書の一部を構成し、本発明のいくつかの実施例を例示しており、詳細な記述とともに、本発明の原理を説明する役割を果たしている。

【図面の簡単な説明】

【００１３】

本発明は、添付の図面とともに、以下の詳細な記述を読むことによって、もっと完全に理解することができ、添付図面では、同様な参照表示が同様な要素を指定するために使用されており、また、次のようになっている。すなわち、

【００１４】

【図１】は、ケーブルシステムのアウトオブバンドネットワークにおける帯域幅調整を提供するシステムの一実施例を描写している例示的なアーキテクチャー図である。

【００１５】

【図２】は、本発明の実施形態によるトラフィックマネージメントサーバーの一実施例を描写している例示的なアーキテクチャー図である。

【００１６】

【図３】は、本発明の実施形態によるトラフィックマネージメントクライアントの一実施例のプログラマー図を描写している例示的なアーキテクチャー図である。

【００１７】

【図４（ａ）】は、従来技術のＨＴＴＰのＴＣＰ／ストリームソケット実装を描写しているフローチャートである。

【００１８】

【図４（ｂ）】は、本発明の一実施形態によるＨＴＴＰの同期（ブロッキング）仮想ソケット実装を描写しているフローチャートである。

【００１９】

【図４（ｃ）】は、本発明の一実施形態によるＨＴＴＰの非同期（非ブロッキング）仮想ソケット実装を描写しているフローチャートである。

【００２０】

【図５（ａ）】は、例示的な情報パケットのグラフィック表示である。

【００２１】

【図５（ｂ）】は、例示的なデータパケットのグラフィック表示である。

【００２２】

【図５（ｃ）】は、例示的なＡＣＫパケットのグラフィック表示である。

【００２３】

【図５（ｄ）】は、例示的なプローブパケットのグラフィック表示である。

【００２４】

【図６（ａ）】は、仮想ソケット構造のグラフィック表示である。

【００２５】

【図６（ｂ）】は、仮想ソケットテーブル構造のグラフィック表示である。

【００２６】

【図６（ｃ）】は、予約待ち行列構造のグラフィック表示である。

【００２７】

【図７（ａ）】は、帯域幅調整無しのネットワーク負荷のシミュレーションを描写しているグラフである。

【００２８】

【図７（ｂ）】は、帯域幅調整有りのネットワーク負荷のシミュレーションを描写しているグラフである。

【００２９】

【図７（ｃ）】は、帯域幅調整有りと無しのシミュレーションされたネットワーク負荷間の相違を描写しているグラフである。

【発明を実施するための形態】

【００３０】

図１に例示されているように、本開示は、トラフィックマネージメント（ＴＭ）システム１００および方法のコンテキストにおいて、トラフィックスケジューリングを使用するネットワーク帯域幅調整に関する。より具体的には、一実施形態が、デジタルケーブルＴＶアウトオブバンド（ＯＯＢ）ネットワークのコンテキストにおいて開示され、そこでは、エンハンスドテレビジョン（ＥＴＶ）プラットフォームサーバー２００が、エンハンスドＴＶバイナリーインターチェンジフォーマット（ＥＢＩＦ）ユーザエージェント４１０を起動する複数のセットトップボックス（ＳＴＢ）４００を有するユーザデータグラムプロトコル（ＵＤＰ）を使用して互いにやりとりされるハイパーテキストトランスファープロトコル（ＨＴＴＰ）ペイロードを経由して、コミュニケーションする。帯域幅調整を提供するのに加えて、開示されたＴＭシステム１００および方法は、保証された配信、パケットシーケンシング、フロー制御およびデータ圧縮をサポートするＵＤＰを使用する軽量の送信制御プロトコル（ＴＣＰ）機能性を提供するだろう。

【００３１】

ここにて記述されたシステムおよび方法は、また、ＴＣＰおよび他のプロトコルを利用するトラフィックとともに使用のためや、インバンドケーブルＴＶ、ＤＳＬおよびセルラー／モバイルネットワークのようなその他のネットワークのためやその他のアーキテクチャーおよびコンポーネントに基づいたネットワークのために適切であろう。

【００３２】

本実施形態では、ＴＭシステム１００および方法は、デジタルケーブルＯＯＢネットワークの以下の特性を考慮に入れている。すなわち、

【００３３】

非対称性：ダウンストリームネットワーク帯域幅（ヘッドエンド→ＳＴＢ）がアップストリームより広い。

【００３４】

低帯域幅：いずれの方向での帯域幅がＭｂｐｓではなくＫｂｐｓで測定される。

【００３５】

長い待ち時間：ハイブリッドファイバー同軸（ＨＦＣ）ネットワーク上のトンネリングインターネットプロトコル（ＩＰ）が遅延を発生する。

【００３６】

ＴＣＰよりも優先されるＵＤＰ：帯域幅とタイムを節約し、全てのケーブルＯＯＢネットワークがＴＣＰを提供しているのではない。

【００３７】

ＨＴＴＰが優位に立つ：ほとんどのトラフィックはＨＴＴＰリクエスト／応答上でモデル化されるが、例外があるだろう。

【００３８】

リクエスト／応答２分法：ほとんどのトラフィックが一方向または他方向に移動するデータからなり、リクエスト方向がはるかに少ないデータを運ぶ傾向にある。

【００３９】

脆弱性：アップストリーム混雑がネットワーク上のＯＯＢネットワークおよびＳＴＢをクラッシュさせるだろう。

【００４０】

混合された前景／背景：いくつかのリクエストがタイムリーな応答を必要とする一方で、その他のリクエストはそれを必要としない。

仮定および定義

【００４１】

現在の実施形態の記述を容易にするために、基本的なネットワークについて、以下の仮定がなされるだろう。すなわち、

【００４２】

ケーブルＯＯＢネットワークは、ＩＰｖ６ではなく、ＩＰｖ４を排他的に使用する。

【００４３】

ＩＰパケットヘッダーは、２０バイトを専有する。
【００４４】
ＵＤＰパケットヘッダーは、８バイトを専有する。
【００４５】
メディアアクセス制御（ＭＡＣ）レイヤーセルは、少なくとも３４バイト（サイズは変わります）のペイロードを有する。
【００４６】
ＩＰおよびＵＤＰヘッダーの２８バイトと共に一つのＭＡＣセルに収まり得る最大ＵＤＰメッセージサイズは、３４－２８＝６バイトである。これが、ＴＭパケットヘッダーのサイズを駆動する。
【００４７】
ハイブリッドファイバー同軸ネットワークノード（ＨＦＣノード、すなわち、デモジュレータ）当たりの合計アップストリーム帯域幅が、５００～２０００ＳＴＢで共有されて、２５６Ｋｂｐｓである。
【００４８】
スロットされたアロハネットワークが、衝突により、３６．８％の理論的な最大効率を有する。
【００４９】
モジュレータ当たりの合計のダウンストリーム帯域幅が、３２，０００ＳＴＢまで共有されて、２Ｍｂｐｓである。
【００５０】
現在の実施例の性能は、実装に使用されるであろう以下の数値定数に基づいているだろう。代表値は、実験と以下の配備環境の特性に基づいて変化する可能性が指摘されている。すなわち、
【００５１】
ＭＴＵ：ＴＭアップストリームトラフィックのための最大送信ユニット（すなわち、ＵＤＰパケットサイズ）は、２４４バイトまたは１９５２ビットだろう。これが、ＩＰおよびＵＤＰヘッダーの２８バイトを含む６個のＭＡＣセル（２７２バイト）内に、一つのＭＴＵが正確に収まることを可能としている。全アップストリームメッセージの９９％が、１ＭＴＵに収まるべきである。ダウンストリームは、ＭＴＵがより大きいだろうし、最大９８７バイトまである。
【００５２】
ＳＬＯＴ：２５６Ｋｂｐｓ、または、０．００８秒（８ミリ秒）で、一つのＭＴＵを送信するために要求されるタイム。
【００５３】
ＳＬＯＴＳ＿ＰＥＲ＿ＳＥＣＯＮＤ：一定の１２５（すなわち、１０００ｍｓ／ｓｅｃ／８ｍｓ／ＳＬＯＴ）。
【００５４】
ＳｌｏｔＩＤ：００：００：００で０１／０１／２０００のときから、一つのＳＬＯＴ（すなわち、８ミリ秒時間スライス）をユニークに識別する６バイト（またはより大きい）署名の無い整数。つまり、０のＳｌｏｔＩＤは、００：００：００．０００で０１／０１／２０００にスタートして、００：００：００．００８で０１／０１／２０００に終わる、８ミリ秒時間スライスを指し、ＳｌｏｔＩＤ２は、００：００：００．００８で０１／０１／２０００に始まって、００：００：００．０１６で０１／０１／２０００に終わる等の次回の時間スライスを指す。例えば、２０１０年５月８日３：２１：０４ＡＭの日付時刻のための３４番目のＳＬＯＴのＳｌｏｔＩＤは、４０８２５５０８０３４である。この日付の１００年記念秒を表す最初のＳＬＯＴのＳｌｏｔＩＤは、２４２００７９０８０００である。
【００５５】
ＶＰＯＲＴ：全ＴＭトラフィック用に使用される実際のポート数（１９６２）。この数

は、ＢＩＡＰ社に登録されている。
【００５６】
ＶＳＥＮＤ＿ＷＡＳＴＥ：データ送信関数（例えば、ｖｓｅｎｄ（））がブロックし続けるべきであるパケット間のミリ秒の最大数。最初は、１００ミリ秒に設定されている。
【００５７】
ＶＳＥＮＤ＿ＴＲＩＥＳ：ｖｓｅｎｄ（）が戻る前に試みるべきであるパケット再送信の最大数。最初は、５に設定される。
【００５８】
ＶＲＥＣＶ＿ＷＡＳＴＥ：データ受信関数（例えば、ＶＲＥＣＶ（））がデータパケットを待ってブロックし続けるべきであるミリ秒の最大数。最初は、１００ミリ秒に設定される。
トラフィックマネージメントアーキテクチャー
【００５９】
図１は、ケーブルシステムのＯＯＢネットワークにおける帯域幅調整を提供するシステム１００の一実施例を描写している例示的なアーキテクチャー図である。描写されたＴＭクライアント４２０およびＴＭサーバー２１０コンポーネントは、内蔵された帯域幅調整を有するＵＤＰ上で、ＴＣＰのような機能性を提供するだろう。
【００６０】
本実施例は、木と枝のトポロジーにて複数のハブ（図示せず）およびＨＦＣノード３００を介して接続された顧客構内に設置された複数のＳＴＢ４００とコミュニケーションするケーブルオペレータヘッドエンドに設置された一つまたはそれ以上のＥＴＶプラットフォームサーバー（ＥＰＳ）２００から成るだろう。ＥＰＳ２００は、ＨＴＴＰ／ＴＣＰを経由してアパッチＨＴＴＰサーバー２２０へＨＴＴＰペイロードからなる着信ＴＭ／ＵＤＰトラフィックをリレーし、それから、中間ＨＦＣノード３００とハブを介してＭ／ＵＤＰ経由でＨＴＴＰ／ＴＣＰ応答を逆に送信元のＳＴＢ４００へリレーする、高性能ＵＤＰプロキシサーバーであろうＴＭサーバー２１０を具備するだろう。
【００６１】
ＳＴＢ（すなわち、クライアントノード）４００は、ＳＴＢ４００上で作動しているＥＢＩＦアプリケーション（図示せず）のデータネットワーキングニーズをサポートするためにＴＭクライアント４２０を利用するＥＢＩＦユーザエージェント４１０を具備するだろう。ＴＭクライアントコンポーネント４２０は、単一の実データグラムソケット／ポートのみを使用しながら複数の同時「仮想」ストリームソケットおよび接続を提供する、ＴＣＰネットワーキング用の標準ストリームソケットアプリケーションプログラミングインターフェース（ＡＰＩ）を模倣する、静的ライブラリであろう。
【００６２】
示されたＴＭ／ＵＤＰプロトコルは、メッセージ配信を保証するために、簡単なパケットフラグメンテーション、シーケンシングおよびアクノリッジメントスキームを使用するだろう。帯域幅調整は、ネットワークの負荷を予測するためにアップストリームトラフィックをモニターするであろうＴＭサーバー２１０によって制御され、ＴＭクライアント４２０に、アップストリームメッセージをどのようにスケジュールするかを（例えば、帯域幅予約を経由して）告げるＳＴＢ４００へ制御メッセージをブロードキャストするだろう。ダウンストリーム帯域幅調整は、ＴＭサーバー２１０内にて、完全に達成されるだろう。
【００６３】
図１に描写されたアーキテクチャーを考慮すると、支援される３つの運用モードがあるだろうし、その一つのみが、任意の特別ケーブルオペレータ用に使用されるだろう。どのモードが支援されるかは、（ａ）負荷情報のＳＴＢ４００へのＵＤＰブロードキャスト用のダウンストリーム帯域幅の連続的なトリクルの利用可能性、（ｂ）クライアントシステム上での一つのソケット／ポート常駐割り当ての（ａ）および（ｂ）に依存するだろう。これらのモードは以下であろう。すなわち、

