(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4920220号
(P4920220)

(45)発行日　平成24年4月18日(2012.4.18)　(24)登録日　平成24年2月10日(2012.2.10)

(51)Int.Cl.		F I		
G06F 13/00	(2006.01)	G06F	13/00	351A
H04L 12/56	(2006.01)	H04L	12/56	230Z
H04N 7/173	(2011.01)	H04N	7/173	620Z
		H04L	12/56	260Z

請求項の数 26　(全 40 頁)

(21)出願番号　特願2005-251994 (P2005-251994)
(22)出願日　平成17年8月31日 (2005.8.31)
(65)公開番号　特開2006-79606 (P2006-79606A)
(43)公開日　平成18年3月23日 (2006.3.23)
審査請求日　平成20年9月1日 (2008.9.1)
(31)優先権主張番号　10/934,823
(32)優先日　平成16年9月3日 (2004.9.3)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　10/951,482
(32)優先日　平成16年9月28日 (2004.9.28)
(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　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　チン　リー
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】ピアツーピアネットワークでの受信側主導のシステム及び方法

(57)【特許請求の範囲】

【請求項１】

ピアツーピア（Ｐ２Ｐ）ネットワークでのマルチメディアデータパケットのクライアントドリブンのストリーミングを提供するコンピュータ実行可能命令を記録するコンピュータ読み取り可能な記録媒体において、

前記コンピュータ実行可能命令は、

クライアントコンピュータ上で複数のクライアントリクエストキューを保持することであって、各クライアントリクエストキューはサービングピアのクラスタ内の複数のサービングピアのうちの１つに対応すること、

１つ又は複数のデータパケットのクライアントリクエストを前記クライアントコンピュータから前記サービングピアのうちの１つ又は複数に送信することであって、サービングピアへの前記クライアントリクエストは、ＴＣＰ通信プロトコルを介して提供されるため、前記サービングピアが前記クライアントリクエストに応答して送る前記データパケットを識別する必要がないこと、

データパケットリクエストが前記クライアントコンピュータから前記サービングピアの一つに送信される場合、各データパケットリクエストを前記対応するリクエストキューに追加すること、

前記クライアントコンピュータが前記サービングピアから前記対応するデータパケットを受信する場合に、各データパケットリクエストを前記対応するリクエストキューから取り出すこと、

前記クライアントコンピュータが保持する共通ステージングキューに各受信データパケットを提供すること、
前記共通ステージングキュー内の前記データパケットを対応するマルチメディアデータパケットに集めること、
を含むことを特徴とするコンピュータ読み取り可能な記録媒体。
【請求項２】
クライアントコンピュータ上で複数のクライアントリクエストキューを保持することは、前記サービングピアのクラスタに続いて加わる追加のサービングピアに対応する追加のクライアントリクエストキューを作成すること及び保持することを更に含むことを特徴とする請求項１記載のコンピュータ読み取り可能な記録媒体。
【請求項３】
クライアントコンピュータ上で複数のクライアントリクエストキューを保持することは、前記サービングピアのクラスタから除外されたサービングピアに対応するリクエストキューに残存するデータパケットリクエストを１つ又は複数の他のクライアントリクエストキューに移すことを更に含むことを特徴とする請求項１記載のコンピュータ読み取り可能な記録媒体。
【請求項４】
前記サービングピアのクラスタ内の各サービングピアにおいて大体一致するリクエスト達成時間（ＲＦＴ）を保持するデータパケットリクエストのクライアントマネージメントを提供することにより、前記サービングピアのクラスタ内の前記各サービングピアにおいて負荷バランシングを維持することを特徴とする請求項１記載のコンピュータ読み取り可能な記録媒体。
【請求項５】
前記クライアントコンピュータがサービングピアから受信する入力パケットは、前記データパケットを提供した前記サービングピアに対応する前記リクエストキュー内の第１のデータパケットリクエストに対応することを特徴とする請求項１記載のコンピュータ読み取り可能な記録媒体。
【請求項６】
特定のサービングピアへの特定のデータパケットのクライアントリクエストは、前記各サービングピアから取り出された可用性ベクトルのクライアント分析に基づいてなされ、各サービングピアから受信される前記可用性ベクトルは、対応する各サービングピアに格納される利用可能なデータパケットを少なくとも定義することを特徴とする請求項１記載のコンピュータ読み取り可能な記録媒体。
【請求項７】
前記クライアントコンピュータ上で前記集められたマルチメディアデータパケットをデコーディングしレンダリングし、前記クライアントコンピュータ上でストリーミングメディア再生を提供することを更に含むことを特徴とする請求項１記載のコンピュータ読み取り可能な記録媒体。
【請求項８】
前記データパケットは、埋め込みコード化データパケットであることを特徴とする請求項１記載のコンピュータ読み取り可能な記録媒体。
【請求項９】
１つ又は複数のデータパケットの前記クライアントリクエストを前記サービングピアのうちの１つ又は複数に自動的に制限することによって、埋め込み各コード化データパケットのストリーミングビットレートでクライアント制御を維持することを更に含み、前記制限は、
前記サービングピアの集合サービング帯域幅、
前記共通ステージングキューのサイズ、
所望のリクエスト達成時間、
前記共通ステージングキュー内の受信パケット長、

各リクエストキュー内の未リプライの長さ、
前記埋め込み符号化データパケットの基層ビットレート
に応じることを特徴とする請求項８記載のコンピュータ読み取り可能な記録媒体。
【請求項１０】
ピアツーピア（Ｐ２Ｐ）ネットワークにおけるメディアファイルのクライアント制御ストリーミングを提供する方法において、コンピューティングデバイスを用いて、
メディアヘッダとメディア本体を備える符号化されたメディアファイルの１つ又は複数のパケットを利用可能なサービングピアのクラスタ内の１つ又は複数のサービングピア上に格納し、これにより、前記符号化されたメディアファイルが前記サービングピアの少なくとも一つにキャッシュされるようにすること、
前記クライアントを使用し、前記クライアントが利用可能な複数のサービングピアのリストを取り出すこと、
前記クライアント上で複数のパケットリクエストキューを保持することであって、各パケットリクエストキューは、サービングピアのクラスタ内の複数のサービングピアのうちの１つに対応すること、
データパケットリクエストが前記クライアントから前記サービングピアの一つに送信される場合、各データパケットリクエストを前記対応するパケットリクエストキューに追加すること、
前記クライアントが前記サービングピアから前記対応するデータパケットを受信する場合、前記対応するパケットリクエストキューから各データパケットリクエストを取り出すこと、及び、
前記受信されたデータパケットをデコーディングしレンダリングし、前記クライアント上でストリーミングメディア再生を提供することを特徴とする方法。
【請求項１１】
可用性ベクトルを各利用可能なサービングピアから前記クライアントにダウンロードすることを更に備え、各可用性ベクトルは、各対応するサービングピアに格納される利用可能なデータパケットを表し、１つ又は複数の特定のデータパケットに対するクライアントリクエストは前記ダウンロードされた可用性ベクトルに基づくことを特徴とする請求項１０記載の方法。
【請求項１２】
前記受信データパケットをデコーディングしレンダリングすることは、前記クライアント上で再生する前に、前記デコードされレンダリングされたデータパケットの少なくとも一部をバッファリングすることを更に含むことを特徴とする請求項１０記載の方法。
【請求項１３】
前記クライアント上で複数のパケットリクエストキューを保持することは、前記サービングピアのクラスタに加わるサービングピアに対応する追加のパケットリクエストキューを作成し維持することを更に含むことを特徴とする請求項１０記載の方法。
【請求項１４】
前記クライアントが利用できなくなるサービングピアに対応する前記パケットリクエストキューに残存するデータパケットを１つ又は複数の前記他のパケットリクエストキューに移すことと、前記対応するサービングピアに前記対応するパケットを要求することを更に含むことを特徴とする請求項１０記載の方法。
【請求項１５】
前記利用可能なサービングピアの各々において所望のリクエスト達成時間（ＲＦＴ）を維持するためのデータパケットリクエストの動的クライアントマネージメントを提供することによって、前記利用可能な各サービングピア間の帯域幅負荷バランシングを実行することを更に含むことを特徴とする請求項１０記載の方法。
【請求項１６】
サービングピアへの前記クライアントデータパケットリクエストは、前記サービングピアが特別の各クライアントリクエストに応答して送信される前記データパケットを識別し

ないＴＣＰ通信プロトコルを介して提供され、前記クライアントがサービングピアから受信する入力データパケットは、前記データパケットを提供した前記サービングピアに対応する前記リクエストキュー内の第１のデータパケットリクエストに対応することを特徴とする請求項１０記載の方法。

【請求項１７】

前記データパケットは、埋め込みコード化データパケットであり、クライアント制御は、１つ又は複数の特定のデータパケットのクライアントリクエストを１つ又は複数の特定のサービングピアに自動的に制限することによって、各埋め込みコード化データパケットのストリーミングビットレートで維持されることを特徴とする請求項１０記載の方法。

【請求項１８】

前記サービングピアの集合サービング帯域幅、

前記共通ステージングキューのサイズ、

所望のリクエスト達成時間、

前記共通ステージングキュー内の受信パケット長、

各パケットリクエストキューの長さ、および

前記埋め込み符号化データパケットの基層ビットレートに応じて、前記クライアントリクエストを自動的に制限することを特徴とする請求項１７記載の方法。

【請求項１９】

複数の非協働ピアのクラスタからクライアントドリブンのメディアストリーミングを調整するシステムにおいて、

クライアントとの通信に利用可能なサービングピアのクラスタ内の複数のサービングピアにおける符号化されたメディアファイルの全パケットを分散させること、

クライアントリクエストに応答して、前記クラスタ内の前記複数のサービングピアのリストを前記クライアントに提供すること、

前記クライアント上に複数のパケットリクエストキューを提供することであって、各パケットリクエストキューは、前記サービングピアのクラスタ内の前記複数のサービングピアのうちの一つに対応すること、

データパケットリクエストが前記クライアントから前記サービングピアの一つに送信される場合、各データパケットリクエストを前記対応するパケットリクエストキューに追加すること、

前記クライアントが前記サービングピアから前記対応するデータパケットを受信する場合、前記対応するパケットクエストキューから各データパケットリクエストを取り出すこと、