【００６４】
モード１：（ａ）と（ｂ）双方が利用可能な時は、ＴＭサーバー２１０は、情報パケット（以下を参照のこと）を、全ＳＴＢ４００に周期的に、例えば、重い負荷状態の間に１０秒毎かもっと頻繁にブロードキャストするだろう。これは、約３０ｂｐｓ（０．０３Ｋｂｐｓ）を必要とすべきである。
【００６５】
モード２：（ａ）は利用可能だが（ｂ）が利用可能でなく、ＴＭクライアント４２０が任意のメッセージを送信する前に情報パケットを待たなければならない時は、これらのパケットは、もっと頻繁に送信されなければならない。これは、約２７０ｂｐｓ（０．２７Ｋｂｐｓ）を必要とすべきである。
【００６６】
モード３：（ａ）が利用可能でない時は、すべてのメッセージを送信する前に、プローブパケット（以下に記述される）は、「なりゆきを見る」ために使用されなければならない。これは、構成された制限を越えて、簡潔な帯域幅スパイクを引き起こし、従って、その他のモードよりも好ましくない。
トラフィックマネージメントサーバー
【００６７】
図２は、トラフィックマネージメント（ＴＭ）サーバー２１０の一実施形態を描写している例示的なアーキテクチャー図である。
【００６８】
ＴＭサーバー２１０は、配信プロトコルおよびＯＯＢ帯域幅調整を保証したＴＭを実装する高性能のＵＤＰ→ＴＣＰリレーサーバーだろう。着信接続およびデータは、ＵＤＰリッスナープロセス２１４を経由して受信されるだろうし、接続／パケットは、アジェンダ２１８上にタスクとして配置されるだろう。図２に破線で描写されているように、アジェンダ２１８は、いくつかのプロセスによって読み取られるだろう。
【００６９】
データまたはプローブパケットが、ＩＰアドレス、ポートおよびＣｏｎｎＩＤ（以下のパケットフォーマットセクションを参照のこと）ヘッダーフィールドの新しい組み合わせで到達するたびに、接続が、暗黙のうちに「オープンされる」だろう。それぞれが複数のワーカースレッドを実施する複数のチャイルドプロセス２１６が、これらのパケットを、それらの到達につれて、プロセスするだろうし、それらをオリジナルメッセージへと適切な順番で再構成し、必要に応じてクライアントへＡＣＫパケットを返す。完成したメッセージは、それから、標準ＨＴＴＰ／ＴＣＰ／ＩＰ接続を経由して、アパッチＨＴＴＰサーバー２２０へ送信されるだろう。アパッチからのＨＴＴＰ応答は、それから、「オープン」仮想接続を経由して（すなわち、ダウンストリームデータパケットを経由して）、対応するクライアントへ戻されるだろう。
【００７０】
帯域幅調整プロセス２１２は、着信パケッリトを、リアルタイムにて、モニターするだろう。この情報とその変化率が、負荷傾向を推定するのに使用されるだろう。この推定から、対応する送信確率が演算されるだろう。この送信確率およびクロック同期情報が、ＵＤＰを経由して、全ＴＭクライアント４２０へ周期的にブロードキャストされるだろう。
トラフィックレポーティング
【００７１】
このセクションは、負荷情報をＴＭクライアント４２０へコミュニケーションするための例示的な方法を記述している。
【００７２】
ＴＭサーバー２１０およびＴＭクライアント４２０は、メッセージを送信するのに使用されるタイムを広げることにより、アップストリームＯＯＢ帯域幅をマネージするだろう。すべてのメッセージは、複数の固定サイズのデータパケットに分割されるだろうし、各パケットは、ランダム遅延により、バインドされるだろう。例えば、もし、現在の広域送

信確率（すなわち、可変ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ）が３２７６７ならば、（ＳＬＯＴでの）遅延は、次のように計算されるだろう。すなわち、

【００７３】

ｔｍ＿ｒａｎｄｏｍ（）％（１３１０７２．０／３２７６８＋０．５）
＝ｔｍ＿ｒａｎｄｏｍ（）％４

【００７４】

これは、０と３の間の値という結果になる。例えば、もし、ｔｍ＿ｒａｎｄｏｍ（）が４４６を返すならば、その時は、４４６％４＝２である。ＴＭクライアント４２０上のチャネル／スロット予約関数への呼出が、呼出元へ、１６ミリ秒の予約遅延（すなわち、８＊２で、そこでは、各々ＳＬＯＴが８ミリ秒である）を返すであろう。この情報は、また、現在のＳｌｏｔＩＤ値に２だけ遅延して可変であるｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔを更新するのに使用されるだろう。これは、ＴＭクライアント４２０により維持された予約待ち行列内の最初の仮想接続が、今から１６ミリ秒（すなわち、２ＳＬＯＴ）後に送信するためのパケットを有するということを意味すると解釈される。その時、予約関数への後続の呼出は、予約されたタイムが、到来したことを理解して、その後にメッセージを送信する呼出元に、０を返すでしょう。ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ値は、次の如く設定されるだろう。すなわち、

【００７５】

（１）それは、１６，３８７のデフォルト値に初期化される。

【００７６】

（２）モード１および２では、その値は、ＴＭサーバー２１０によりブロードキャストされた情報パケットから直接受信される。この値は、任意のより古いｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ値に取って代わる。モード３では、プローブパケットが、対応するＡＣＫパケットを発生するために、可能な限り早く送信される。

【００７７】

（３）すべてのＡＣＫパケットは、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ値の更新を含有する。この値は、任意のより古いｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ値に取って代わる。

【００７８】

（４）送信されたパケットアクノリッジメントがタイムアウトする（すなわち、パケットがアクノリッジされない）度に、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙは半分にされる（最小１）。ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙがこのようにして半分にされることなしで経過する秒毎に、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙは、最大値１６，３８７まで、100増やされるようにされる。

【００７９】

上記のように、モード１は、ダウンストリーム帯域幅約０．０３Ｋｂｐｓを必要とする情報パケットのブロードキャストストリームを発生する。モード２は、０．２７Ｋｂｐｓを必要とする。モードに依存して、クライアントは、真のソケットが利用可能の後に、できる限り早く、情報パケットのリッスニングをスタートすべきである。

ランダム数発生

【００８０】

このセクションは、本発明のさまざまな実施形態のメッセージ遅延とその他のニーズを計算するために適切なランダム数発生器（すなわち、ｔｍ＿ｒａｎｄｏｍ（）とｔｍ＿ｒａｎｄｏｍｉｚｅ（））の例示的な実施を提示する。すなわち、

```
ｓｔａｔｉｃユニット３２　ｔｍ＿ｓｅｅｄ＝１９６１０５０８；

ユニット３２　ｔｍ＿ｒａｎｄｏｍ（）｛
　　ｃｏｎｓｔユニット３２　ａ＝１６８０７；
　　ｃｏｎｓｔユニット３２　ｍ＝２１４７４８３６４７；
　　ｃｏｎｓｔユニット３２　ｑ＝１２７７７３、／＊ｍ／ａ＊／
　　ｃｏｎｓｔユニット３２　ｒ＝２８３６、／＊ｍ％ａ＊／
```

```
      ｉｎｔ３２　ｌｏ、ｈｉ、テスト；

      ｈｉ＝ｔｍ＿ｓｅｅｄ／ｑ；
      ｌｏ＝ｔｍ＿ｓｅｅｄ％ｑ；
      テスト＝ａ＊ｌｏ－ｒ＊ｈｉ；
      もし（ｉｆ）、（テスト＞０）であれば
       ｔｍ＿ｓｅｅｄ＝テスト；
      それ以外であれば（ｅｌｓｅ）、
       ｔｍ＿ｓｅｅｄ＝テスト＋ｍ；
    リターン　ｔｍ＿ｓｅｅｄ；
}

ｖｏｉｄ　ｔｍ＿ｒａｎｄｏｍｉｚｅ（ユニット３２　ｓ）｛
     ｔｍ＿ｓｅｅｄ＝ｓ、
}
```

【００８１】

ｔｍ＿ｒａｎｄｏｍｉｚｅ（＜ｉｐ＿ａｄｄｒｅｓｓ＞）は、起動時に正確に一度呼出されるべきであり、そこでは、＜ｉｐ＿ａｄｄｒｅｓｓ＞が、ＳＴＢのＩＰｖ４アドレスのユニット３２（４バイト）値である。

クライアントクロック同期

【００８２】

このセクションは、本発明のさまざまな実施形態に好適なクライアントクロック同期の例示的な方法を記述する。

【００８３】

ＴＭサーバー２１０からの情報パケットブロードキャストは、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ更新と共に、タイミング同期情報を含有するだろう。タイミング情報は、情報パケットの２つのフィールドに入るだろう。すなわち、

【００８４】

ＳｙｎｃｈＳｅｃｏｎｄ（１バイト）：０．．１１９の範囲の値であって、パケットが、どの秒に、サーバークロックにより送信されたかを識別する。この値は、最新の偶数分のスタートからのオフセットである。

【００８５】

ＳｙｎｃｈＰｈａｓｅ（１バイト）：このパケットを送信する前の同期秒内に経過したＳＬＯＴ（０．．１２４）の数。

【００８６】

例えば、もし、情報パケットがＴＭサーバー２１０上で構築されるタイムが、1秒の何分の１を含んで、１３：２３：４３．７２８ＧＭＴならば、ＳｙｎｃｈＳｅｃｏｎｄフィールドに割り当てられる値は１０３であろう。分の値２３が奇数なので、直近の偶数分は２２である。１３：２２：００から１３：２３：４３までのオフセットは、０１：４３または１０３秒である。

【００８７】

一旦この秒が識別されると、残りの０．７２８秒（７２８ミリ秒）が考慮されなければならない。一つのＳＬＯＴが０．００８秒（８ミリ秒）を専有すると仮定すると、スロットの数は７２８／８＝９１として計算される。この値（例えば、９１）は、情報パケット内のＳｙｎｃｈＰｈａｓｅフィールドに割り当てられる。

【００８８】

ＴＭクライアント４２０が情報パケットを受信するたびに、それは、ｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃ変数を更新するために、以下のステップを実行するだろう。すなわち、