クライアントステージングキューに各受信データパケットをキャッシングすること、および、

前記受信データパケットをデコーディングしレンダリングし前記クライアント上でストリーミングメディア再生を提供すること

を備えることを特徴とするシステム。

【請求項２０】

クライアントリクエストに応答して前記クラスタ内の各サービングピアから可用性ベクトルを提供することを更に備え、前記各可用性ベクトルは対応する各サービングピアに格納される利用可能なデータパケットを表すことを特徴とする請求項１９記載のシステム。

【請求項２１】

１つ又は複数の特定のデータパケットに対する１つ又は複数のクライアントリクエストを、前記クラスタ内の１つ又は複数の特定のサービングピアに送信することは、前記クラスタ内の各サービングピアに対応する前記可用性ベクトルのクライアント分析に基づくことを特徴とする請求項２０記載のシステム。

【請求項２２】

前記クライアントステージングキューの前記長さは、前記受信データパケットをデコーディングしレンダリングする前に、前記リクエストされたデータパケットの所望のバッフ

ァリング量を提供するために変化することを特徴とする請求項１９記載のシステム。
【請求項２３】
前記クライアント上の複数のパケットリクエストキューを提供することは、前記クラスタに加わり及び前記クラスタから離れるサービングピアに応答してクライアントリクエストキューを動的に追加し及び取り出すことを更に備えることを特徴とする請求項１９記載のシステム。
【請求項２４】
前記クラスタから離れるサービングピアに応答してクライアントリクエストキューを取り出すことは、取り出されたクライアントリクエストキュー内に残存するデータパケットリクエストを１つ又は複数の他のクライアントリクエストキューに移すことと、前記対応するサービングピアに前記対応するパケットをリクエストすることを備えることを特徴とする請求項１９記載のシステム。
【請求項２５】
前記クラスタ内の各サービングピアの所望のリクエスト達成時間（ＲＦＴ）を維持するためにクライアントデータパケットリクエストを動的にバランシングすることにより、前記クラスタ内の前記各サービングピア間の帯域幅負荷バランシングを実行することを更に備えることを特徴とする請求項１９記載のシステム。
【請求項２６】
前記クライアントデータパケットリクエストは、ＴＣＰ通信プロトコルを介して前記様々なサービングピアに送信され、前記クライアントが特定のサービングピアから受信する入力データパケットは、前記対応するクライアントリクエストキュー内の第一のデータパケットに対応することを特徴とする請求項１９記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、緩やかに結合されたピアツーピア（Ｐ２Ｐ）ネットワークでの受信側主導(receiver-driven)のＰ２Ｐメディアストリーミングに関し、詳細には、複数のピアから１つのクライアントに、クライアントのリアルタイムの調整および制御の下で、ピアツーピアコラボレーションを提供せずにメディアをストリーミングするシステムおよび方法に関する。
【背景技術】
【０００２】
最近の市場調査では、２００４年に、米国の半数を超えるインターネットユーザが何らかの形のストリーミングメディアにアクセスしたことが示されている。ストリーミング音楽へのアクセスは非常に人気の高いアクティビティであるが、ストリーミングビデオの人気も急速に高まっている。
【０００３】
残念ながら、典型的なＷｅｂページと異なり、ストリーミングメディアファイルは、通常、極めてサイズが大きい。例えば、毎秒２メガビット（Ｍｂｐｓ）で符号化された３分間の映画予告編は、使用されるコーデックによっては４５メガバイト（ＭＢ）のメディアファイルを生じ得る。ストリーミングメディアによって対処されなければならない別の問題が、パケット配信のクリティカルタイミングである。その結果として、ストリーミングメディアファイルのサイズの大きさおよびパケット配信タイミング要件により、典型的なストリーミングメディアサーバは、セットアップし、運営するのに比較的高くつくことになる。例えば、ある最新の見積もりは、ストリーミングメディアの実勢価格を、サービング(serving)トラフィック１ＧＢあたり１０ドルと設定している。４５ＭＢのファイルサイズの例を使用すると、これは、配信される映画予告編１本当たり０．４５ドルの帯域幅コストを生じ得る。明らかに、そのようなコストは、メディアストリーミングの量が増大するにつれて急速に拡大し得る。
【０００４】

メディアストリーミングの比較的高いコストへの１つの解決法は、「ピアツーピア」（Ｐ２Ｐ）ネットワークを使って個々のクライアントにメディアストリーミングを提供することである。一般に、Ｐ２Ｐネットワークの基本的考え方とは、ストリーミングメディアを配信する際に各ピアノードにメディアサーバを支援させるというものである。メディアストリーミングでのＰ２Ｐネットワークの成功は、Ｐ２Ｐネットワークを実施する多数の従来の手法を生み出している。

【０００５】

例えば、「エンドシステムマルチキャスト」および「ピアキャスト」(PeerCast)と呼ばれる従来のＰ２Ｐ方式は、メディアストリーミングにアプリケーションレベルマルチキャスト（ＡＬＭ）を使用する。具体的には、ＥＳＭでもピアキャストでも、各ピアノードが、既存のＩＰネットワークを介したオーバーレイツリー(overlay tree)に自己編成される。次いで、ストリーミングデータがそのオーバーレイツリーに沿って配信される。その場合、帯域幅を提供するコストは、それらのピアノードの間で分担され、それによって、メディアサーバを運営する帯域幅負担（ゆえに金銭的コスト）が低減される。しかしながら、ＥＳＭでもピアキャストでも、その配信ツリーの各葉ノードは、ストリーミングメディアを受け取るだけで、コンテンツ配信には貢献しない。

【０００６】

他の２つの従来方式、すなわち「ＣｏｏｐＮｅｔ」および「ＳｐｌｉｔＳｔｒｅａｍ」は、ＥＳＭやピアキャストなどの方式のコンテンツ配信制限に、配信元とピアノードにまたがる複数の配信ツリーを使って対処する。その場合、ＣｏｏｐＮｅｔおよびＳｐｌｉｔＳｔｒｅａｍにおける各ツリーは、別個のストリーミングメディアを送信することができる。その結果、すべてのピアノードがコンテンツ配信に関与することができる。

【０００７】

従来のＰ２Ｐメディアストリーミング解決法の別の例には、「ＯＳｔｒｅａｍ」と呼ばれるストリーミング方式が含まれる。Ｏｓｔｒｅａｍは、「キャッシュアンドリレー」の手法を使って、各ピアノードが、各クライアントに、そのキャッシュから以前に配信されたメディアを供給することができるようにする。別の従来のシステム、「ＧｎｕＳｔｒｅａｍ」は、よく知られている「グヌーテラ（Ｇｎｕｔｅｌｌａ）」システムの上に構築される受信側主導のＰ２Ｐメディアストリーミングシステムを提供する。「ＣｏｌｌｅｃｔＣａｓｔ」と呼ばれるさらに別の従来方式は、ネットワーク変動およびピア障害に動的に適合しつつ、最適なストリーミング品質を達成する可能性の最も高いサービングピア(serving peers)を積極的に探し出す。

【０００８】

別の種類の従来方式は、１つのファイルの各部分がいくつかのピアにわたって広く分散される一種の分散ファイル共用を提供する。その場合、クライアントがそのファイルのダウンロードを要求するたびに、その要求は、サーバから直接にではなく、複数のピアからサービング(serving)される。例えば、「Ｓｗａｒｍｃａｓｔ」と呼ばれるそのような一方式は、人気のあるダウンロード可能なコンテンツを提供するあるＷｅｂサイトに置かれた負荷を、ファイルをずっと小さな部分に分割することによって広く分配する。ユーザがＳｗａｒｍｃａｓｔクライアントプログラムをインストールすると、そのユーザのコンピュータは、他のユーザのコンピュータと自動的に協力して、それらがすでにダウンロードしてあるデータの各部分を順に受け渡し（すなわちサービスを提供し）、それによって中央サーバ上の全体的サービング負荷を低減させる。「ビットトレント（ＢｉｔＴｏｒｒｅｎｔ）」と呼ばれる類似の方式は、よく似た原理に従って機能する。具体的には、低い負荷では、ビットトレント方式を使って大きなファイルを供給するＷｅｂサイトは、典型的なＨＴＴＰサーバそっくりに振る舞う。というのは、そのサイトがそのサービングの大部分をそれ自体で実行するからである。しかしながら、サーバの負荷が何らかの比較的高いレベルに達すると、ビットトレントは、他のダウンロード側クライアントにサービングするためにアップロード負担の大部分がダウンロード側クライアント自体によって担われる状態にシフトする。

【０００９】
残念なことに、Ｓｗａｒｍｃａｓｔやビットトレントといった方式は、Ｐ２Ｐネットワークサイズの関数として劇的に増大するサーバ容量のためにファイルの各部分を分散させるのには非常に有用であるが、これらのシステムは、効率良くメディアをストリーミングするのには適合されていない。具体的には、Ｓｗａｒｍｃａｓｔやビットトレントといった方式は、ダウンロードされる１つまたは複数のファイルを構成するデータパケットの配信の順序またはタイミングには注意を払わない。これらのファイルは、単に、様々なピアから１つのクライアントに分割してブロードキャストされ、次いで、クライアントコンピュータ上で元のファイルを再構築するために、単に、ローカルで正しい順序に組み立て直されるだけである。しかしながら、ストリーミングメディアの場合には、そのメディアの効率良いストリーミングを提供するために、データパケットのタイミングおよび順序が慎重に考慮され、制御されなければならない。

【発明の開示】

【発明が解決しようとする課題】

【００１０】
したがって、求められているのは、緩やかに結合されたピアの集合体からクライアントへのメディアストリーミングの受信側主導の制御のシステムおよび方法である。そのようなシステムは、ピア間の通信も協働も必要とすべきではない。さらに、そのようなシステムおよび方法は、クライアントに任意の必要な計算操作の大部分を実行するよう要求することにより、各ピアに出される計算要求を最小限に抑えるべきである。

【課題を解決するための手段】

【００１１】
本明細書で説明する「ＰｅｅｒＳｔｒｅａｍｅｒ」は、緩やかに結合されたＰ２Ｐネットワークでの受信側主導のピアツーピア（Ｐ２Ｐ）メディアストリーミングを提供する。ネットワーク中の各ピアは、単純な動作だけを実行し、ストリーミングメディアの全部または一部をキャッシュすることができ、他のピアと協働せず、信頼性が低い可能性があり、任意の所与のストリーミングセッション中にオフラインになることも、オンラインになることもある。ネットワーク中のクライアント（または受信側）は、各ピアを調整し、複数のピアからメディアをストリーミングし、負荷均衡化を実行し、各ピアのオンライン／オフライン状態を処理し、ストリーミングメディアの復号化およびレンダリングを実行するようにリアルタイムで動作する。