【００８９】

（１）浮動小数点変数ｃｕｒｒｅｎｔ＿ｔｉｍｅ内のいくつかのエポック以来の秒として

現在の日付／時刻（ＧＭＴ）を計算する。
【００９０】
（２）浮動小数点変数ｃｕｒｒｅｎｔ＿ｔｉｍｅ内で同じエポック以来の秒として情報パケット（ＧＭＴ）で見出された２つのフィールドにより定義されたタイムを計算する。
【００９１】
（３）浮動小数点変数ｓｙｎｃｈ＿ｏｆｆｓｅｔ＿ｍｓｅｃ＝（ｓｙｎｃｈ＿ｔｉｍｅ–ｃｕｒｒｅｎｔ＿ｔｉｍｅ）＊１０００、ミリ秒での最新のオフセット、を計算する。
【００９２】
（４）ｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃ＝ｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃ＊０．９＋ｓｙｎｃｈ＿ｏｆｆｓｅｔ＿ｍｓｅｃ＊０．１を設定する。これは、条件がタイムとともに変化してタイムオフセットの低速移動平均を発生する。任意の情報パケットの受信に先立って、ｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃをゼロ（０）に初期化する。
【００９３】
モード１では、ｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃを情報パケット毎に更新することは必要ではないだろう。１分毎の一つの更新で十分であろう。モード２では、このタイムオフセットは、望ましくは、情報パケット毎に計算されるだろう。
【００９４】
「現在タイム」が参照されるたびに、それが、ＳＴＢ（ＧＭＴ）の現在タイム、プラス、適切なタイムユニット（すなわち、もし「現在タイム」が秒にてレポートされるならば、ｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃ＊１０００）にスケーリングされたｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃ値、であると仮定されるだろう。
トラフィックマネージメントクライアント
【００９５】
図３は、本発明の実施形態によるトラフィックマネージメントクライアント４２０の一実施例のプログラマー図を描写している例示的なアーキテクチャー図である。それは、クライアントデバイス（例えば、ＳＴＢ）４００により提供されるハードウェアおよびオペレーテイングシステム（Ｏ／Ｓ）４５０のその他の能力と共にＳＴＢミドルウェア４４０のソケット媒介能力を利用することで複数のＥＢＩＦアプリ４３０用のトラフィックマネージメントサービスを提供するために、ＴＭクライアント４２０がＥＢＩＦユーザエージェント４１０によりどのように利用されるだろうかを示している。さらに記述されるように、ＴＭクライアント４２０は、標準ストリームソケットＡＰＩを利用し、トラフィックスケジューリングを使用する帯域幅調整のために同期および非同期両モードを提供し、クライアントおよびサーバー両接続を許容し、複数の「仮想」ソケット／ポートをサポートし、保証された配信を提供し、データ圧縮を提供し、そして、先験的な帯域幅制限はなにも課さないだろう。
ネットワークＡＰＩ
【００９６】
図４（ａ）は、ネットワーキングクライアントにより使用されるだろうＨＴＴＰの従来技術のＴＣＰ／ストリームソケット実装を描写している例示的なフローチャートである。スタート（ステップ５００）すると、ｓｏｃｋｅｔ（）は、新しいソケットｉｄを得るために、呼出されるだろう（ステップ５０２）。次に、ＩＰアドレスおよびポート数により定義されたサーバーとの接続が、ｃｏｎｎｅｃｔ（）を呼出すことにより得られるだろう（ステップ５０４）。リクエストストリングが、それから、ｓｅｎｄ（）関数によって、サーバーへ送信されるだろう（ステップ５０６）し、全リクエストをプロセスするために、おそらく、ｓｅｎｄ（）に複数の呼出を要求する（ステップ５０８）。全てのデータが送信された（すなわち、全て送信した？＝Ｙｅｓ（ステップ５０８））後に、応答を待つために、ｒｅｃｖ（）へ呼出がなされるだろう（ステップ５１０）。ｓｅｎｄ（）（ステップ５０６、５０８）と同様に、ｒｅｃｖ（）への複数の呼出が、全ＨＴＴＰ応答を検索するために要求される（すなわち、全て受信した？＝Ｙｅｓとなるまで繰り返し適用され

る）だろう（ステップ５１２）。最後に、ｃｌｏｓｅ（）が、接続をクローズするために呼出されるだろうし（ステップ５１６）、その後に、プロセスがストップする（ステップ５１８）。また、全部のＨＴＴＰ応答は受信されないことが生じるだろうが、接続はクローズされる（ステップ５１４）。この例では、プロセスをストップさせる（ステップ５１８）のためにｃｌｏｓｅ（）が呼出されるだろう（ステップ５１６）。このタイプの同期ＨＴＴＰクライアントを実装することは、当業者により知られているであろう。

【００９７】

図４（ｂ）および図４（ｃ）は、本発明の一実施形態によるＨＴＴＰの仮想ソケット実装の実施形態を描写している。図４（ｂ）におけるフローチャートは、同期（ブロッキング）バージョンを描写している。図４（ｃ）におけるフローチャートは、非同期（非ブロッキング）バージョンを描写している。

【００９８】

これらの図に示されているように、標準ストリームソケットＡＰＩ呼出は、仮想等価（例えば、ｓｅｎｄ（）５０６がｖｓｅｎｄ（）６１２になる）であろうし、よく知られている方法でプログラマーにとってストリームソケットが使用できるようにする。影付きブロックで示されているように、少数の追加関数が存在するだろう。本発明のさまざまな実施形態によれば、全アップストリームトラフィックが、ネットワークの現在負荷状況に基づいて、望ましくはスケジュールされるだろう。ネットワーク負荷が低い時、データは、直ちにではなくとも、迅速に送信されるだろう。ネットワーク負荷がより高い時、遅延があるだろう。ＴＭクライアント４２０は、ｖｒｅｓｅｒｖｅ（）６０６のリターン値から、予期された遅延を推定でき、どのように進むか進まないかを決定でき、そして、他のタスクのために介在するサイクルを使用することができる。

【００９９】

示されたＡＰＩ呼出は、静的ライブラリを経由して提供され、Ｃヘッダーファイル（.ｈ）にプロトタイプされるだろう。そのようなヘッダーファイルは、従来のストリームソケットＡＰＩをそれらのトラフィックマネージメントアナログへとマッピングするマクロを＃ｄｅｆｉｎｅするだろう。そのようなマクロは、現存するソケットコードがトラフィックマネージメントシステムおよび方法と共に働くことを許容するであろうし、送信呼出および非同期プロセシングのスケジューリングのために些細な変更のみを要求する。

【０１００】

以下の段落では、開示されたステップのコンテキスト中で、図４（ｂ）および図４（ｃ）に描写された仮想呼出の詳細が提供される。記述されたすべてのＡＰＩ呼出について、パラメータとリターンのタイプは、従来のソケットＡＰＩにて使用されるのと同じタイプを一般的に参照するであろうし、従って、ここではそれらを記述していない。文字「ｖ」で始まる名称は、ここにて記述されたシステムおよび方法に固有である。標準（すなわち、非ＴＭ）ソケットＡＰＩは、ｈｔｏｎｌ（）、ｎｔｏｈｌ（）、ｇｅｔａｄｄｒｉｎｆｏ（）等のように同様に利用可能であると仮定されている。その他の従来のソケットＡＰＩは、ｐｏｌｌ（）、ｓｅｌｅｃｔ（）およびｓｈｕｔｄｏｗｎ（）のように、トラフィックマネージメントを使用する時に、アナログを持っていないであろう。

【０１０１】

図４（ｂ）は、本発明の一実施形態によるＨＴＴＰの同期（ブロッキング）仮想ソケット実装を描写している例示的なフローチャートである。

【０１０２】

プロセスがスタート（ステップ６００）の後、最初の呼出はｖｓｏｃｋｅｔ（）であり、仮想ソケット記述子を割り当てる（ステップ６０２）だろう。この関数により返された記述子は、ｖｌｉｓｔｅｎ（）、ｖｂｉｎｄ（）、ｖｓｅｎｄ（）およびその他の関数への後続の呼出用に使用されるだろう。ＴＭクライアント４２０での実装は、仮想ソケットテーブルにおける新しい空の仮想ソケット構造を割り当てることと、そのエントリーのテーブルインデックスを仮想ソケット記述子として返すことを含むだろう。モード２では、この呼出が、また、ｔｍ＿ｓｏｃｋｅｔのオープニングおよび初期化を引き起こすだろうし

、従って、それは情報パケットをリッスンし始めることができる。

【０１０３】

次のステップにおいて、リクエストノードが、ｖｃｏｎｎｅｃｔ（）（ステップ６０４）を呼出して、仮想ソケットをＴＭサーバー２１０に接続するだろう。ｖｃｏｎｎｅｃｔ（）６０４に渡す関連するパラメータは、仮想ソケット記述子と、ＩＰアドレスと、接続先サーバのポート番号を含むだろう。もし、呼出プログラムが、どんなローカルポートが接続に使用されるかを特定することを望むならば、ｖｂｉｎｄ（）への呼出がなされるだろう（図４（ｂ）には描写されていない）。ｖｂｉｎｄ（）は、特定された仮想ソケット記述子をローカルＩＰアドレスおよび明示的なローカルポート数と関連付けるだろう。仮想ソケットのポート数は、仮想ソケット記述子によりインデックスを付された仮想ソケット構造テーブルエントリーに保存されるだろう。もし、ｖｂｉｎｄ（）が、ソケット記述子上で、前もって呼出されていなかったならば、示されたソケット記述子は、ローカルノードのＩＰアドレスおよびランダムローカルポートに自動的にバインドされるだろう。ノードは、たぶん、どのローカルポートが使用されるのかを気にしないので、呼出ノードがサーバーでないならば、これは容認されるべきである。一旦仮想ソケットが接続されると、ｖｓｅｎｄ（）およびｖｒｅｃｖ（）は、データを送信および受信するために、必要に応じて、呼出されることができる。サーバーのホストＩＰアドレスとポートは、ｓｏｃｋｆｄによりインデックスを付された仮想ソケット構造用の仮想ソケットテーブルに保存されるだろう。

【０１０４】

次に、ｖｒｅｓｅｒｖｅ（）（ステップ６０６）への呼出が、データアップストリームを送信するための予約を行うためになされるだろう。ｖｒｅｓｅｒｖｅ（）６０６は、仮想ソケット記述子（ｖｓｏｃｋｆｄ）を受け入れ、（ｖｓｅｎｄ（）を使用して）メッセージを送信する以前に待機するために、ミリ秒の数を返すだろう。ｖｒｅｓｅｒｖｅ（）６０６がゼロを返す（すなわち、送信準備できている？＝Ｙｅｓ（ステップ６０８））際に、メッセージは、ブロックされる最小の確率にて送信できる。予約されたスロットを待ちながら、呼出プログラムは、その他のタスクを実行するだろう。ｖｒｅｓｅｒｖｅ（）６０６は、予約ステータスをチェックするために、周期的に呼出されるだろう。

【０１０５】

次のステップにおいて、ｖｓｅｎｄ（）が、ｖｓｏｃｋｅｔ（）呼出によって返された仮想ソケットを経由して、前もって接続されたサーバーへメッセージを送信するために、使用されるだろう（ステップ６１２）。ｖｓｅｎｄ（）６１２は、接続された仮想ソケット記述子（ｖｓｏｃｋ）、送信されるデータを含有するバッファー（ｂｕｆ）へのポインターおよび送信するバイトの数（ｌｅｎ）を示す入力として、パラメータを受け入れるだろう。それは、実際に送信されたバイトの数を返すだろう。図４（ａ）（ステップ５０６）に描写された従来のｓｅｎｄ（）呼出と同様に、ｖｓｅｎｄ（）は、同期（すなわち、ブロッキング）呼出であり、全てのデータが送信される以前（すなわち、全て送信した？＝Ｎｏ（ステップ６１４））に、返すだろう。これにより、ｖｓｅｎｄ（）は、ｖｒｅｓｅｒｖｅ（）への追加の呼出と連動して繰り返し呼出されるだろう（ステップ６１２および６１４）。大きなメッセージ用、または、ネットワークが重く負荷される時に、これが、たぶん、そのケースであろう。

【０１０６】

もし、ｖｓｅｎｄ（）６１２が呼出されて、特定された仮想ソケット用になんの予約も見出されないならば、ｖｓｅｎｄ（）６１２は、内部的にｖｒｅｓｅｒｖｅ（）６０６を呼出し、予約が到達するまでブロックし（すなわち、スリープし）、それから、送信し始めるだろう。もし、ｖｓｅｎｄ（）６１２が、予約なしで呼出されるならば、または、ｖｒｅｓｅｒｖｅ（）６０６がゼロを返す以前は、ｖｓｅｎｄ（）６０６は、データを送信できるまでブロックするだろう。そのような場合、介在するタイム、たぶん、数秒が、浪費されるであろう。

【０１０７】

アップストリームメッセージスケジューリングは、与えられたメッセージのためのｖｒｅｓｅｒｖｅ（）６０６への最初の呼出で、または、問題の仮想ソケットのためになんの予約もされて来なかった時のｖｓｅｎｄ（）６１２への呼出で、生じるだろう。ｖｓｅｎｄ（）６１２ＡＰＩは、（ａ）その仮想ソケットのための予約テーブル内に予約が存在するまで、および、（ｂ）ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔ内のＳｌｏｔＩＤが現在のＳｌｏｔＩＤとマッチするまでは、仮想ソケットのためのデータを実際に送信することはないだろう。メッセージスケジューリングプロセスは、以下のように、機能するだろう。

【０１０８】

呼出がｖｒｅｓｅｒｖｅ（ｖｓｏｃｋｅｔ＿ｄｅｓｃｒｉｐｔｏｒ）へなされると仮定すると、次のようになる。すなわち、

【０１０９】

（１）現在タイムのＳｌｏｔＩＤを計算し、そこでは、ｃｕｒｒｅｎｔ＿ｓｌｏｔ＿ｉｄ＝＃＿ｏｆ＿ｗｈｏｌｅ＿ｓｅｃｏｎｄｓ＿ｓｉｎｃｅ＿０１／０１／００＿００：００：００＊１２５＋ｆｒａｃｔｉｏｎ＿ｏｆ＿ｃｕｒｒｅｎｔ＿ｓｅｃｏｎｄ／０．００８である。

【０１１０】

（２）もし、ｖｓｏｃｋｅｔ＿ｄｅｓｃｒｉｐｔｏｒが、予約待ち行列内の位置Ｘにて見出されるならば、ｑｕｅｕｅ＿ｐｏｓ＝Ｘに設定して、スキップしてステップ５に進む。そうでなければ、ｑｕｅｕｅ＿ｐｏｓ＝ｓｉｚｅ＿ｏｆ＿ｒｅｓｅｒｖａｔｉｏｎ＿ｑｕｅｕｅ＋１に設定する。

【０１１１】

（３）もし、ｑｕｅｕｅ＿ｐｏｓ＝＝１（すなわち、待ち行列が空）ならば、ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔ＝ｃｕｒｒｅｎｔ＿ｓｌｏｔ＿ｉｄ＋（ｔｍ＿ｒａｎｄｏｍ（）％（１３１０７２．０／ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ＋０．５）に設定する。

【０１１２】

（４）予約待ち行列にｖｓｏｃｋｅｔ＿ｄｅｓｃｒｉｐｔｏｒを加える。

【０１１３】

（５）ｖｒｅｓｅｒｖｅ（）の結果として（８＊（ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔ–ｃｕｒｒｅｎｔ＿ｓｌｏｔ＿ｉｄ）＊ｑｕｅｕｅ＿ｐｏｓ）を返す。

【０１１４】

図４（ｂ）に戻って参照すると、ｖｓｅｎｄ（）を使用して全データが送信された後（すなわち、全て送信した？＝Ｙｅｓ（ステップ６１４））に、前もって接続された仮想ソケットを経由してサーバーからメッセージを受信するために、呼出が、ｖｒｅｃｖ（）になされるだろう（ステップ６１６）。ｖｒｅｃｖ（）呼出（ステップ６１６）は、接続された仮想ソケット記述子（ｖｓｏｃｋ）、受信データブッファー（ｂｕｆ）へのポインター、および、受信データバッファーのバイトでのサイズ（ｌｅｎ）を示す入力パラメータを受け入れるだろう。もし、サーバーへの仮想接続が任意の理由によってクローズするならば、それは、実際に受信されたバイトの数またはゼロ（０）を返すだろう。実際に受信されたバイトの数は、バッファーのサイズよりも小さいであろうから、ｖｒｅｃｖ（）６１６は、全てのメッセージが受信完了（すなわち、全て受信した？＝Ｙｅｓ（ステップ６１８））まで、繰り返して呼出されるだろう。もし、全てのデータが受信完了でなく（すなわち、全て受信した？＝Ｎｏ（ステップ６１８））、サーバーとの接続がクローズされなかった（すなわち、接続がクローズした？＝Ｎｏ（ステップ６２０））ならば、ｖｒｅｃｖ（）は再び呼出されるだろう（ステップ６１６）。

【０１１５】

ｖｒｅｃｖ（）６１６の場合には、ＴＭクライアント４２０が適切なＡＣＫパケットを発生して、それをＴＭサーバー２１０に返送しなければならないことを除いて、ｖｒｅｃｖ（）６１６の実装は、ｖｓｅｎｄ（）６１２と類似しているだろう。また、ｖｒｅｃｖ（）６１６は非同期的に動作できるので、ＴＭサーバー２１０からデータパケットを受信することなくＶＲＥＣＶ＿ＷＡＳＴＥミリ秒が経過した後か、または、提供された受信バッ

ファーが充填完了時に、それは、呼出元に返すべきである。

【０１１６】

図４（ｂ）に戻って、もし、全てのデータが受信完了した（すなわち、全て受信した？＝Ｙｅｓ（ステップ６１８））、または、サーバー接続がクローズされた（すなわち、接続がクローズされた？＝Ｙｅｓ（ステップ６２０））ことが決定されるならば、クライアントノードが仮想ソケットをクローズするためにｖｃｌｏｓｅ（）を呼出すだろうし（ステップ６２２）、プロセスはストップする（ステップ６２４）。小さな数の利用可能な仮想ソケットがあるだろうから、必要とされない時にそれらをクローズすることは、重要である。ｖｃｌｏｓｅ（）６２２の実装は、ｖｓｏｃｋｆｄによりインデックスを付された仮想ソケットテーブル内の仮想ソケット構造を解放することを必要とするだろう。

【０１１７】

図４（ｂ）に描写された同期（ブロッキング）ＨＴＴＰトランザクションは、受信されるべき応答を待ちながら追加のプロセシングを行う必要がない多くのアプリケーションに対しては、十分であろう。しかし、データを待ちながらその他のタスクを実行しなければならないアプリケーションに対しては、非同期（非ブロッキング）コールバックメカニズムが、図４（ｃ）に描写されたように実装されるだろう。この非同期メカニズムは知られているｓｅｌｅｃｔＡＰＩとは異なるだろうが、使用して実装するためにははるかに簡単であろう。

【０１１８】

さて、図４（ｃ）を参照すると、初期のステップは図４（ｂ）のこれらと同一である。プロセスをスタートする時（ステップ６００）、ｖｓｏｃｋｅｔ（）が仮想ソケット記述子を割り当てるために呼出される（ステップ６０２）。そして、ｖｃｏｎｎｅｃｔ（）が仮想ソケットをＴＭサーバー２１０に接続するために呼出される（ステップ６０４）。

【０１１９】

次に、ｖａｓｙｎｃｈ（）が、仮想ソケットを非同期コールバック関数と関連させるために、呼出される（ステップ６０５）。この関数は、ｖｓｏｃｋｅｔ（）を経由して得られた仮想ソケット記述子（ｖｓｏｃｋｆｄ）と、および、ｖｓｏｃｋｆｄ、メッセージコード（ｍｓｇ）と任意のアクティビティ特有の情報（データ）を参照するコールバック関数への参照とを受け入れるだろう。仮想ソケット上の全ての将来アクティビティ（例えば、エラー、予約、受信されたデータなど）は、コールバック関数を起動してアクティビティを識別する適切なメッセージコードを特定するだろう。メッセージコードは以下の如くであろう。すなわち、

【０１２０】

ＶＡＥＲＲＯＲ：実際のエラーコードに対して、ｖｅｒｒｎｏをチェックする。

【０１２１】

ＶＡＲＥＳＯＰＥＮ：ｖｒｅｓｅｒｖｅ（）を経由してオープンされる予約が、ここではオープンである。ｖｓｅｎｄ（）を経由してデータを送信する。

【０１２２】

ＶＡＧＯＴＤＡＴＡ：データがこの仮想ソケットに対して受信された。それを得るために、ｖｒｅｃｖ（）を呼出す。

【０１２３】

ＶＡＧＯＴＣＯＮＮ：新しい接続がこの仮想ソケット上に到達した。それを得るために、ａｃｃｅｐｔ（）を呼出す。

【０１２４】

ｖａｓｙｎｃｈ（）６０５の実装は、仮想ソケット記述子によりインデックスを付された仮想ソケット構造テーブルエントリーに仮想ソケットのコールバック関数を保存することを必要とするだろう。もし、接続が確立完了し、メッセージがアップストリームを送信したならば、パケットヘッダー内のＣｏｎｎＩＤが、受信仮想ソケットを識別するために使用されるだろう。アラートメッセージに対して、ＣｏｎｎＩＤとＳｅｑＮｕｍ用のヘッダースペースがＴＭサーバー２１０により取り込まれ、ポート数を保存するために使用され

るだろう。このポートが、アラートを受信するために、バインドされたソケットを識別するのに使用されるだろう（ＣｏｎｎＩＤ、ＳｅｑＮｕｍおよびパケットヘッダーについての更なる情報に対しては、以下のパケットフォーマットセクションを参照のこと）。

【０１２５】

図４（ｃ）を続けると、ｖａｓｙｎｃｈ（）が呼出された（ステップ６０５）後、ｖｒｅｓｅｒｖｅ（）への呼出が、示されたソケット上でトラフィック予約をなすためになされるだろう（ステップ６０６）し、プロセスがストップする（ステップ６０７）。

【０１２６】

図４（ｃ）に示されているように、送信および受信するプロセスは非同期であり、それぞれのプロセスをスタートするためにコールバック関数（すなわち、ｍｙ＿ｃａｌｌｂａｃｋ（））に依存する。図の点線矢印は、その他のタスクがコールバック起動間で実行されるであろう期間を描写している。具体的には、データを送信するプロセスがｍｙ＿ｃａｌｌｂａｃｋ（）（ステップ６１０）により示されるコールバック関数でスタートし（ステップ６０９）、ｖｓｅｎｄ（）への呼出（ステップ６１２）が続き、それからプロセスがストップする（ステップ６１３）。データを受信するプロセスは、同様である。それは、コールバック関数（すなわち、ｍｙ＿ｃａｌｌｂａｃｋ（）６１０）で、スタート（ステップ６１５）し、図４（ｂ）に記述された同じデータ受信ステップが続く。ｖｒｅｃｖ（）は、全データが受信される（ステップ６１８）か、サーバー接続がクローズされる（ステップ６２０）まで、呼出される（ステップ６１６）だろうし、それから、ｖｃｌｏｓｅ（）への呼出（ステップ６２２）がなされるだろうし、プロセシングがストップする（ステップ６２４）。

【０１２７】

図４（ｂ）または図４（ｃ）に描写されていなくて、有用であろう２個の追加の関数は、ｖｌｉｓｔｅｎ（）とａｃｃｅｐｔ（）を含んでいる。ｖｌｉｓｔｅｎ（）呼出は、示された（すなわち、通過された）バインドされた仮想ソケット（ｖｓｏｃｋｆｄ）上で着信接続をリッスンするためにサーバーとして動作する必要のあるノードによって、使用されるだろう。そのような機能性は、アプリケーションサーバー（例えば、アラートメッセージ用）からの接続を待つノード用に適切であろう。ｖｌｉｓｔｅｎ（）を呼出す前に、ｖｓｏｃｋｅｔ（）とｖｂｉｎｄ（）への呼出が、仮想ソケットを割り当てて、その上で（それぞれ）リッスンするポートを定義するために、なされるべきである。同期リッスンニングが多くのアプリケーション（例えば、ＯＯＢケーブルネットワークでのＳＴＢ）においてなにも意味をなさないだろうから、ａｓｙｎｃｈ（）への呼出が、仮想ソケット非同期をなすために必要とされるだろう。ｖｌｉｓｔｅｎ（）呼出は、直ちに返すでしょうし、非同期コールバック関数が、接続が到達する時に、ＶＧＯＴＣＯＮＮメッセージで起動されるでしょう。ｖａｃｃｅｐｔ（）呼出が定義され、そのような新しい接続を受け入れるために使用されるだろう。ｖｂｉｎｄ（）を経由してポートにバインドされる任意の非同期仮想ソケットは、接続が到達する時に、ＶＧＯＴＣＯＮＮメッセージを受信するでしょう。ｖａｃｃｅｐｔ（）呼出は、接続を受け入れて、データを送信および受信するために使用できる新しい同期仮想ソケット記述子を返すだろう。元のリッスニングソケット記述子（ｖｓｏｃｋｆｄ）は、変化しないで、依然としてリッスンしているだろう。ｖａｃｃｅｐｔ（）によって返された仮想ソケット記述子は、もはや必要ではない時は、ｖｃｌｏｓｅ（）を使用してクローズされるだろう。

【０１２８】

本発明の一実施形態によるＨＴＴＰの仮想ソケット実装の別の実施形態では、ｖｒｅｓｅｒｖｅ（）６０６により提供される予約関数が、ｖｓｅｎｄ（）６１２とｖｒｅｃｖ（）６１６、および、提供されないｖｒｅｓｅｒｖｅ（）６０６呼出に含まれるだろう。この実施例では、（それぞれ）送信または受信されたバイト数を返すｖｓｅｎｄ（）６１２とｖｒｅｃｖ（）６１６に加えて、各呼出は、帯域幅予約を透過的になして、データが直ちに送信または受信され得ない場合に、送信または受信スロットが利用可能になるまでミリ秒数を表す負の数を返すだろう。これは、図４（ｂ）に描写されたｖｒｅｓｅｒｖｅ（