【００１２】
本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒシステムは、複数のクライアントおよびピアを備える大規模なＰ２Ｐネットワークでの使用に適用可能であるが、以下の説明では、説明を明確にするために、一般的に個々のクライアントに言及する。説明するＰｅｅｒＳｔｒｅａｍｅｒによって提供されるシステムおよび方法が複数のクライアントに適用可能であることを当分野の技術者は理解するであろう。また、本明細書で説明するピアは、受信側またはクライアントにメディアを供給するのに使用されるため、Ｐ２Ｐネットワーク中のピアのクラスタを、本明細書では一般に、ピア、または「サービングピア」と呼ぶ。また、これらの「サービングピア」を、本明細書で説明する、個々のストリーミングメディアファイルがそこから最初に発信される「メディアサーバ」と混同すべきでないことにも留意すべきである。

【００１３】
一般に、ＰｅｅｒＳｔｒｅａｍｅｒは、受信側主導のメディアストリーミングを提供する。ＰｅｅｒＳｔｒｅａｍｅｒ動作は、各受信側クライアントが、要求されたストリーミングメディアの全部または一部を保持する近隣のピアのリストを取り出すことから開始する。このコンテキストでは、メディアサーバは、サービングピアの１つとしても働くことができることに留意されたい。このリストは、サービングメディア(serving media)の完全な、または一部のコピーを保持する１つまたは複数の近隣のサービングピアの集合のＩＰアドレスおよびリスニングポートを含む。このリストを取り出す方法には、１）メディ

アサーバから直接リストを取り出すこと、２）知られているサービングピアからリストを取り出すこと、および３）分散ハッシュ表（ＤＨＴ）の手法を使ってサービングピアを識別することが含まれる。

【００１４】

クライアントは、利用可能なサービングピアのリストを取り出すと、各サービングピアに接続し、その「可用性ベクトル」を獲得する。一般に、各サービングピアごとの可用性ベクトルは、そのサービングピアによって保持されているメディアの正確な部分の簡潔な記述である。次いで、クライアントがこれらの可用性ベクトルを使って、正確に、符号化されたメディアのどのブロックが様々なサービングピアによって保持されているか判定する。

【００１５】

例えば、特定のサービングピアがサービングメディア全体を保持している場合、そのピアの可用性ベクトルは、そのサービングピアが完全なメディアコピーを保持していることを示す単一のフラグとすることができる。同様に、サービングピアがサービングメディアの一部だけを保持している場合、そのサービングピアの可用性ベクトルは、クライアントに、そのサービングピアによってメディアのどの部分が保持されているか、例えば、そのサービングピアによって保持されている各パケットのブロック数およびブロック索引などを知らせる。

【００１６】

さらに、以下で説明する消去符号化技法(erasure coding techniques)など、別の符号化が使用されている場合、可用性ベクトルは、サービングピアに割り当てられたメディア消去符号化キー、およびそのサービングピアによって保持されている消去ブロック(erasure blocks)の数を含むことになる。また、サービングピアが消去符号化を使用し、メディアも埋め込み符号化されている場合、可用性ベクトルは、割り当てられたメディア消去符号化キー、埋め込み符号化によって使用される異なるビット速度レベルでの各パケットの消去ブロックの数を含むことになる。

【００１７】

一般に、符号化メディアファイルは、通常、「メディアヘッダ」と、それに続く符号化メディアを表すいくつかのメディアパケット（すなわち「メディア本体」）を含む。可用性ベクトルが与えられたと仮定すると、次のステップは、クライアントが、そのピアクラスタからストリーミングされる符号化メディアファイルから導出されるメディアヘッダおよび「メディア構造」の長さを取り出すことである。パケットの集合のメディア構造は、単に、パケットヘッダにパケットビットストリーム長を加えたものである。これらの長さが取り出された後、クライアントは、そのメディアヘッダおよびメディア構造の「データユニットＩＤ」を計算し、協働してそのピアクラスタ中の１つまたは複数のピアからそれらを取り出す。

【００１８】

メディアヘッダが到着すると、クライアントはそのメディアヘッダを分析し、次いで、ストリーミングされる特定の種類（すなわち、ＭＰＥＧ１／２／４、ＷＭＡ、ＷＭＶなど）のメディアを復号化し、レンダリングし、または再生するのに必要とされるあらゆるオーディオ／ビデオデコーダおよびレンダリングデバイスを構成し、または初期設定する。この初期セットアップ段階が完了すると、クライアントは、続いて、以下で説明するように、ピアクラスタからのメディア本体の進行中のストリーミングを調整する。

【００１９】

具体的には、特定のストリーミングメディアの前述のメディア構造が与えられたと仮定すると、クライアントは、そのストリーミングメディア（すなわちメディア本体）のパケットのデータユニットＩＤを計算し、次いで、それらのパケットを１つずつ取り出す。関連する一実施形態では、ＰｅｅｒＳｔｒｅａｍｅｒは、埋め込み符号化メディアを使用し、その場合、ストリーミングビット速度は、利用可能なサービング帯域幅およびクライアントキュー状況によって変動する。この場合、メディア本体のメディアパケットの進行中

の取り出しは、利用可能な帯域幅に基づき最小速度ひずみを提供するパケットに対応する。

【００２０】

いずれの場合も、クライアントは、サービングピアリストを定期的に更新し、可能な新規のサービングピアに接続する。検査済みの一実施形態では、クライアントは、各可能なサービングピアごとに定期的な従来方式のＴＣＰ接続関数呼び出しを発することによって、可能な新規のサービングピアの有無をチェックした。クライアントは、新規のサービングピアへの接続を確立した後、まず、前述の可用性ベクトルを取り出す。次いで、その新規のピアは、受信側／クライアントの指示で、クラスタ中のその他のアクティブなピアに加わることができる。次いで、クライアントは、それらのピアを調整し、それらのサービング帯域幅およびコンテンツ可用性に従って各ピアのサービング負荷を均衡化させ、切断され、または時間切れになったピアの未完了の要求をその他のアクティブなピアの１つまたは複数に宛先変更する。次いで、ストリーミング動作は、そのストリーミングメディア全体が受け取られ、あるいはストリーミング動作がユーザによって定義されるまで、このようにして続行される。

【００２１】

一実施形態では、ＰｅｅｒＳｔｒｅａｍｅｒは、高速消去耐性符号化(high rate erasure resilient coding)を使って、複数のサービングピアが競合せずに一部のメディアを保持することを可能にし、それによって、どこでどの特定のブロックが取り出されるかに関わらず、クライアントが単純に固定数の消去符号化ブロックを取り出すようにする。この場合、受け取られた消去符号化ブロックは、クライアントのステージングキューに置かれ、次いでそこで、メディアパケットが組み立てられる。次いで、完全に組み立てられたメディアパケットが下流に送られて、ストリーミングされている特定の種類のメディアを復号化し、レンダリングし、または再生するために構成され、または初期設定されているあらゆるオーディオ／ビデオデコーダおよびレンダリングデバイスを使って復号化され、再生される。この場合、ステージングキューの長さ、要求キューの長さ、および圧縮オーディオ／ビデオバッファの長さを制御することにより、クライアントは、ある程度の所望の期間（検査済みの実施形態では、４秒間程度）のストリーミングバッファを維持する。次いで、この組み合わされたバッファを使って、ネットワーク損失およびジッタに対処する。

【００２２】

前述の概要を考察すれば、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒがＰ２Ｐネットワークにおいて受信側主導のメディアストリーミングを提供する独自のシステムおよび方法を提供することが明らかである。前述の利点以外にも、以下の詳細な説明を、添付の図面と併せて読めば、ＰｅｅｒＳｔｒｅａｍｅｒの他の利点が明らかになるであろう。

【００２３】

本発明の特有の特徴、態様および利点は、以下の説明、添付の特許請求の範囲、および添付の図面を参照すればより良く理解されるであろう。

【発明を実施するための最良の形態】

【００２４】

以下の本発明の好ましい実施形態の説明では、本明細書の一部を形成し、本発明が実施され得る具体的実施形態を例として示す、添付の図面を参照する。他の実施形態を利用することもでき、本発明の範囲を逸脱することなく構造的変更を加えることもできることが理解される。

【００２５】

１．０　例示的動作環境

図１に、本発明が実施され得る適当なコンピューティングシステム環境１００の一例を示す。コンピューティングシステム環境１００は、適当なコンピューティング環境の一例にすぎず、本発明の用途または機能の範囲に関するどんな限定を示唆するものでもない。また、コンピューティング環境１００は、例示的動作環境１００に示す構成要素のいずれ

か１つまたはそれらの組合せに関連するどんな依存関係または要件を持つものであるとも解釈されるべきではない。

【００２６】

本発明は、他の多数の汎用または専用コンピューティングシステム環境または構成と共に動作する。本発明と共に使用するのに適し得る公知のコンピューティングシステム、環境、および／または構成の例には、それだけに限らないが、パーソナルコンピュータ、サーバコンピュータ、携帯電話やＰＤＡといったハンドヘルド、ラップトップまたはモバイルコンピュータまたは通信機器、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラム可能家庭用電化製品、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、前述のシステムまたは機器のいずれかを含む分散コンピューティング環境などが含まれる。

【００２７】

本発明は、マイクロホンアレイ１９８の構成要素を含むハードウェアモジュールと組み合わせて、コンピュータによって実行される、プログラムモジュールなどのコンピュータ実行可能命令の一般的状況で説明され得る。一般に、プログラムモジュールには、個々のタスクを実行し、または個々の抽象データ型を実施するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。本発明は、タスクが、通信ネットワークを介してリンクされたリモート処理装置によって実行される分散コンピューティング環境でも実施され得る。分散コンピューティング環境では、プログラムモジュールは、メモリ記憶装置を含むローカルとリモート両方のコンピュータ記憶媒体に位置することができる。図１を参照すると、本発明を実施する例示的システムは、コンピュータ１１０の形で汎用コンピューティングデバイスを含む。

【００２８】

コンピュータ１１０の構成要素には、それだけに限らないが、処理装置１２０、システムメモリ１３０、およびシステムメモリを含む様々な構成要素を処理装置１２０に結合するシステムバス１２１が含まれ得る。システムバス１２１は、様々なバスアーキテクチャのいずれかを用いた、メモリバスまたはメモリコントローラ、周辺バス、およびローカルバスを含むいくつかの種類のバス構造のいずれでもよい。例をあげると、それだけに限らないが、そのようなアーキテクチャには、産業標準アーキテクチャ（ＩＳＡ）バス、マイクロチャネルアーキテクチャ（ＭＣＡ）バス、拡張ＩＳＡ（ＥＩＳＡ）バス、ビデオ電子装置規格化協会（ＶＥＳＡ）ローカルバス、およびメザニンバスともいう周辺装置相互接続（ＰＣＩ）バスが含まれる。