）６０６～ｖｓｅｎｄ（）６１２ループに示されたそれと同じ方法で、データを受信するために待機する間にクライアントノード上で追加のプロセシングが発生するのを許可することにより、ｖｒｅｃｖ（）６１６のブロッキング実装での使用に対して有利であるだろう。それは、また、一つの非標準呼出（ｖｒｅｓｅｒｖｅ（）６０６）を除去することによりＡＰＩを簡略化するだろう。

パケットフォーマット

【０１２９】

単一パイプ上で複数の種類のトラフィックをマネージするために、本発明の一実施形態は、図５（ａ）～図５（ｄ）に関して以下に特定されるように４個の別々のメッセージパケットタイプを定義するだろう。これらは図１に示されたＴＭ／ＵＤＰプロトコルを含むだろう。

情報パケット

【０１３０】

図５（ａ）は、例示的な情報パケット７００のグラフィック描写である。ＴＭサーバー２１０は、システム負荷状況とタイミング同期についての情報をコミュニケーションするために、これらの小さなＵＤＰメッセージを全ＳＴＢ４００に同時にブロードキャストするだろう。情報パケット７００は以下のものを含むだろう。すなわち、

【０１３１】

ＰａｃｋｅｔＴｙｐｅ７０２（１バイト）：有効なＴＭパケットを識別して、それが４つのタイプのいずれであるかを特定するであろう定数。全てのＴＭＰａｃｋｅｔＴｙｐｅ値７０２は、フィルターマスクとして使用できる高位の６ビット０ｘＡ８を含有するだろう。低位の２ビットはパケットタイプによって変化する。情報パケット７００に対して、これらのビットは、情報パケット７００ＰａｃｋｅｔＴｙｐｅ０ＸＡ８＆０ｘ００＝０ＸＡ８用の値をなす０ｘ００である。

【０１３２】

ＳｙｎｃｈＳｅｃｏｎｄ７０４（１バイト）：０．．１１９の範囲の値であって、パケットが、どの秒に、サーバークロックにより送信されたかを識別する。この値は、最新の偶数分のスタートからのオフセットであろう。

【０１３３】

ＳｙｎｃｈＰｈａｓｅ706（１バイト）：このパケットを送信する前の同期秒内に経過したスロット（０．．１２４）の数。

【０１３４】

ＳｅｎｄＰｒｏｂ７０８（２バイト）：送信パケット毎に課されるランダム遅延を計算するために使用されるだろう１６ビット数。

データパケット

【０１３５】

図５（ｂ）は、例示的なデータパケット７１０のグラフィック描写である。データ送信アップストリームは、そのような、それぞれが単一スロット以内で送信されるのに十分小さいＭＴＵサイズのパケットに分割されるだろう。ダウンストリームデータは、同じパケットフォーマットを使用するだろうが、ダウンストリームＭＴＵサイズは、より大きいだろう。データパケット以下のものを含むだろう。すなわち、

【０１３６】

ＰａｃｋｅｔＴｙｐｅ７２２（１バイト）：有効なＴＭパケットを識別して、それが４つのタイプのいずれであるかを特定するであろう定数。データパケット７１０に対して、ＰａｃｋｅｔＴｙｐｅ値７１２は０ｘＡ９である。

【０１３７】

ＣｏｎｎＩＤ７１４（１バイト）：すべての仮想接続と関係付けられた仮想ソケットに対応するユニークな接続ＩＤ。

【０１３８】

ＳｅｑＮｕｍ７１６（１バイト）：このフィールドの値は、メッセージ（フラグフィー

ルドを参照のこと）内のパケットの位置に依存するだろうし、次の如くである。すなわち、

【０１３９】

（１）もし（ＩＦ）、これがメッセージ内の最初のパケットでありパケット数が＞１ならば、この値がメッセージ（１．．２５６）マイナス１内のパケットの全数である（これは、最大アップストリームメッセージサイズが２５６＊（ＭＴＵ－４）、または、６１，４４０バイトであることを意味する）。

【０１４０】

（２）それ以外で、もし(ＥＬＳＥ・ＩＦ)、これがメッセージ内の最後のパケットであれば、この値がこのパケットのペイロードのサイズ（１．．ＭＴＵ－４）マイナス１を定義する（これは、受信が、最後のパケットを受信するまで、実際のメッセージサイズを知らないが、パケットカウントに基づく推定が多くてもＭＴＵ－５バイト、または、０．３９％を浪費することを意味する）。

【０１４１】

（３）それ以外であれば(ＥＬＳＥ)、この値はパケットのシーケンス数マイナス１である。

【０１４２】

Ｆｌａｇ７１８（１バイト）：以下のビットフラッグを含有するビットベクトル。すなわち、

【０１４３】

ＦｉｒｓｔＰａｃｋｅｔ：もし、これがメッセージ内の最初のパケットであれば、（例えば、＝１）に設定する。

【０１４４】

ＬａｓｔＰａｃｋｅｔ：もし、これがメッセージ内の最後のパケットであれば、（例えば、＝１）に設定する。

【０１４５】

Ｒｅｓｅｎｄ：もし、これが、このパケットが送信される初回ではないならば、（例えば、＝１）に設定する。

【０１４６】

Ａｌｅｒｔ：もし、これがサーバー側送信元の「アラート」メッセージならば、（例えば、＝１）に設定する。

【０１４７】

残りの４フラッグビットは、将来使用のために予約されるだろう。

【０１４８】

サーバー送信元のメッセージ（すなわち、アラート）に対しては、ＩＰアドレスがＳＴＢ４００を識別するために使用されるだろう。ＣｏｎｎＩＤ６１４とＳｅｑＮｕｍ６１６用のヘッダースペースは、２バイトを提供するために組み合わされ、そこでは、ＴＭサーバー２１０がポート数を保存するだろう。このポートは、それから、ポートバインディングに基づいて、どの仮想接続がアラートを受信するかを識別するために、使用される。アラートメッセージは、Ｆｌａｇフィールド６１８内のＡｌｅｒｔビットを経由して識別される。全アラートメッセージは、（ダウンストリームＭＴＵ－５）バイト以内に収まらなければいけない。

【０１４９】

ＳｅｑＮｕｍ６１６の定義における「マイナス１」に関しては、例示的なメッセージが、いつも１個と２５６個のパケットの間で構成されているが、１個のバイトが０と２５５の間の整数を表わす。従って、ＳｅｑＮｕｍ値６１６は、いつもそれが表わす値より１小さい。

ＡＣＫパケット

【０１５０】

図５（ｃ）は、例示的なＡＣＫパケット７２０のグラフィック描写である。データパケ

ット７１０の受信が、これらのパケットの一つを返信することによりアクノリッジされるだろう。ＡＣＫパケット７２０は以下のものを含むだろう。すなわち、

【０１５１】

ＰａｃｋｅｔＴｙｐｅ７２２（１バイト）：有効なＴＭパケットを識別して、４つのタイプのどれであるかを特定するだろう定数。ＡＣＫパケット７２０に対して、ＰａｃｋｅｔＴｙｐｅ値７２２は０ｘＡＡである。

【０１５２】

ＣｏｎｎＩＤ７２４（１バイト）：受信されたデータパケット７１０からのＣｏｎｎＩＤ値７１４。

【０１５３】

ＳｏＦａｒＣｔ７２６（１バイト）：最初の欠落パケット以前に、これまでに受信されたメッセージのパケット数。

【０１５４】

ＡＣＫビット７２８（１バイト）：パケット＃ＳｏＦａｒＣｔ後の、８パケットまでのアクノリッジメント。全ての送信されたデータと同様に、それは、高位のビットによりアクノリッジされたＳｏＦａｒＣｔ後の最初のパケットおよび低位のビットによりアクノリッジされたＳｏＦａｒＣｔ後の８番目のパケットと共にネットワークバイト順であろう。

【０１５５】

ＳｅｎｄＰｒｏｂ７２９（２バイト）：送信パケット毎に課されるランダム遅延を計算するために使用される１６ビット数。

プローブパケット

【０１５６】

図５（ｄ）は、例示的なプローブパケット７３０のグラフィック描写である。通常、メッセージ内の全データパケット７１０は、最後のパケットを除いて、最大サイズのペイロード（ＭＴＵ－６）を含有しなければならない。これが、パケットサイズを推定することを容易にする。しかしながら、負荷情報が利用できない時は、メッセージの最初のパケットが、対応するＡＣＫ内の負荷情報を得るために、プローブとして送信される必要があるだろう。プローブパケット７３０は、望ましくは、小さいだろうし、一つのＭＡＣセルペイロード（３４バイト）以内に収まるだろう。従って、プローブパケット７３０は、（ＰａｃｋｅｔＴｙｐｅ＝０ｘＡＢ付きであるが）ペイロード無しのメッセージ７３２の最初のデータパケットと同じ６バイトのヘッダー情報を含有する。ＳｅｑＮｕｍ値７３６は、後続の１～２５６データパケットのみをカウントして、任意のバッファー長計算において、プローブをパケットとしてカウントしない（予測ネットワーク負荷におけるプローブデータの更なる使用の記述に対するＴＭサーバーセクションを参照のこと）。

ＴＭクライアントデータ

【０１５７】

図６（ａ）～図６（ｃ）は、記述されたＡＰＩをサポートするためにＴＭクライアント４２０の一実施形態において使用されるだろう構造を描写している。ＴＭクライアント４２０（図には描写されていない）により利用されるであろう、それらとその他のデータタイプと変数は、次の如くであろう。すなわち、

【０１５８】

（１）図６（ａ）に描写されるように、仮想ソケット構造８１０は、仮想ストリームソケットを実装するために必要とされる全情報を含有するだろう。これは、状態情報８１２、ポートバインディング８１３、非同期コールバック情報８１４、リモートホストＩＰ８１５とポート８１６、および、プロセスされる現在メッセージ８１７についての情報を含むだろう。

【０１５９】

（２）ｔｍ＿ｓｏｃｋｅｔ変数は実際のデータグラムソケット記述子を含有して、着信メッセージをリッスンするために使用されるポートのＶＰＯＲＴにバインドされるだろう。このソケットは、もし可能ならば、クライアントノードエージェントまたはアプリケーシ

ョン（例えば、ＥＢＩＦユーザエージェント４１０）がアクティブである限りオープンのままであるべきである。そうでなければ、ソケットはオープンされて、必要に応じて初期化されるだろう。一般的に、このソケットは標準ソケットＡＰＩを経由して作成される。しかしながら、いくつかの実装では、他のソケット取得ＡＰＩが提供されるだろう。

【０１６０】

（３）ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ変数は、ｖｓｅｎｄ（）６１２を経由してパケット送信をスケジュールするために、ｖｒｅｓｅｒｖｅ（）６０６により使用される送信確率を表す値を含有するだろう。浮動小数点数（すなわち、０と１の間の値）として確率を表すよりはむしろ、この値は、一番近い整数へ切り上げた、送信確率と６５，５３６の積として、もっと便利に表されるだろう。この変数に含有される値が、情報７００とＡＣＫパケット７２０の両方内にてＴＭサーバー２１０により送出されるだろう。

【０１６１】

（４）図６（ｂ）に描写されているように、仮想ソケットテーブル８２０は、Ｓ仮想ソケット構造８１０を含有するテーブル（すなわち、アレイやリストなど）であろうし、そこでは、Ｓは利用可能なスペースに基づいた実装固有であるが、現在の実施形態に従って最大で１６である。ソケットＩＤは、ＴＭパケットヘッダー内のＣｏｎｎＩＤ（すなわち、接続ＩＤ）フィールド７１４、７２４、７３４であるように、このテーブル内へのインデックス（例えば、１．．Ｓまたは０．．Ｓ－１）である。ソケットＩＤは、また、仮想ソケット記述子として知られるだろう。

【０１６２】

（５）ｖｅｒｒｎｏ変数が、プログラマーにエラーをレポートするために、ＴＭ仮想ソケットにより使用されるだろう。この変数は、標準ソケットＡＰＩにより使用されるｅｒｒｎｏ変数に加えられるだろう。