【００２９】

コンピュータ１１０は、通常、様々なコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピュータ１１０によってアクセスされ得る任意の利用可能な媒体とすることができ、それには揮発性と不揮発性、着脱式と固定式両方の媒体が含まれる。例をあげると、それだけに限らないが、コンピュータ可読媒体には、コンピュータ記憶媒体および通信媒体が含まれ得る。コンピュータ記憶媒体には、コンピュータ可読命令、データ構造、プログラムモジュール、または他のデータといった情報の記憶のための任意の方法または技術で実施された揮発性および不揮発性、着脱式および固定式の媒体が含まれる。

【００３０】

コンピュータ記憶媒体には、それだけに限らないが、ＲＡＭ、ＲＯＭ、ＰＲＯＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリなどのメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）などの光ディスク記憶、磁気カセット、磁気テープ、磁気ディスク記憶などの磁気記憶装置、あるいは所望の情報の記憶に使用することができ、コンピュータ１１０によってアクセスされ得る他の任意の媒体が含まれる。通信媒体は、通常、コンピュータ可読命令、データ構造、プログラムモジュールまたは他のデータを、搬送波や他の搬送機構などの変調データ信号として実施し、任意の情報配信媒体を含む。「変調データ信号」という用語は、その特性の１つまたは複数が、その信号中に情報を符号化するようなやり方で設定され、または変更されている信号を意味する。例をあげると、それだ

けに限らないが、通信媒体には、有線ネットワークや直接配線接続などの有線媒体、および音響、ＲＦ、赤外線、その他の無線媒体などの無線媒体が含まれる。前述のいずれかの組合せも、コンピュータ可読媒体の範囲内に含むべきである。

【００３１】

システムメモリ１３０は、読取り専用メモリ（ＲＯＭ）１３１やランダムアクセスメモリ（ＲＡＭ）１３２などの揮発性および／または不揮発性メモリの形でコンピュータ記憶媒体を含む。基本入出力システム１３３（ＢＩＯＳ）は、始動時などに、コンピュータ１１０内の諸要素間の情報転送を支援する基本ルーチンを含み、通常はＲＯＭ１３１に格納される。ＲＡＭ１３２は、通常、処理装置１２０から即座にアクセス可能であり、かつ／またはそれによって現在操作されているデータおよび／またはプログラムモジュールを含む。例をあげると、それだけに限らないが、図１には、オペレーティングシステム１３４、アプリケーションプログラム１３５、他のプログラムモジュール１３６、およびプログラムデータ１３７が示されている。

【００３２】

コンピュータ１１０は、他の着脱式／固定式、揮発性／不揮発性コンピュータ記憶媒体も含み得る。例にすぎないが、図１に、固定式、不揮発性磁気媒体との間で読取りまたは書込みを行うハードディスクドライブ１４１、着脱式、不揮発性磁気ディスク１５２との間で読取りまたは書込みを行う磁気ディスクドライブ１５１、およびＣＤ－ＲＯＭや他の光媒体などの着脱式、不揮発性光ディスク１５６との間で読取りまたは書込みを行う光ディスクドライブ１５５を示す。例示的動作環境で使用され得る他の着脱式／固定式、揮発性／不揮発性のコンピュータ記憶媒体には、それだけに限らないが、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭなどが含まれる。ハードディスクドライブ１４１は、通常、インターフェース１４０などの固定式メモリインターフェースを介してシステムバス１２１に接続され、磁気ディスクドライブ１５１および光ディスクドライブ１５５は、通常、インターフェース１５０などの着脱式メモリインターフェースによってシステムバス１２１に接続される。

【００３３】

前述の、図１に示す各ドライブおよびそれに関連するコンピュータ記憶媒体は、コンピュータ１１０のためにコンピュータ可読命令、データ構造、プログラムモジュールおよびその他のデータの記憶を提供する。図１では、例えば、ハードディスクドライブ１４１は、オペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７を格納するものとして示されている。これらのコンポーネントは、オペレーティングシステム１３４、アプリケーションプログラム１３５、その他のプログラムモジュール１３６、およびプログラムデータ１３７と同じでも、異なっていてもよいことに留意されたい。オペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７には、少なくともそれらが異なるコピーであることを示すために、図では異なる番号を付してある。ユーザは、キーボード１６２や、一般にマウス、トラックボール、タッチパッドと呼ばれるポインティングデバイス１６１などの入力装置を介してコンピュータ１１０にコマンドおよび情報を入力することができる。

【００３４】

他の入力装置（図示せず）には、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、無線受信機、テレビまたは放送ビデオ受信機などが含まれ得る。上記その他の入力装置は、しばしば、システムバス１２１に結合された有線または無線のユーザ入力インターフェース１６０を介して処理装置１２０に接続されるが、例えば、パラレルポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）、ＩＥＥＥ１３９４インターフェース、ブルートゥース（商標）無線インターフェース、ＩＥＥＥ８０２．１１無線インターフェースといった他の従来方式のインターフェースおよびバス構造によっても接続され得る。さらに、コンピュータ１１０は、やはり、例えば、パラレル、シリアル、Ｕ

ＳＢ、ＩＥＥＥ１３９４、ブルートゥース（商標）といった従来方式の有線または無線インターフェースを含む、マイクロホンやマイクロホンアレイ１９８などの音声またはオーディオ入力装置、ならびに、オーディオインターフェース１９９を介して接続されたラウドスピーカ１９７またはその他の音声出力装置も含み得る。

【００３５】

モニタ１９１または他の種類の表示装置も、ビデオインターフェース１９０などのインターフェースを介してシステムバス１２１に接続される。モニタ１９１以外に、コンピュータは、プリンタ１９６など他の周辺出力装置を含むこともでき、それらは、出力周辺インターフェース１９５を介して接続され得る。

【００３６】

コンピュータ１１０は、リモートコンピュータ１８０など、１つまたは複数のリモートコンピュータへの論理接続を使ってネットワーク化環境で動作し得る。リモートコンピュータ１８０は、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、または他の一般的ネットワークノードとすることができ、図１にはメモリ記憶装置１８１だけしか示していないが、通常は、コンピュータ１１０に関連して前述した要素の多くまたはすべてを含む。図１に示す論理接続には、ローカルエリアネットワーク（ＬＡＮ）１７１および広域ネットワーク（ＷＡＮ）１７３が含まれるが、他のネットワークも含まれ得る。そのようなネットワーク環境は、オフィス、企業規模の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よく見られるものである。

【００３７】

ＬＡＮネットワーク環境で使用されるとき、コンピュータ１１０はネットワークインターフェースまたはアダプタ１７０を介してＬＡＮ１７１に接続される。ＷＡＮネットワーク環境で使用されるとき、コンピュータ１１０は、通常、モデム１７２またはインターネットなどのＷＡＮ１７３を介して通信を確立する他の手段を含む。モデム１７２は、内蔵でも外付けでもよく、ユーザ入力インターフェース１６０または他の適当な機構を介してシステムバス１２１に接続され得る。ネットワークで接続された環境では、コンピュータ１１０に関連して示すプログラムモジュール、またはその一部は、リモートのメモリ記憶装置にも格納し得る。例として、それだけに限らないが、図１に、メモリ装置１８１上にあるものとしてリモートアプリケーションプログラム１８５が示されている。図示のネットワーク接続は例示的なものであり、コンピュータ間で通信を確立する他の手段も使用され得ることが理解されるであろう。

【００３８】

以上で例示的動作環境について論じたが、この説明の残りの部分は、分散メディアストリーミングのための受信側主導のピアツーピア（Ｐ２Ｐ）ネットワークにおいて１つまたは複数のピアのクラスタに対する動的リアルタイムクライアント制御を実現する「ＰｅｅｒＳｔｒｅａｍｅｒ」を実施するプログラムモジュールおよびプロセスの説明に充てる。

【００３９】

２．０　序論

本明細書で説明する「ＰｅｅｒＳｔｒｅａｍｅｒ」は、緩やかに結合されたＰ２Ｐネットワークでの受信側主導のピアツーピア（Ｐ２Ｐ）メディアストリーミングを提供する。ネットワーク中の各ピアは、単純な動作だけを実行し、ストリーミングメディアの全部または一部をキャッシュすることができ、他のピアと協働せず、信頼性が低い可能性があり、任意の所与のストリーミングセッション中にオフラインになることも、オンラインになることもある。ネットワーク中の各クライアントは、ピアを調整し、複数のピアからメディアをストリーミングし、負荷均衡化を実行し、各ピアのオンライン／オフライン状態を処理し、ストリーミングメディアの復号化およびレンダリングを実行するようにリアルタイムで動作する。

【００４０】

本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒシステムは、複数のクライアントおよびピアを備える大規模Ｐ２Ｐネットワークでの使用に適用可能であるが、以下の説明では、

説明を明瞭にするために、一般的に個々のクライアントに言及することに留意されたい。ＰｅｅｒＳｔｒｅａｍｅｒによって提供される説明するシステムおよび方法が複数のクライアントに適用可能であることを当分野の技術者は理解するであろう。また、本明細書で説明するピアはメディアを受信側またはクライアントに供給するのに使用されるため、本明細書では、Ｐ２Ｐネットワーク中のピアのクラスタを、一般に、ピア、または「サービングピア」と呼ぶ。また、「サービングピア」を、本明細書で説明する、個々のストリーミングメディアファイルがそこから最初に発信される「メディアサーバ」と混同すべきでないことにも留意すべきである。

【００４１】

一般に、ＰｅｅｒＳｔｒｅａｍｅｒは、図２で示すようなネットワークなどのＰ２Ｐネットワークで動作する。個々のストリーミングセッションでは、「サーバ」２００は、最初にストリーミングメディアを発信するＰ２Ｐネットワーク中のノードであると定義される。「クライアント」（または受信側）２１０は、現在ストリーミングメディアを要求しているノードであると定義される。また、「サービングピア」２２０は、クライアントに、ストリーミングメディアの完全な、または一部のコピーを供給するノードであると定義される。

【００４２】

一般に、サーバ２００、クライアント２１０およびサービングピア２２０は、すべて、インターネットなどのネットワークに接続されたエンドユーザノードである。サーバ２００は常にストリーミングメディアを供給することができるため、サーバノードは、サービングピア２２０としても働く。また、サーバノード２００は、例えば、利用可能なサービングピアのリストの維持、デジタル権限管理（ＤＲＭ）機能の実行などといった、サービングピア２２０によって実行され得ないメディア管理機能も実行することができる。また、従来のＰ２Ｐ方式と同様に、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒは、ますます多くのストリーミングピアノード２２０が配置されるにつれて増大される効率から恩恵を受けるものである。具体的には、ストリーミングピアノード２２０の数が増大するにつれて、メディアサーバ２００上の負荷が減少し、それによって運営コストがより下がると同時に、各クライアントノード２１０は、個々のメディアストリーミングセッション時により良いメディア品質を受け取ることができるようになる。

【００４３】

また、他の多くのＰ２Ｐ型ネットワークと同様に、個々のノードの役割が変化し得ることも明らかなはずである。例えば、個々のノードは、ある特定のストリーミングセッションではクライアント２１０として働くことができ、別のセッションではサービングピア２２０として働くこともできる。さらに、個々のノードは、同時に、クライアントノード２１０とサーバ２００またはサービングピア２２０の両方として働き、１つまたは複数の他のサービングピアから他のストリーミングメディアを受け取っている間に、同時に、１つまたは複数のメディアファイル、またはメディアファイルの一部をストリーミングすることができる。

【００４４】

ストリーミングセッションの間、クライアント２１０は、まず、所望のメディアの一部または全部を保持するいくつかの近隣のピア２２０を探し出し、次いで、（サーバ２００を含み得る）複数のピアからそのメディアをストリーミングする。その結果として、各サービングピア２２０は、クライアント２１０のダウンロード要求の一部をサービングして全体のアップロード負担を低減することによってサーバ２００を支援するように働く。その結果、クライアント２１０は、特に多くのクライアントがある場合には、しばしば、ずっと優れたストリーミングメディア品質を受け取ることができる。というのは、サーバ２００を支援する多くのストリーミングピア２２０があるときには、大幅に高いサービング帯域幅が利用可能になるからである。

【００４５】

どんなＰ２Ｐネットワークとも同様に、個々のピア２２０は、１つまたは複数のクライ

アント２１０にサービングすることから直接利益を享受しない。しかしながら、一実施形態では、従来のＰ２Ｐ「公平性機構」(fairness mechanism)を使って、協働し合うピア２２０同士は、その後のストリーミング要求を満たされる際に、サービングピアとして働く際に等しく協働していない別のピアよりも高い優先順位を受け取ることが保証される。その結果として、ＰｅｅｒＳｔｒｅａｍｅｒと共にそのような公正性機構を実施すると、協働し合うピア２２０は、通常、次にそれがクライアント２１０になるときに、より良いメディア品質を期待することができる。

【００４６】

その結果として、任意の特定のストリーミングセッション中に各サービングピア２２０が、実際上、クライアント２１０とサーバ２００の両方に有利に働くことを考慮すると、優れた設計理念とは、サービングピアが軽量であり、Ｐ２Ｐネットワークが緩やかに結合されるようにすることである。言い換えると、サービングピア２２０は、低いＣＰＵ負荷を伴う非常に単純な動作だけを実行すればよいはずである。さらに、一実施形態では、サービングピア２２０は、本質的に各サービングピアによって寄付される記憶空間を最小限に抑えるために、メディアの一部だけをキャッシュするように選択することもできる。また、ピア２２０間通信の任意の帯域幅コストを低減するために、各サービングピアは他のピアと協働する必要がないようにすべきである。最後に、任意の特定のサービングピア２２０上で走っている他のプログラムは、任意の特定の時点においてＣＰＵおよびネットワークリソースを要求する際により高い優先順位を持つこともあり、あるいは特定のピアが、単に、任意の時点において電源をオンまたはオフにされることもある。その結果、個々のサービングピア２２０は、利用可能なサービング帯域幅の変動を伴う、信頼性の低いものである可能性がある。実際、個々のサービングピアは、ストリーミングセッション中の任意の時点において、単に、オフラインになることも、オンラインになることもある。

【００４７】

逆に、ストリーミングセッションにリソースを充てるためにクライアント２１０側の負担を増大させることは妥当である。具体的には、クライアント２１０は複数のピア２２０からストリーミングメディアを受け取る必要があり、そのため、これはそれらのピアにすでに接続されている。さらに、クライアント２１０には、これ自体のストリーミング体験を向上させるためにそれらのピア２２０と効果的に協働し、またはそれらを管理しようとする動機がある。

【００４８】

その結果として、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒシステムおよび方法は、緩やかに結合されたＰ２Ｐネットワークにおいてサービングピアに対する受信側主導の制御を利用し、その場合、クライアントは、様々なストリーミングピアの間でパケット要求を送信し、調整する役割を果たす。

【００４９】

２．１　システム概要

前述のように、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒは、緩やかに結合されたＰ２Ｐネットワークでの受信側主導のピアツーピア（Ｐ２Ｐ）メディアストリーミングのシステムおよび方法を提供する。ネットワーク中の各ピアは単純な動作だけを実行し、ストリーミングメディアの全部または一部をキャッシュすることができ、他のピアと協働せず、信頼性が低い可能性があり、任意の所与のストリーミングセッション中にオフラインになりこともオンラインになることもある。ネットワーク中のクライアント（または受信側）は、各ピアを調整し、複数のピアからメディアをストリーミングし、負荷均衡化を実行し、各ピアのオンライン／オフライン状態を処理し、ストリーミングメディアの復号化およびレンダリングを実行するようにリアルタイムで動作する。

【００５０】

一般に、ＰｅｅｒＳｔｒｅａｍｅｒは、受信側主導のメディアストリーミングを提供する。ＰｅｅｒＳｔｒｅａｍｅｒ動作は、まず、各受信側クライアントが、要求されたストリーミングメディアの全部または一部を保持する近隣のサービングピアのリストを取り出

すことから開始する。この状況では、メディアサーバは、サービングピアの１つとしても働くことができることに留意されたい。このリストは、サービングメディアの完全な、または一部のコピーを保持する１つまたは複数の近隣のサービングピアの集合のＩＰアドレスおよびリスニングポートを含む。このリストを取り出す方法には、１）メディアサーバから直接リストを取り出すこと、２）知られているサービングピアからリストを取り出すこと、および３）分散ハッシュ表（ＤＨＴ）の手法を使ってサービングピアを識別することが含まれる。

【００５１】

クライアントが利用可能なサービングピアのリストを取り出すと、クライアントは、各サービングピアに接続し、その「可用性ベクトル」を獲得する。一般に、各サービングピアごとの可用性ベクトルは、各サービングピアによって保持されているメディアの正確な部分の簡潔な記述である。次いで、クライアントがこの可用性ベクトルを使って、正確に、符号化されたメディアのどのブロックがサービングピアによって保持されているか判定する。

【００５２】

例えば、特定のサービングピアがそのサービングメディア全体を保持している場合、そのピアの可用性ベクトルは、そのサービングピアが完全なメディアコピーを保持していることを示す単一のフラグとすることができる。同様に、サービングピアが、そのサービングメディアの一部だけを保持している場合、そのサービングピアの可用性ベクトルは、クライアントに、そのサービングピアによってメディアのどの部分が保持されているか、例えば、そのサービングピアによって保持されている各パケットのブロック数およびブロック索引などを知らせる。

【００５３】

さらに、以下で説明する消去符号化技法など別の符号化が使用されている場合、可用性ベクトルは、サービングピアに割り当てられたメディア消去符号化キー、およびそのサービングピアによって保持されている消去ブロック数を含むことになる。また、サービングピアが消去符号化を使用し、メディアも埋め込み符号化されている場合、可用性ベクトルは、割り当てられたメディア消去符号化キー、その埋め込み符号化によって使用される異なるビット速度レベルでの各パケットの消去ブロック数を含むことになる。

【００５４】

可用性ベクトルが与えられたと仮定すると、次のステップは、クライアントが、ピアクラスタからストリーミングされるメディアの「メディアヘッダ」および「メディア構造」の長さを取り出すことである。これらの長さが取り出された後、クライアントは、メディアヘッダおよびメディア構造の「データユニットＩＤ」を計算し、各サービングピアごとの可用性ベクトルを分析した結果として得られたどのピアがどのパケットＩＤを持っているかの知識に基づき、ピアクラスタ中のピアの１つまたは複数からそれらを取り出す。

【００５５】

メディアヘッダが到着すると、クライアントはそのメディアヘッダを分析し、次いで、ストリーミングされる特定の種類のメディア（すなわちＭＰＥＧ１／２／４、ＷＭＡ、ＷＭＶなど）を復号化し、レンダリングし、または再生するのに必要とされるあらゆるオーディオ／ビデオデコーダおよびレンダリングデバイスを構成し、または初期設定する。この初期セットアップ段階が完了すると、クライアントは、続いて、以下で説明するように、ピアクラスタからのメディア本体の進行中のストリーミングを調整する。具体的には、特定のストリーミングメディアの前述のメディア構造が与えられたと仮定すると、クライアントは、そのストリーミングメディアのパケットのデータユニットＩＤを計算し、次いで、様々なピアからそれらのパケットを１つずつ取り出す。

【００５６】

次いで、クライアントは、（サービングピアを識別する前述の方法の１つを使って）サービングピアリストを定期的に更新し、可能な新規のサービングピアに接続する。検査済みの一実施形態では、クライアントは、各可能なサービングピアごとに定期的なＴＣＰ接

続関数呼び出しを発することによって可能な新規のサービングピアの有無をチェックした。クライアントは、新規のサービングピアへの接続を確立した後、まず、前述の可用性ベクトルを取り出す。次いで、新規のピアは、受信側／クライアントの指示で、クラスタ中のその他のアクティブピアに加わることができる。次いで、クライアントは、各ピアを調整し、それらのサービング帯域幅およびコンテンツ可用性に従って各ピアのサービング負荷を均衡化させ、切断され、または時間切れになったピアの未完了の要求をその他のアクティブピアの１つまたは複数に宛先変更する。次いで、ストリーミング動作は、そのストリーミングメディア全体が受け取られ、あるいはストリーミング動作がユーザによって停止されるまで、このようにして続行される。