【０１６３】

（６）図６（ｃ）に描写されているように、予約待ち行列８００は、送信するためのパケットを持つ仮想ソケット８２０をそれぞれが識別する予約のアレイ（待ち行列）であろう。新しいパケットが予約待ち行列８００に加えられるか、または、待ち行列内の最初のパケットが送信されるたびに、待ち行列内の次回のパケットを送信する時間が計算されて、ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔに保存されるだろう。この計算は、以下において、ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔの定義にて与えられ、そこでは、ｃｕｒｒｅｎｔ＿ｓｌｏｔ＿ＩＤは、計算時のＳｌｏｔＩＤを指す。

【０１６４】

（７）ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔ変数はＳｌｏｔＩＤ（または、それと等価）だろうし、そこでは、次回のパケットが送信されることになる（予約待ち行列を参照のこと）。新しいｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙが、ＡＣＫ７２０または情報パケット７００のいずれかを介してクライアントに到達するたびに、ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔの値が再計算されなければならない。

ｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔ＝ｃｕｒｒｅｎｔ＿ｓｌｏｔ＿ＩＤ＋（ｔｍ＿ｒａｎｄｏｍ（）　％

（１３１０７２．０／ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ＋０．５））

【０１６５】

（８）ｔｉｍｅ＿ｏｆｆｓｅｔ＿ｍｓｅｃ変数は、タイムオフセットを示すだろうし、それは、クライアントノード（例えば、ＳＴＢ）上の内部クロックタイム（ＧＭＴ）に付加される時に、時刻を、ＴＭサーバー２１０およびその他のクライアントノードと同期させる。ＳｌｏｔＩＤを正確に計算するため、および、要求されるだろうその他の能力のために、正確な時刻が必要とされる。

メッセージの保証された配信

【０１６６】

以下の段落では、メッセージの保証された配信を実施する一実施形態について、さらなる詳細が提供される。まず、予約が特定の仮想ソケットに対して既に存在しており予約さ

れたＳｌｏｔＩＤが到達したと仮定して、モード１のコンテキストにおけるｖｓｅｎｄ（）６１２について、詳細が提供される。そして、その他のモードにとって必要であろう追加の記述がなされる。

【０１６７】

上記したように、ｖｓｅｎｄ（）６１２は、そのＢＳＤソケットカウンターパートやｓｅｎｄ（）５０６のように、一つの呼出にて全メッセージを送信することを保証されていないので、それは、メッセージ（または、部分メッセージ）が送信されるまで、ブロックするでしょう。これは、ｓｅｎｄ（）５０６の問題ではないであろう。しかし、本発明のさまざまな実施形態は、スロット衝突を回避するように経時的にメッセージパケットを広めるので、アプリケーションが、返答するためにｖｓｅｎｄ（）６１２を待っている多くの秒期間に、ブロックされ得る。このため、もしデータパケットのための予約タイムが長いならば、ｖｓｅｎｄ（）６１２が、呼出元に、制御を返すでしょう。全メッセージが送信されない時に、ｖｓｅｎｄ（）６１２は、現在のｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ値に基づいて、次回のｖｓｅｎｄ（）６１２呼出のための予約を作成するだろう。

【０１６８】

一つの例示的な実施において、ｖｓｅｎｄ（）６１２は、メッセージを（ＭＴＵ－５）バイト（または、最後のパケットに対してはより少ないバイト）のパケットに分解して、ｖｒｅｓｅｒｖｅ（）６０６によってスケジュールされるように、それらのパケットを送信する。ｖｓｅｎｄ（ｖｓｏｃｋｅｔ＿ｄｅｓｃｒｉｐｔｏｒ、ｍｅｓｓａｇｅ＿ｐｔｒ、ｍｅｓｓａｇｅ＿ｌｅｎｇｔｈ）をプロセシングすることが、次の如く生じるであろう。すなわち、

【０１６９】

（１）ｄａｔａ＿ｓｅｎｔ＝－１を設定し、ｒｅｓｅｎｄ＿ｃｏｕｎｔ＝０を設定する。

【０１７０】

（２）ｗａｉｔ＿ｔｉｍｅ＝ｖｒｅｓｅｒｖｅ（ｖｓｏｃｋｅｔ＿ｄｅｓｃｒｉｐｔｏｒ）を設定する。

【０１７１】

（３）もし（ＩＦ）、ｗａｉｔ＿ｔｉｍｅ＞ＶＳＥＮＤ＿ＷＡＳＴＥならば、ｖｅｒｒｎｏ＝ＶＳＥＮＤＬＡＴＥＲを設定し、ｄａｔａ＿ｓｅｎｔを返す。

【０１７２】

（４）もし（ＩＦ）、ｒｅｓｅｎｄ＿ｃｏｕｎｔ＞＝ＶＳＥＮＤ＿ＴＲＩＥＳならば、ｖｅｒｒｎｏ＝ＶＭＡＸＲＥＳＥＮＤＳを設定し、ｄａｔａ＿ｓｅｎｔを返す。

【０１７３】

（５）もし（ＩＦ）、ｗａｉｔ＿ｔｉｍｅ＞０ならば、ｗａｉｔ＿ｔｉｍｅミリ秒の期間スリープし、それから、ステップ２に戻る。

【０１７４】

（６）もし（ＩＦ）、ｖｓｏｃｋｅｔ＿ｄｅｓｃｒｉｐｔｏｒに対して現在進行中のメッセージが送信（ｓｅｎｄ）状態であり、そのメッセージバッファーがｍｅｓｓａｇｅ＿ｐｔｒ［ｍｅｓｓａｇｅ＿ｌｅｎｇｔｈ］を含有しているならば、スキップしてステップ８に進む。

【０１７５】

（７）仮想ソケットテーブル（図６（ｂ）と関係付けられた議論を参照）内のｖｓｏｃｋｅｔ＿ｄｅｓｃｒｉｐｔｏｒに対するメッセージプログレスデータを初期化する。

【０１７６】

（８）もし（ＩＦ）、メッセージの全てのパケットが、未だ一度も送信完了していなければ、送信するために次回のパケットを構築し、それに応じて仮想ソケット構造を更新し、そして、スキップしてステップ１１に進む。

【０１７７】

（９）もし（ＩＦ）、全ての送信されたパケットが、未だアクノリッジ完了していなければ、最小数の否定応答されたパケットを再構築し、再送信（ｒｅｓｅｎｄ）フラッグビッ

トを設定し、それに応じてメッセージ構造を更新し、ｒｅｓｅｎｄ＿ｃｏｕｎｔをインクリメントし、そして、スキップしてステップ１１に進む。

【０１７８】

（１０）仮想ソケットのメッセージ情報を更新し、ｍｅｓｓａｇｅ＿ｌｅｎｇｔｈを返す。

【０１７９】

（１１）次回のＳＬＯＴ境界を待って、必要ならば、（再）構築されたパケットを送信する。

【０１８０】

（１２）ｄａｔａ＿ｓｅｎｔ＝ＳｏＦａｒＣｔ＊（ＭＴＵ－４）を設定して、ステップ２に戻る。

【０１８１】

ｖｓｅｎｄ（）６１２のこの定義は、ｖｓｅｎｄ（）６１２がプロセシングしている間に、情報７００とＡＣＫパケット７２０の両方の受領が、非同期信号割り込みを介して取り扱われることを前提としている。特に、それは、情報パケット７００がｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ７０８とｒｅｓｅｒｖａｔｉｏｎ＿ｓｌｏｔを更新すること、および、ＡＣＫパケット７２０がＳｏＦａｒＣｔ７２６とＣｏｎｎＩＤ７２４によって定義された仮想ソケット構造８１０におけるＡＣＫビット７２８フィールドを更新することとを前提としている。すべてのデータパケットが、ＡＣＫパケット７２０によって、アクノリッジされる必要はない。実際に、ＡＣＫ７２０の頻度は、サーバー負荷（すなわち、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ）における変化率とアウトオブシーケンスパケットの受信比率に反比例するだろう。理想的には、軽い負荷の下で、ＡＣＫ７２０は、４番目の受信パケット毎に送信されるだろう。さまざまな実施形態において、唯一の要求は、ＡＣＫパケット７２０がメッセージの最初と最後のパケットの両方に対して発生されなければならないことであろう。

【０１８２】

モード２において、仮想ソケットがｖｓｏｃｋｅｔ（）６０２でオープンされるまでｔｍ＿ｓｏｃｋｅｔが割り当てられずにポートのＶＰＯＲＴにバインドされることを除いて、上記変更は何もないし、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ７０８を定義するために情報パケット７００が受信されるまでは、ｖｒｅｓｅｒｖｅ（）６０６がブロックするでしょう。

【０１８３】

モード３において、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙがデフォルト送信確率値であると仮定すると、ｖｒｅｓｅｒｖｅ（）６０６、および／または、ｖｓｅｎｄ（）６１２は、メッセージに対してプローブパケット７３０を送信しなければならないし、先に進む前にプローブ（実際の送信確率値を提供している）に対してＡＣＫパケット７２０を受信しなければならない。短いタイムアウト（例えば、３秒）以前のこのプローブのＡＣＫ７２０の受信失敗は、呼出を失敗させて、ｖｅｒｒｎｏにて適切なエラーを設定する。

仮想ソケット

【０１８４】

以下の段落では、仮想ソケット実装の一実施形態がさらに記述される。

【０１８５】

本発明のさまざまな実施形態によるトラフィックマネージメントは、単一のデータグラムソケット（すなわち、ｔｍ＿ｓｏｃｋｅｔ記述子）を使用して以下の要求を満足するように努めるだろう。すなわち、

【０１８６】

（１）複数のストリームソケットの効果を模擬する、

【０１８７】

（２）ＴＭサーバー２１０からのブロードキャストである情報パケット７００をリッスンする、および、

【０１８８】
（３）アプリケーションサーバーからクライアントノード（例えば、ＳＴＢ４００）へのユニキャストであるアラートメッセージをリッスンする。これは、全ての３つのモードにおいてトゥルーである。相違は、いかに長いデータグラムソケットがリッスニングのスタート上で保持されるかだけである。
【０１８９】
ｖｓｅｎｄ（）６１２とｖｒｅｃｖ（）６１６以外のＴＭクライアントＡＰＩは、ｔｍ＿ｓｏｃｋｅｔで何もしないであろう。その代わりに、それは、データパケットを送信または受信する時に後続して使用される仮想ソケットテーブル８２０（または、予約待ち行列８００）にて、情報を修正するだろう。例えば、ＡＣＫパケット７２０またはデータパケット７１０が受信される時に、パケット内のヘッダーフィールドは、どのソケットが、そして、これによりどのコールバック関数が実際のパケットの受領者であるかを決定するために、仮想ソケットテーブル８２０にてデータとマッチングされるだろう。
【０１９０】
割り当てられたｔｍ＿ｓｏｃｋｅｔがＶＰＯＲＴ（すなわち、１９６２）にバインドされ、信号メカニズムを経由して着信トラフィックに対して「リッスンする」。着信パケットのｐａｃｋｅｔ＿ｔｙｐｅフィールド７０２、７１２、７２２、７３２が、非ＴＭパケットをフィルター除去するため、および、受信されるだろうさまざまなパケットタイプを適切に取り扱うための両方のために、検査される。情報パケット７００は、トラフィックレポーティングとクライアントクロック同期セクションに開示されているように、ＴＭサーバー２１０とＴＭクライアント４２０により内部的に取り扱われるだろう。ＡＣＫ７２０とデータ７１０パケット（アラート以外）は、パケットが所属する仮想ソケット接続を識別するＣｏｎｎＩＤフィールド７１４、７２４（仮想ソケットテーブルへのインデックス）を含有するだろう。アラートメッセージは、Ｆｌａｇフィールドビットと Ａｌｅｒｔにより識別されるだろうし、ＣｏｎｎＩＤとＳｅｑＮｕｍビットは、２バイトポート数を定義するために組み合わされて、そのポート数にバインドされた仮想ソケットがメッセージの受領者である。
【０１９１】
ＴＭクライアント４２０は、ｔｍ＿ｓｏｃｋｅｔのソケット上のアクティビティ（またはエラー）がある時に、さまざまなＴＭ内部ルティーンを呼び出すために、非同期Ｉ／Ｏ信号通信（ＳＩＧＩＯ）に依存するだろう。例えば、（ｖａｓｙｎｃｈ（）６０５を経由して）仮想ソケットと関係付けられたＴＭ非同期コールバック関数は、信号取扱者により呼出されるだろう。
【０１９２】
データの送信および受信のためのさまざまな実施形態が、メッセージセクションの保証された配信において、より詳細に記述される。
メッセージデータ圧縮
【０１９３】
ケーブルＯＯＢネットワーキングコンテキストの一つの特異性は、９９％のトラフィックがＨＴＴＰメッセージからなるであろうことである。これにより、シンプルな静的圧縮方法が、著しい節約を生み出すことができる。静的テーブルベースのスキームは、以下と同様に、トランスポートに先立って全メッセージを圧縮するために採用されるだろう。解凍は簡潔であるべきである。

```
ｃｈａｒ＊ｓｔｒｉｎｇ［１２８］＝｛
　　「ＨＴＴＰ」，　　／＊他のＨＴＴＰキーワードを加える＊／
　　「コンテンツ長」，　／＊他のヘッダーストリングを加える＊／
　　「＆ａｍｐ；」，　　／＊他のエンティティを加える＊／
　　「ａｍｒ」，　　／＊他のＩＡＭキーワードを加える＊／
　　「￥ｒ￥ｎ」，／＊他の共通のマルチ文字ストリングを加える＊／
　　／＊１２３以上のストリングを必要とする＊／
```

```
                ／＊一度、全ての前記ストリングが定義される、ａ＊／
                ／＊高速スイッチベースのルックアップメカニズム＊／
                ／＊実装できる＊／
},

static  char output[MAX_MSG];
static int outlen;

void compress(char*message, int inlen) {
    int pos, code;

    outlen=0;
    for (pos=0; pos<inlen; code<0?pos++:0
) {
        for (code=0; code<128; code++) {
            if ( (code[0]==message[pos])
         && !strncmp(&(message[pos]), stri
ng[code], strlen(string[code]))
            ) {
          output[outlen++]=(char) -code;
          break;
            }
        }
    }
    ／＊結果は長さを有する‘output’である、‘outlen’＊／
}
```