【００５７】

２．２　システムアーキテクチャ概要

前述したプロセスを図３の全体システム図で示す。具体的には、図３のシステム図には、本明細書で説明する、ＰｅｅｒＳｔｒｅａｍｅｒを実施するプログラムモジュール間の相互関係が示されている。図３に破線または点線で表す任意のボックスおよびボックス間の相互接続は、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒの代替実施形態を表すこと、およびこれらの代替実施形態のいずれかまたはすべては、以下で説明するように、本明細書全体にわたって説明する他の代替実施形態と組み合わせて使用され得ることに留意すべきである。

【００５８】

一般に、ＰｅｅｒＳｔｒｅａｍｅｒは、各クライアント２１０に関する動作を、そのクライアントに、ピア位置検出(peer locating)モジュール３０５を使って要求されたストリーミングメディアの全部または一部を保持する近隣のサービングピア２２０のリスト３１０を取り出させ、または識別させることから動作を開始する。この状況では、メディアサーバ２００はサービングピア２２０の１つとしても働くことができることに留意されたい。様々な方法が、ピア位置検出モジュール３０５によってピアリスト３１０を取り出すために使用される。例えば、一実施形態では、ピアリスト３１０はサーバ２００から直接提供される。別の実施形態では、ピアリスト３１０は、知られているサービングピア２２０から取り出される。最後に、別の実施形態では、ピア位置検出モジュール３０５により、従来方式の分散ハッシュ表（ＤＨＴ）を使ってサービングピア２２０が識別される。前述のように、ピアリスト３１０は、サービングメディアの完全な、または一部のコピーを保持する１つまたは複数の近隣のサービングピア２２０のＩＰアドレスおよびリスニングポートを含む。

【００５９】

サービングメディア自体は、サーバ２００上に存在するメディア符号化モジュール３００によって、例えば、ＭＰＥＧ１／２／４、ＷＭＡ、ＷＭＶなどを含むいくつかの従来方式のコーデックのいずれかを使って符号化される。メディアを符号化するのに使用されるコーデックは、本明細書でさらに詳細に説明するように、埋め込み式とすることも非埋め込み式とすることもできることに留意されたい。さらに、一実施形態では、以下でさらに詳細に説明する「高速消去耐性符号化」をコーデックのいずれかと組み合わせて使って、本来的に信頼性の低いサービングピア２２０にさらなる頑強性が提供される。

【００６０】

最初に、符号化されたメディアは、そのメディアがそこで最初に符号化されたサーバ上だけに存在する。次いで、その全部または一部が、サービングピア２２０の１つまたは複数に分散される（やはり、サーバ２００は、メディアストリーミングのためのサービングピアとして働くこともできる）。サービングピア２２０への分散は、各ピアへのメディアストリームのパケットの直接分散の結果であり、または（クライアント２１０として働くときに）そのメディアをすでにストリーミングしているピアの１つまたは複数に、単に、それが最初にそのサービングピアにストリーミングされるときにそのメディアの全部または一部をキャッシュさせた結果である。いずれの場合も、説明のために、（ピアリスト３１０によって定義される）いくつかの知られているピアがあり、各ピアは、ストリーミン

グされる符号化メディアの全部または一部を保持しているものと想定される。

【００６１】

クライアント２１０は、利用可能なサービングピアのリスト３１０を取り出すと、各ピアから前述の可用性ベクトルを取り出す可用性ベクトル取り出しモジュール３２０を介して各サービングピア２２０に接続する。次に、各ピア３２０ごとに可用性ベクトルの情報が与えられたと仮定すると、次いで、クライアント２１０は、メディアヘッダ／メディア構造分析モジュール３２５を使って、ピアクラスタ２２０からストリーミングされるメディアヘッダおよびメディア構造の「メディアヘッダ」および「メディア構造」の長さを取り出す。これらの長さが取り出された後、クライアント２１０は、メディアヘッダを分析し、次いで、クライアント構成モジュール３３０を使って、ストリーミングされる特定の種類のメディア（すなわち、ＭＰＥＧ１／２／４、ＷＭＡ、ＷＭＶなど）を復号化し、レンダリングし、または再生するのに必要とされるあらゆるオーディオ／ビデオデコーダおよびレンダリングデバイスを構成し、または初期設定する。

【００６２】

また、メディアヘッダ／メディア構造分析モジュール３２５は、メディア構造およびメディアヘッダの分析から、埋め込み符号化メディアまたは高速消去耐性符号化、あるいはその両方がストリーミングされるメディアの符号化に使用されているかどうかについての判定も行う。

【００６３】

次いで、データユニットＩＤ計算モジュール３３５を使って、メディアヘッダおよびメディア構造中に含まれる情報に基づき、ストリーミングメディアのパケットの「データユニットＩＤ」が計算される。次いで、データユニット要求モジュール３４０は、算出されたデータユニットＩＤを使って、ピアクラスタ２２０中の様々なピアに、ストリーミングメディアの特定のパケットまたはデータブロックを要求する。

【００６４】

ＰｅｅｒＳｔｒｅａｍｅｒが埋め込み符号化メディアを使用する場合、ストリーミングビット速度は、以下でより詳細に説明するように、利用可能なサービング帯域幅およびクライアントキュー状況に従って変動する。この場合、データユニット要求モジュール３４０によるメディアパケットまたはデータユニットの取り出しを求める進行中の要求は、利用可能な帯域幅に基づいて最小速度ひずみを提供するパケット（またはデータブロック）に対応する。さらに、高速消去耐性符号化が使用される別の場合には、複数のサービングピアが競合せずに一部のメディアを保持して、どこでどの特定のブロックが取り出されるかにかかわらずクライアントが、単純に、固定数の消去符号化ブロックを取り出すようにする。

【００６５】

いずれの場合も、クライアント２１０は、データユニット処理モジュール３４５を介してメディアのストリーミングブロックを取り出す際に、以下で説明するように、それらのパケットを復号化するために渡し、またはデータユニット処理モジュールが、まず、データブロックからメディアストリームのパケットを再構築する（以下の高速消去符号化の説明を参照されたい）。また、クライアント２１０は、（前述のサービングピアを識別する方法の１つを使って）サービングピアリスト３１０を定期的に更新する。リスト３１０が更新されえるたびに、または所望の頻度で、クライアント２１０は、前述の可用性ベクトルを取り出すために可能な新規のサービングピアに接続する。その場合、新規はピアは、受信側／クライアント２１０の指示で、クラスタ２２０中のその他のアクティブピアに加わることができる。

【００６６】

次いで、クライアント２１０は、それらのピア３２０を調整し、それらのサービング帯域幅およびコンテンツ可用性に従って各ピアのサービング負荷を均衡化させ、切断され、または時間切れになった未完了の要求をその他のアクティブピアの１つまたは複数に宛先変更する。次いで、ストリーミング動作は、そのストリーミングメディア全体が受け取ら

れ、復号化／レンダリング／再生モジュール３５０によって復号化され、レンダリングされ、再生されるまで、このようにして続行される。復号化メディアの再生は、その入力を復号化／レンダリング／再生モジュール３５０から提供される従来の表示装置３５５および／またはスピーカ３６０によって実現されることに留意されたい。

【００６７】

３．０　動作概要

前述の各プログラムモジュールが、ＰｅｅｒＳｔｒｅａｍｅｒを実施するのに用いられる。前述したように、ＰｅｅｒＳｔｒｅａｍｅｒは、緩やかに結合されたピアツーピア（Ｐ２Ｐ）ネットワークでの受信側主導のＰ２Ｐメディアストリーミングを提供する。以下の各項では、ＰｅｅｒＳｔｒｅａｍｅｒの動作、および第２項で図２に関連して説明した各プログラムモジュールを実施する例示的方法の詳細な説明を行う。具体的には、以下の第３．１項および３．２項でＰｅｅｒＳｔｒｅａｍｅｒ動作の詳細な説明を行い、続いて、図１０に、その詳細な説明を考慮してＰｅｅｒＳｔｒｅａｍｅｒの全体動作を要約する動作流れ図を提示する。

【００６８】

３．１　ＰｅｅｒＳｔｒｅａｍｅｒの動作詳細

以下の各段落では、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒの特有の動作実施形態および代替実施形態について詳述する。具体的には、以下の各段落では、ＰｅｅｒＳｔｒｅａｍｅｒによって使用される「ストリーミングメディアモデル」、要求されたストリーミングメディアの「メディア構造」（基本的には、メディアパケットまたは「データユニット」を取り出すためのデータＩＤを算出するのに必要とされるストリーミングメディアの特性を定義する「コンパニオンファイル」）、ストリーミングのためのメディアパケットの固定サイズ部分を表すＰｅｅｒＳｔｒｅａｍｅｒデータユニット、記憶所要量を減らすためのメディアの部分キャッシング、本来的に信頼性の低いサービングピアに対してＰｅｅｒＳｔｒｅａｍｅｒシステムの頑強性を増大させるためのメディアの高速消去符号化について述べる。

【００６９】

３．１．１　ストリーミングメディアモデル

一般に、ストリーミングメディアは、それらが到着する際に復号化され、レンダリングされるパケットの流れからなる（それでストリーミングという名前で呼ばれる）。ストリーミングを使用しない場合、そのメディア全体を１つの大きな塊としてダウンロードしてからでないと、それを使用することができない。ＰｅｅｒＳｔｒｅａｍｅｒによって使用されるストリーミングメディアファイルの一般的構造を図４に示す。

【００７０】

具体的には、図４に示すように、メディアは「メディアヘッダ」に先導され、これは、そのメディアを記述する全体的な情報、例えば、メディア中のチャネル数、各チャネルの属性および特性（オーディオサンプリング速度、ビデオ解像度／フレーム速度）、使用されるコーデック、メディアの著者／著作権保持者などを含む。メディアヘッダは、普通、クライアントがその後に受け取るパケットを復号化し、レンダリングするための必要なツールをセットアップすることができるように、ストリーミングセッションの開始前にダウンロードされる。ストリーミングメディアは、そのそれぞれが独立に選択され、符号化され得る別個のメディアコンポーネントである、例えば、英語のオーディオトラック、スペイン語のオーディオトラック、４：３ビデオ、１６：９ビデオなどといったいくつかのチャネルから構成され得ることに留意されたい。

【００７１】

メディアヘッダの後にはメディアパケットのシーケンスが続き、そのそれぞれは、短期間に及ぶ特定のチャネルの圧縮されたビットストリームを含む。各メディアパケットはパケットヘッダに先導され、これは、チャネル索引、パケットの開始タイムスタンプ、パケットのデュレーション(duration)、ならびに、例えば、そのパケットがキーフレーム（ＭＰＥＧ１フレームなど）であるかどうか、そのパケットが（切り捨て可能なビットストリ

ームを伴う）埋め込み符号化パケットであるかどうかなどを示すいくつかのフラグといった情報を含む。次いで、パケットの圧縮されたビットストリームが続く。

【００７２】

目下、ＭＰＥＧ１／２／４オーディオ／ビデオ、ＷＭＡ／ＷＭＶ、ＲｅａｌＡｕｄｉｏ（登録商標）／ＲｅａｌＶｉｄｅｏ（登録商標）などといった、従来方式の圧縮メディアコーデックの大部分は、非埋め込み符号化メディアパケットを生成する。その結果として、そのようなシステムによって生成されるメディアパケットのサイズは変更され得ない。さらに、そのようなビットストリーム中のメディアパケットの１つが失われ、または過度に遅れるたびに、結果として、そのメディアは復号化不能になり、または再生が途切れ途切れまたは断続的になり、ストリーミングメディアの再生品質が劣化する。これらの従来のコーデックとの互換性を保持するために、一実施形態では、ＰｅｅｒＳｔｒｅａｍｅｒシステムおよび方法は、メディアパケットを非埋め込み符号化（拡張不能）とすることができる。しかしながら、従来の圧縮されたメディア形式をサポートすることに加えて、ＰｅｅｒＳｔｒｅａｍｅｒは、一実施形態では、埋め込み符号化メディアもサポートする。

【００７３】

埋め込み符号化メディアを用いた場合、各メディアパケットは、それが後で独立に切り捨てられ得るようなやり方で符号化される。一般には、ＰｅｅｒＳｔｒｅａｍｅｒによって２種類の埋め込み符号化、すなわちビットプレーン符号化および拡張階層符号化(enhancement layer coding)がサポートされる。これら２種類の埋め込み符号化は、当分野の技術者にはよく知られていることに留意されたい。したがって、以下の各段落ではそのような符号化について一般的にのみ説明する。

【００７４】

例えば、ビットプレーン符号化を用いる場合、メディアブロックの拡張可能な符号化が、一般に、１ブロックのオーディオ／ビデオ変換係数を１ビットプレーンずつ、最上位ビットプレーン（ＭＳＢ）から最下位ビットプレーン（ＬＳＢ）まで符号化することによって達成される。ビットストリームが符号化後に切り捨てられる場合、その情報は、それらの係数すべてのうちの最上位ビットプレーンのいくつかについて保持される。さらに、切り捨てられるビットストリームはより低いビット速度の圧縮ビットストリームに対応し、これはより高いビット速度の圧縮ビットストリームに埋め込まれているとみなすことができ、それで埋め込み符号化という名前で呼ばれる。その結果、埋め込みエンコーダによって生成されたメディアパケットは、適切な速度／ひずみトレードオフで切り捨てられ得る。

【００７５】

拡張階層符号化を用いる場合、メディアコンテンツは、基本階層と１つまたは複数の拡張階層に圧縮され、そのそれぞれは、通常、別個のチャネルを占める。具体的には、そのような符号化は、基本階層にサブスクライブすることによって、最低品質のメディアストリームが受け取られるようにする。各連続する拡張階層を付加すると、復号化メディアの品質が向上する。したがって、そのようなシステムを用いる場合、受信側またはクライアントは、通常、基本階層、および利用可能な帯域幅に応じて、できるだけ多くの拡張階層にサブスクライブすることによって、受け取られる情報の品質を最適化する。

【００７６】

３．１．２　ＰｅｅｒＳｔｒｅａｍｅｒメディア構造

受信側主導のモードで動作するには、ＰｅｅｒＳｔｒｅａｍｅｒクライアントは、各ピアにどのパケットおよび各パケットのどの部分を要求するか知るために、要求されるメディアパケットの構造を知る必要がある。この情報は、要求されるストリーミングメディアの構造の定義を含む一種の「コンパニオンファイル」として提供される。一般に、このメディア構造は、ＰｅｅｒＳｔｒｅａｍｅｒクライアントにメディア全体の鳥瞰図（例えば、各パケットの開始タイムスタンプ、各パケットのデュレーションなど）を提供して、クライアントがＰ２Ｐストリーミングセッションを知的に計画し、個々のメディアパケットが復号化およびレンダリングに遅れずに到着することを確認できるようにする。メディア

構造情報を含むコンパニオンファイルは、最初、そのメディアファイルが最初に符号化されるときに生成され、次いで、要求に応じて、各ストリーミングセッションの開始時に、そのメディアヘッダを求める最初の要求と共にクライアントにストリーミングされることに留意されたい。コンパニオンファイル中の情報は、メディアが従来のコーデックによって符号化された後で、メディアヘッダおよびパケットヘッダを分析することによっても生成され得ることに留意されたい。

【００７７】

具体的には、パケットの集合のメディア構造は、パケットヘッダにパケットビットストリームの長さを加えたものからなる。したがって、クライアントは、この情報を使って、どの特定のパケットを要求すべきか、それらのパケットを要求すべき時刻、およびそこにそれらのパケットを要求すべきピアを決定することができる。したがって、ＰｅｅｒＳｔｒｅａｍｅｒは、まず、ストリーミング「セットアップ」段階においてメディア全体のメディア構造を取り出す。実際にメディアをストリーミングするのに先立ってこの情報の取り出すと、ストリーミングの始動に際してわずかな遅延を生じる。しかしながら、メディアストリーミングに先立ち、最初にこの情報を取り出すことにより、クライアントにメディア構造情報を供給するために（メディアストリーミング中に）追加の帯域幅コストが生じない。

【００７８】

ストリーミング開始に際しての前述の遅延は、通常、ストリーミングメディア全体の長さに対してごくわずかにすぎないことに留意されたい。例えば、ＰｅｅｒＳｔｒｅａｍｅｒの検査済みの実施形態では、サイズが３１メガバイト（ＭＢ）から４９ＭＢにわたる５つの検査用映画クリップが、約３７キロバイト（ＫＢ）から約５３ＫＢまでの範囲のメディア構造コンパニオンファイルを持っていた。したがって、そのメディア構造サイズは、メディア本体全体の約０．１０～０．１５％程度になることが観察されている。したがって、それらのサービング帯域幅がメディアビット速度以上であり、メディア構造がメディア本体の０．１５％であると想定すると、１０分間のクリップのメディア構造をダウンロードするのに０．９秒未満の追加遅延が生じる。

【００７９】

関連する一実施形態では、何らかの所定の長さ（すなわち、１０秒、３０秒、１分など）の連続するメディアセグメントのために部分的メディア構造が生成される。その場合、各部分的メディア構造は、対応するメディアセグメントが近い将来にストリーミングされる前にのみ取り出される。これは、メディア構造要求および送信がメディアパケット要求および送信と共存し得るために、帯域幅所要量をわずかに増大させる。しかしながら、メディア構造のサイズはこの場合非常に小さいため、帯域幅所要量全体に対する影響は、通常、無視できるほど小さい。

【００８０】

３．１．３　ＰｅｅｒＳｔｒｅａｍｅｒデータユニット

一実施形態では、ＰｅｅｒＳｔｒｅａｍｅｒは、メディアパケット、メディアヘッダおよびメディア構造を長さＬの固定サイズのデータユニットに分割する。固定サイズのデータユニットを使用する理由は、そうすれば、ＰｅｅｒＳｔｒｅａｍｅｒクライアントおよびサービングピアがサイズＬのメモリブロックを事前に割り振ることができ、ゆえに、ストリーミングプロセス中の高くつくメモリ割り振り動作を回避することができるからである。さらに、メディアパケット（非常に大きい可能性がある）を小さい固定サイズのデータユニットに分割すれば、ＰｅｅｒＳｔｒｅａｍｅｒクライアントが、より小さい細分度を用いてサービング負荷を各ピアに分散させることも可能になり、ピア間でより良い帯域幅負荷均衡化が達成される。

【００８１】

一般に、（メディアパケット、メディアヘッダまたはメディア構造とすることのできる）長さＰのパケットのサイズＬのブロックへの分割は、各パケットを［Ｐ／Ｌ］個のデータユニットに分割することによって達成され、その場合［ｘ］は、ｘ以上である最小の整

数を返す従来の天井関数である。その場合、おそらく、長さＰをＬで割った余りである各パケットの最後のデータユニットを除いて、すべてのデータユニットは固定長Ｌを有する。

【００８２】

メディアの非埋め込み符号化が使用される場合、各メディアパケットを含むデータユニットは、ネットワーク伝送中にメディア再生品質を損なわずに落とすことができない。したがって、これらのデータパケットは、すべてが受け取られなければならないため、「必須データユニット」と呼ばれる。

【００８３】

逆に、埋め込み符号化メディアパケットがデータユニットに分割されるときには、基本階層データユニットだけが配信されればよく、残りのデータユニットは、サービング帯域幅が不十分である場合、任意選択で落とすことができる。これら任意選択のデータユニットは、「非必須データユニット」と呼ばれる。非必須データユニットのサービングに必要な帯域幅は、次のように計算することができる。例えば、埋め込み符号化の場合には、１つのメディアパケットはＴ秒間続く。このメディアパケットがいくつかのデータユニットに分割されると仮定すると、このデータユニットを階層ｉでクライアントに供給するためには、階層ｉの下のすべてのデータユニットもクライアントに供給されなければならない。その結果、このデータユニットを階層ｉで供給するのに必要とされるサービング帯域幅は以下の通りである。

【００８４】

$R_i = (i+1)L/T$ 式１

したがって、式１は、埋め込み符号化メディアに関するデータユニットのビット速度Ｒを提供する。その場合、ＰｅｅｒＳｔｒｅａｍｅｒクライアントは、利用可能なサービング帯域幅を上回るビット速度を生じるはずの非必須データユニットを落とすことによって変化するサービング帯域幅に適応する。

【００８５】

メディアが非埋め込み符号化であれ、埋め込み符号化であれ、どちらの場合にも、メディアパケット、メディアヘッダおよびメディア構造のデータユニットを含む個々のメディアストリームのすべてのデータユニットが、一意のＩＤ空間にマップされる。例えば、検査済みの一実施形態では、メディアパケットのデータユニットは０ｘ００００００００から０ｘｆｄｆｆｆｆｆｆ（１６進数）まで索引付けされ、メディアヘッダのデータユニットは０ｘｆｅ０００００から０ｘｆｅｆｆｆｆｆｆまで索引付けされ、メディア構造のデータユニットは０ｘｆｆ０００００から０ｘｆｆｆｆｆｆｆｆまで索引付けされた。この検査済みの実施形態で使用されたＰｅｅｒＳｔｒｅａｍｅｒのデータユニットを図５に示す。