【０１９４】

この圧縮スキームは、２進数データを破損しないよう意図されている。これにより、それは、また、必ずしもテキストデータを送信しないであろう非ＨＴＴＰプロトコルに対して適切だろう。

送信確率を計算する

【０１９５】

以下の段落では、送信確率値を計算する例示的な方法が記述される。

【０１９６】

ＴＭクライアント４２０との実際のコミュニケーションを取り扱うＴＭサーバー２１０と帯域幅調整コード間のインターフェースが、以下のように、単一のＡＰＩ呼出を使用することにより実施されるだろう。すなわち、

【０１９７】

```
int got_packet(int size, int isOutOfOrde
r)
```

【０１９８】

ＴＭサーバー４２０が任意のクライアントからパケットを受信するたびに、それは、ｇｏｔ＿ｐａｃｋｅｔ（）を直ちに呼出すべきである。パケットのサイズ（すなわち、ヘッダー＋ペイロード）は、サイズパラメータで渡されるだろう。もし、パケットのシーケンス数が期待値（すなわち、前回のシーケンス数＋１）でなければ、その時は、非ゼロ（トゥルー）値がｉｓＯｕｔＯｆＯｒｄｅｒで渡されるだろうし、そうでなければ、０（フォールス）値が、このパラメータで渡されるだろう。

【０１９９】

ｇｏｔ＿ｐａｃｋｅｔ（）のリターン値は、０か、または、正の非ゼロの整数かのいずれかだろう。もしリターン値が０であるならば、さらなるアクションは要求されない。し

かしながら、もし値が非ゼロであるならば、リターン値は新しい送信確率として使用されるべきであり、ＡＣＫ７２０と情報パケット７００を経由して全ＴＭクライアント４２０にコミュニケーションされるべきである。

【０２００】

一実施形態において、送信確率値は、（ＨＦＣノード当たりの）ターゲット帯域幅と直近の将来に対する（ＨＦＣノード当たりの）期待帯域幅との間の比であろう。ターゲット帯域幅は、単に、最大直下の結果を保持するために小さい調節因子（例えば、０．９）を乗算した、（ＨＦＣノード当たりの）構成最大帯域幅である。期待帯域幅は、さらなるパケットを送信するであろう各ＨＦＣノードにおける「アクティブ」ＳＴＢ（すなわち、クライアントノード）の数である。これは、２＊Ｐ／ＳＰｏｌｄとして推定されるだろうし、そこでは、ＰはＳＬＯＴ当たりの受信されたパケットの計算平均数であり、ＳＰｏｌｄは送信確率の前の値である。

【０２０１】

また、調節因子、Ｋ、は、システム構成値と時折の過渡効果における不正確さを考慮して、定期的に計算されるだろう。

情報パケットのブロードキャスト頻度

【０２０２】

以下の段落では、情報パケット７００のブロードキャスト頻度を決定する例示的な方法が記述される。

【０２０３】

情報パケット７００発生の頻度は、ｇｏｔ＿ｐａｃｋｅｔ（）のリターン値により御されるだろう。この関数が非ゼロ値を返すたびに、新しい情報パケット７００がブロードキャストされるべきである。情報パケット７００ブロードキャストの頻繁すぎか、または、擬似的な発生を防ぐために、注意が払われるべきである。

【０２０４】

具体的には、ｇｏｔ＿ｐａｃｋｅｔ（）が、下記のたびに、別の情報パケットブロードキャストを経由して展開するように、非ゼロｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙ値を返すであろう。すなわち、

【０２０５】

（ａ）最後の情報パケットブロードキャストから１０秒が経過した、または、

【０２０６】

（ｂ）最後の情報パケット情報パケット７００ブロードキャストから少なくとも０．５秒が経過し、アウトオブオーダパケットが見られる、または、

【０２０７】

（ｃ）最後の情報パケットブロードキャストから少なくとも０．５秒が経過し、パケット／第二レートが増加傾向である。

【０２０８】

そうでなければ、ｇｏｔ＿ｐａｃｋｅｔ（）が、ゼロ（０）を返すだろうし、ｓｅｎｄ＿ｐｒｏｂａｂｉｌｉｔｙの前の値が有効のままである。

帯域幅調整のシミュレーション

【０２０９】

このセクションは、本発明の一実施形態による帯域幅調整方法の一実施形態のシミュレーションの記述を提示する。どのようにシミュレーションが行われたかの記述が、図７（ａ）～図７（ｃ）に描写されているように、結果の記載と共に提示される。

【０２１０】

シミュレーションはＱＰＳＫネットワークのモデリングを含み、そこでは、ＴＭサーバー２１０が、ネットワークを経由して、複数のＴＭクライアント４２０とコミュニケーションする。ＱＰＳＫネットワークモデルが、アップストリームとダウンストリームスループットをモニターし、帯域幅制限を越えるいずれかの方向のパケットを捨てる。

【０２１１】

スループットが理論的帯域幅制限に近づくにつれて、パケット損失の確率は１に近づく。理論的最大パケットスループットアップストリームを（１／ｅ）＊２５６，０００ｂｐｓ＝９４，０００ｂｐｓとし、全アップストリームパケット（ＴＭヘッダーと２８バイトのＵＤＰ／ＩＰヘッダーを含む）の総サイズが、Ｓバイト（または、ｓ＝Ｓ＊８ビット）と仮定すると、その時は、失われて（そして完全に無視される）次回のパケットの確率が、Ｐｐａｃｋｅｔ＿ｌｏｓｓ（ｓ）＝（ｓ／９４，０００）として与えられる。プログラム的に、以下のＩＦ条件が、これをテストする。すなわち、

もし（ｉｆ）　（（ｔｍ＿ｒａｎｄｏｍ（）％９４０００）＜ｓ）であれば、｛

```
    ；／＊パケットが失われる＊／
  }  それ以外であれば（ｅｌｓｅ）　｛
    ；／＊パケットをプロセスする＊／
  }
```

【０２１２】

図７（ａ）～図７（ｃ）に描写されたシミュレーション結果において、大きな（構成可能）数のＳＴＢ４００とＨＦＣノード３００が、構成可能なテスト期間（そこでは、１時間に設定）の各秒において、全１２５ＳＬＯＴに対してシミュレーションされた。

【０２１３】

図７（ａ）は、帯域幅調整無しのネットワーク負荷のシミュレーションを描写しているグラフであり、そこでは、（ＨＦＣノード３００当たりの）生の帯域幅負荷が、１時間シミュレーションの各秒で平均化される。

【０２１４】

図７（ｂ）は、帯域幅調整有りのネットワーク負荷のシミュレーションを描写しているグラフであり、図７（ａ）のように同じ生の帯域幅トラフィックにさらされている時の、ＨＦＣノード３００当たりの最大４０Ｋｂｐｓに対して、構成されている。尚、平均のトラフィック負荷は、ここでは、４０Ｋｂｐｓで制限され、高負荷の期間は、それぞれの場合において、数秒長い。

【０２１５】

図７（ｃ）は、帯域幅調整有りと無しのシミュレーションされたネットワーク負荷間の相違を描写しているグラフである。このグラフは、前の２つのグラフを重ねており、帯域幅マネージされたトラフィックに対する６０・秒動作平均を示す線を含む。この線は、本明細書に記載されているように、本発明のさまざまな実施形態の帯域幅調整態様の有益な影響を示している。

【０２１６】

本明細書中に開示された、実施例または実施例の一部分は、専用目的のコンピューター、マイコンを有するコンピューターシステム、ミニコン、または、メインフレーム、例えば、プログラムされたマイクロプロセッサ、マイクロ・コントローラー、周辺用集積回路要素、ＣＳＩＣ（顧客専用集積回路）またはＡＳＩＣ（アプリケーション専用集積回路）またはその他の集積回路、ロジック回路、デジタル信号プロセッサ、ＦＰＧＡやＰＬＤやＰＬＡまたはＰＡＬのようなプログラマブルロジックデバイス、または、本明細書中に記載されたプロセスの任意のステップを実施できる任意のその他のデバイスまたはデバイスの配置を含む、多彩な技術のいずれかを利用するだろう。

【０２１７】

本明細書中に記載された任意のオペレーションを実行するためにコンピューターオペレーテイングシステムを構成する一連のインストラクション（例えば、コンピューターソフトウェア）は、要望により、多彩なメディアまたは媒体のいずれかに含有されるであろうことが、理解されるべきである。さらに、一連のインストラクションによりプロセスされる任意のデータは、また、多彩なメディア又は媒体のいずれかに含有されるだろう。つまり、一連のインストラクションを保持するために利用される特別な媒体（例えば、メモリー）または本明細書中に記載された実施例において使用されるデータは、例えば、様々な

物理形式または送信のいずれかを呈するだろう。例示的には、媒体は、コンパクトディスク、ＤＶＤ、集積回路、ハードディスク、フロッピーディスク、光学ディスク、磁気テープ、ＲＡＭ、ＲＯＭ、ＰＲＯＭ、ＥＰＲＯＭ、配線、ケーブル、ファイバー、コミュニケーションチャンネル、サテライト送信または他のリモート送信、並びに、任意の他の媒体またはコンピューターにより読み取られるだろうデータのソースの形式であろう。

【０２１８】

コンピューター実行中の実行可能なコンピューターソフトウェアのような、本明細書中に記載されたさまざまなコンポーネントは、リモート的に設置されるだろうし、一つ以上のコンピュータネットワーク上で電子送信を介して互いにコミュニケーションするであろうことが、また、理解されるべきである。本明細書中で言及したように、ネットワークは、制限されるものではないが、広域ネットワーク（ＷＡＮ）、ローカルエリアネットワーク（ＬＡＮ）、インターネットのようなグローバルネットワーク、公衆電話交換回線網ネットワークのような電話ネットワーク、ワイヤレスコミュニケーションネットワーク、携帯電話ネットワーク、イントラネットなど、または、それらの任意の組み合わせを含むだろう。さまざまな実施形態において、ネットワークは、スタンドアロンネットワークとしてまたは互いに協力して動作している、一つまたは任意の数の、上述のネットワークの例示的なタイプを含むだろう。本明細書中の用語ネットワークの使用は、ネットワークを単一のネットワークに制限することを意図するものではない。

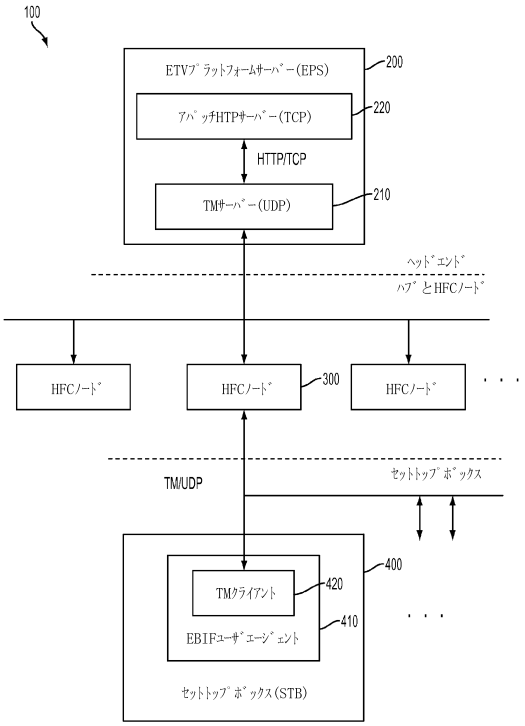
【０２１９】

本発明が、広い実用性と用途に敏感であることは、当業者により容易に理解されるでしょう。本明細書中に記載された以外の、本発明の多くの実施例と適応、並びに、多くの変更、修正および等価配置が、発明の本質または範囲から逸脱することなく、本発明とその前述の説明により、明らであるか、または、合理的に示唆されるでしょう。

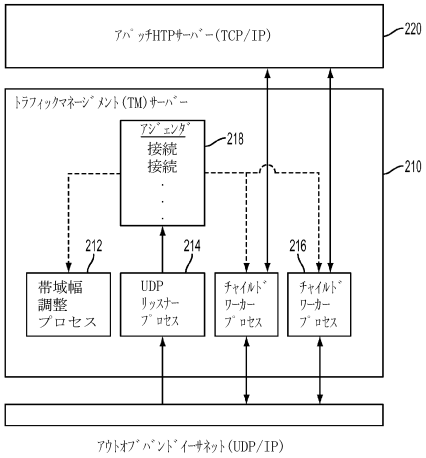
【０２２０】

前に述べたことは、この発明の実施形態を例示して記述しているが、本発明がそこにて開示された構成に限定されないことが、理解されるべきである。本発明は、その精神または本質的な属性から逸脱しないで、他の特定の形式において具体化できる。

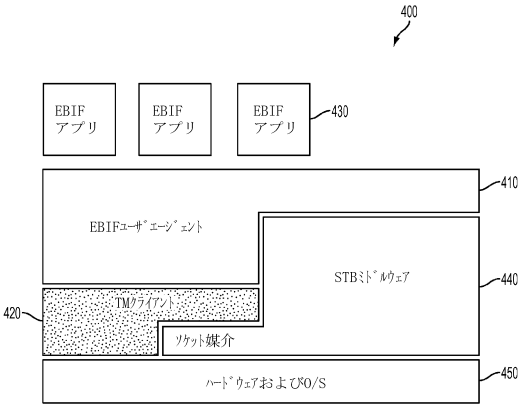
【図１】



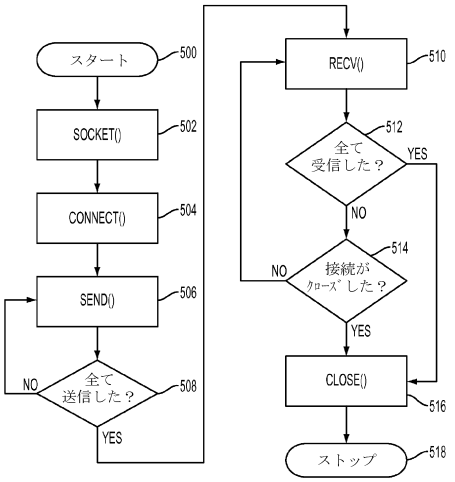
【図２】



【図３】

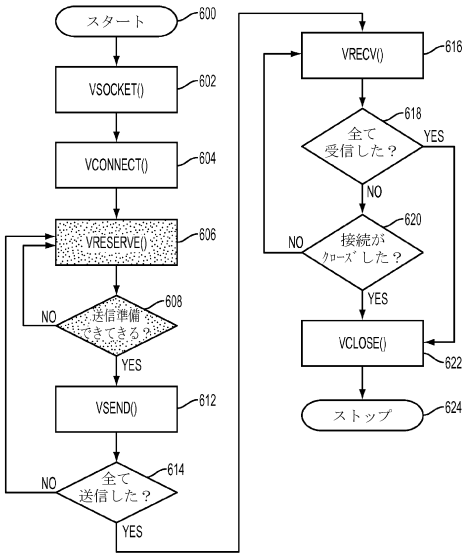


【図４（ａ）】



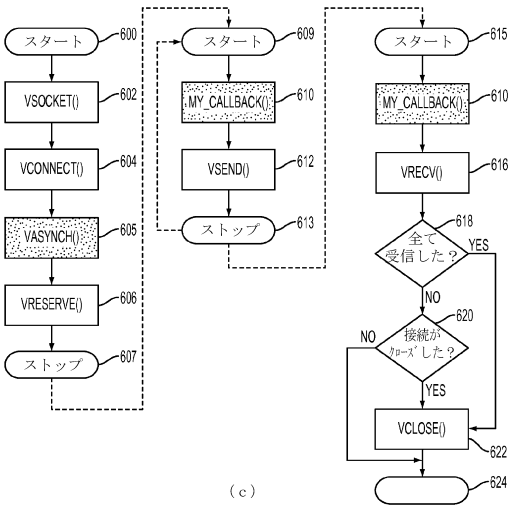
（ａ）
従来技術

【図４（ｂ）】



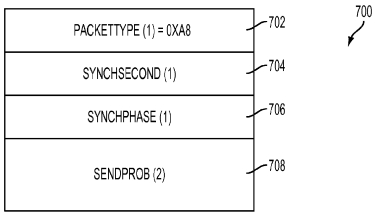
（ｂ）

【図４（ｃ）】



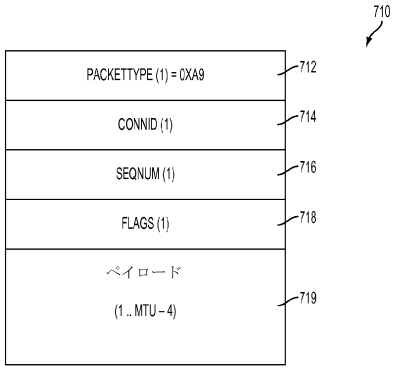
（ｃ）

【図５（ａ）】



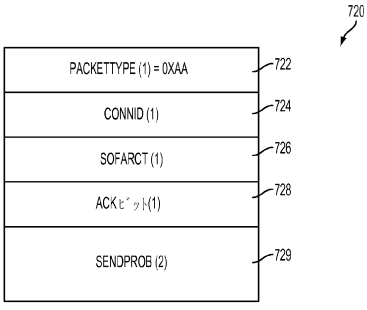
（ａ）

【図５（ｂ）】



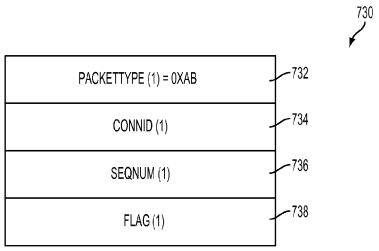
（ｂ）

【図５（ｃ）】



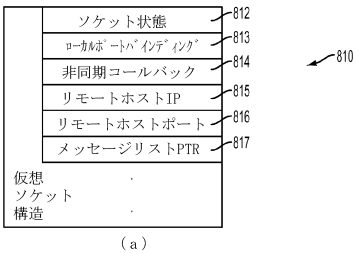
（ｃ）

【図５（ｄ）】



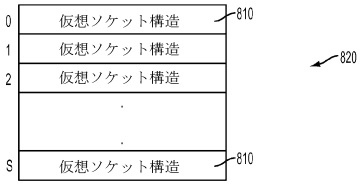
（ｄ）

【図６（ａ）】



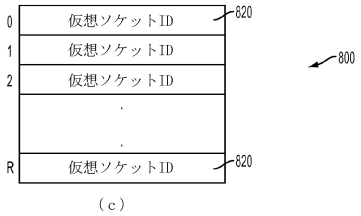
（ａ）

【図６（ｂ）】

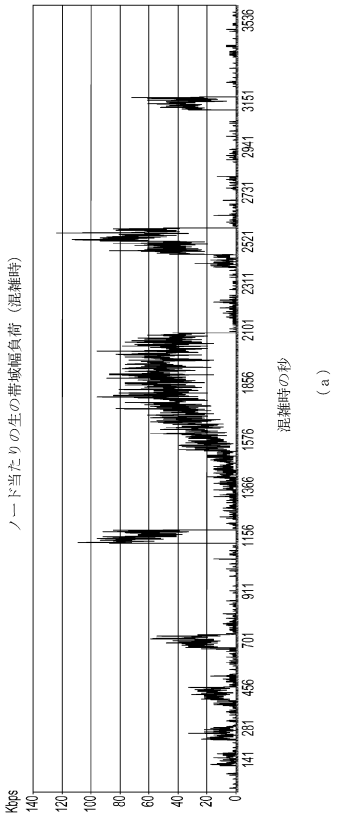


（ｂ）

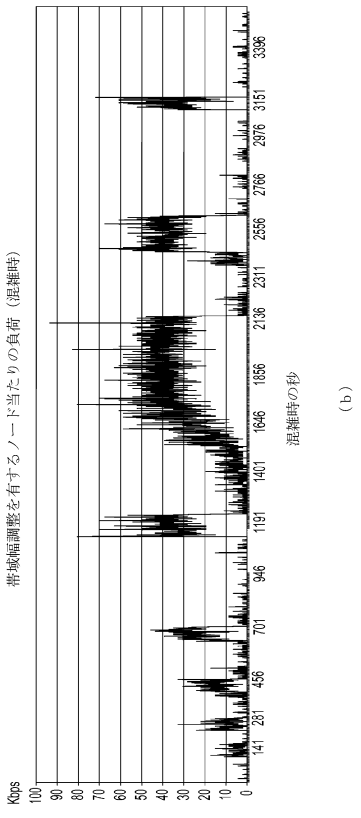
【図６（ｃ）】



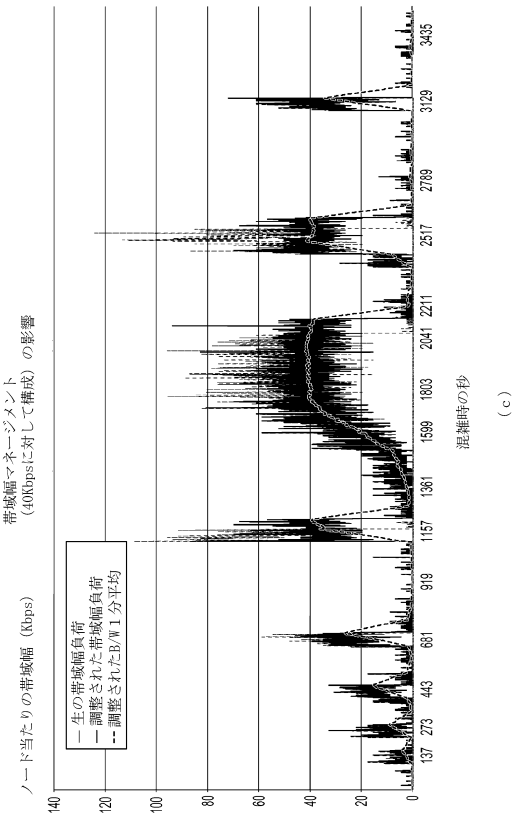
【図７（ａ）】



【図７（ｂ）】



【図７（ｃ）】



フロントページの続き

(56)参考文献 特開２００３－２９８６０４（ＪＰ，Ａ）
特開２０１０－２８３７５９（ＪＰ，Ａ）
米国特許出願公開第２００９／００７０４６８（ＵＳ，Ａ１）
百名 盛久 Morihisa Momona, ＣＡＴＶアクセスネットワークにおける技術課題と標準化動向 Technologies and Standardization Activities in Cable TV Access Networks, 電子情報通信学会技術研究報告 Ｖｏｌ．９８ Ｎｏ．５８９ IEICE Technical Report, 日本, 社団法人電子情報通信学会 The Institute of Electronics,Information and Communication Engineers, 第98巻, 第５７－６４頁

(58)調査した分野(Int.Cl., ＤＢ名)
Ｈ０４Ｌ１２／００－１２／２６，１２／５０－１２／９５５