【００８６】

メディアヘッダおよびメディア構造のデータユニットＩＤを獲得するには、まず、メディアヘッダおよびメディア構造の長さが必要とされることに留意されたい。これらを、これらの「メガ構造」と呼ぶ。メディアパケットのデータユニットＩＤを獲得するには、メディアパケットビットストリームの長さが必要とされる。この情報は、メディア構造に含まれる。

【００８７】

３．１．４　メディアの部分キャッシング

サービングのためには、各サービングピアは、そのサービング帯域幅に比例するメディアの一部を保持するだけでよい。しばしば、インターネットに接続された大部分のコンピュータのサービング（またはアップロード帯域幅）は、実質的に、（各個別ノードが受け取ることのできる最高のストリーミングビット速度を決定付ける）そのダウンロード帯域幅を下回ることがある。その結果として、インターネット上の各エンドユーザノードは、そのアップロード帯域幅とそのダウンロード帯域幅の間に不均衡を有する傾向がある。例えば、ホームユーザから利用可能な典型的な市販のＡＤＳＬ／ケーブルモデムネットワー

ク上のノードが与えられたと仮定すると、そのダウンロード帯域幅がそのアップロード帯域幅より１桁高いことは珍しいことではない。同様に、大学構内／企業ネットワーク上のノードは、通常、任意の所与のノードのＰ２Ｐ型アクティビティへの参加が、他の不可欠の諸機能に影響を及ぼさないように、サービング帯域幅に上限を定めている。

【００８８】

その結果として、各サービングピアは、通常、個別に、クライアントにメディアストリーム全体を供給することができないため、どの１つのサービングピア上でもメディアストリーム全体をキャッシュする必要はない。したがって、サービングピアのいずれかによって必要とされる記憶リソースの量を低減させる１つの有効な方法は、各サービングピアに、ストリーミングされるメディアの一部だけを保持させることである。例えば、非埋め込み符号化メディアをストリーミングするのに必要とされるビット速度がＲであり、あるストリーミングセッションにおいて１つのピアによって提供される最大サービング帯域幅がＢである場合、各ピアノードは、そのキャッシュにそのストリーミングメディアのｐ部分さえ保持すればよく、その場合、値ｐは以下の式２によって示される。

【００８９】

$$p = \underline{\min}(1.0,\ B/R) \qquad \text{式２}$$

例えば、メディアビット速度がサービング帯域幅の２倍、すなわちＲ＝２Ｂであると仮定する。その場合、サービングピアは、その記憶にストリーミングメディアの半分を保持すればよい。というのは、そのピアだけでは、完全なストリーミングビット速度でクライアントにサービングすることができないからである。実際、この例の前述の制限事項が与えられたと仮定すると、このピアが行い得る最善のことは、せいぜいそのメディアの半分を供給することである。したがって、このピアは、そのキャッシュにただメディアの半分を保持すればよい。その場合、ストリーミングされるメディアの残りは、他方のサービングピアによって供給されなければならない。

【００９０】

さらに、その場合、式１と式２を組み合わせると、埋め込み符号化メディアが使用される場合に保持すべきメディアの量の決定が可能になることに留意すべきである。第３．１．３項で論じたように、埋め込み符号化メディアのメディアパケットは、異なるビット速度を有するいくつかのデータユニットに分割される。したがって、Ｒが、特定の階層Ｌでのデータユニットのビット速度である場合、式２は、そのデータユニットのために保持すべきメディアの部分をもたらす。例えば、図６で示すように、埋め込みメディアパケットは、複数のデータユニット（この例では８個）に分割され得る。各データユニットごとにキャッシュする必要のあるメディアの量（Ｌ／Ｔ＝０．５Ｂの場合）は、その場合、図６に示すように、式２に従って求められる。

【００９１】

しかしながら、一実施形態では、個々のサービングピアの記憶リソースが十分に大きい場合、そのサービングピアは、単に、式２においてより高い「潜在的サービング帯域幅」Ｂ’を使用することによって、そのメディアのより大きい部分をキャッシュするよう選択することができる。その場合、キャッシュされたそのメディアの余分な部分は、そのメディアを、途切れ途切れにではあっても高品質で供給することを可能にする。例えば、各サービングピアが、その実際のサービング帯域幅の二倍の潜在的サービング帯域幅Ｂ’、すなわちＢ’＝２Ｂを使用することを選択したと仮定すると、結果として生じるＰ２Ｐネットワーク中のメディアの量は、クライアントが半分のストリーミング速度でそのメディアを取り出すのに十分な量になる。言い換えると、利用可能なピアすべてのサービング帯域幅の総計がＲ／２より大きいと仮定すると、クライアントは、まず、そのメディアの半分をダウンロードし、次いで、残りの半分を続けてストリーミングし、再生することができるはずである。同様に、クライアントは、（時間T_sを用いて）メディアのT_s／２セグメントをダウンロードし、続けて次のT_s／２セグメントをストリーミングし、そのセグメントを再生し、次いで、次のセグメントをダウンロードし、ストリーミングするよう選択することもできる。ゆえに、ストリーミングメディアは、途切れ途切れにではあるが、速

度Ｒで再生され得る。

【００９２】

３．１．５　メディアの高速消去符号化

前述のように、各ピアは、本来的に信頼性が低い可能性がある。したがって、本来的に信頼性の低いサービングピアのサービング挙動を有効に処理するために、ＰｅｅｒＳｔｒｅａｍｅｒシステムおよび方法においてさらなる冗長性を実現する何らかの手段を提供すれば有利である。この問題を処理することは、対処されなければならないいくつかの問題を提起する。例えば、各ピアによってメディアのどの部分ｐが保持されるべきか決定することは問題である。さらに、メディアは、最終的に、前述のデータユニットに分割されるため、各ピアがデータユニットのどの部分ｐを維持すべきか決定することも問題である。

【００９３】

これらの問題に対処する１つの戦略は、各データユニットを単純にｋ個のブロックに分けるというものである。その場合、メディアのｐ部分を保持するピアは、［ｘ］が前述の天井関数であるとすれば、ランダムに［ｋ・ｐ］ブロックを保持することができる。しかしながら、この方式のランダム性に伴う１つの問題は、ピアクラスタ中にｋよりはるかに多くの利用可能なブロックがあったとしても、そのクラスタが、全体として、特定のブロックｊを欠く可能性があり、それによって、そのデータユニット全体が取り出し不可能になることがあり得るということである。さらに、そのような方式では、クライアントは、それでもなお、各ピアからあらゆる別個のブロックを探し出す役割を果たし、それが、クライアントとピアの間のプロトコルの設計を複雑にする。

【００９４】

したがって、１つのより優れた戦略は、「高速消去耐性符号」を使って、各ピアの１つまたは複数が個々のデータユニットを再構築するのに必要なデータブロックを持つようにすると同時に、各ピアのどれが必要なデータを含むか識別するクライアント側での要求を簡略化するというものである。一般に、消去耐性符号とは、パラメータ（ｎ，ｋ）を有するブロック誤り訂正符号であり、その場合ｋは元のメッセージの数であり、ｎは符号化メッセージの数である。高速消去耐性符号は、ｎがｋよりはるかに大きく、ゆえに、ｋ個の元のメッセージがｎ個のメッセージのはるかに大きい符号化メッセージ空間に拡張されるという属性を満たす。消去符号化技法は、一般に、データ符号化ではかなりよく知られているが、本明細書で説明する、Ｐ２Ｐネットワーク環境におけるメディアストリーミングでのそのような技法の適用は、知られていない。

【００９５】

ブロック誤り訂正符号として、高速消去耐性符号の動作は、ガロア体ＧＦ（ｐ）に対する行列乗算によって説明することができる。

【００９６】

【数１】

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \vdots \\ c_{n-1} \end{bmatrix} = \mathbf{G} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{k-1} \end{bmatrix}$$

式３

【００９７】

式中、ｐは、ガロア体の次数であり、｛x_0，x_1，．．．，x_{k-1}｝は元のメッセージであり、｛c_0，c_1，．．．，c_{n-1}｝は符号化メッセージであり、Ｇは生成行列である。式３は、符号化メッセージすべてを一度に生成するのには使用されないことに留意されたい。そうではなく、生成行列Ｇは、１つの符号化メッセージ空間を定義する。したがって、クライアントがｋ個の符号化メッセージ｛c'_0，c'_1，．．．，c'_{k-1}｝を受け取るとき、それらは式４によって以下のように表すことができる。

【００９８】
【数２】

$$\begin{bmatrix} c'_0 \\ c'_1 \\ \vdots \\ c'_{k-1} \end{bmatrix} = \mathbf{G}_k \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{k-1} \end{bmatrix}$$

式４

【００９９】
式中、G_kは、符号化メッセージに対応する生成行列Ｇのｋ行によって形成される従属生成行列である。さらに、従属生成行列Ｇｋが完全な階数ｋを持つ場合、行列G_kは反転させることができ、ゆえに、元のメッセージが復号化され得る。
【０１００】
例えば、リードソロモン消去符号、トルネード符号、およびＬＰＤＣコードなどを含めて、使用し得るいくつかのよく知られた消去符号化技術がある。しかしながら、一実施形態では、ＰｅｅｒＳｔｒｅａｍｅｒは、ガロア体ＧＦ（2^{16}）上の変更されたリードソロモン符号に基づく新規の高速消去耐性符号を提供する。この例では、元のメッセージの数ｋは１６である。符号化メッセージ空間のサイズｎは2^{16}＝６５５３６である。リードソロモン符号は、最大距離分離（ＭＤＳ）符号である。その結果として、生成行列Ｇの任意の１６行は、完全な階数１６を持つ従属行列を形成する。言い換えると、元のメッセージは、任意の１６個の符号化メッセージから回復され得る。他の体サイズ、ｐも使用され得ること、およびＰｅｅｒＳｔｒｅａｍｅｒは、本明細書で説明する特定の体サイズの使用に限定されないことに留意すべきである。さらに、非ＭＤＳ消去符号化を使用する実施形態では、使用される特定の消去符号化によっては、元のメッセージを回復するのにｋ’≧ｋ個のブロックを取り出すことが必要となり得る。リードソロモンに基づく消去符号が一部使用されたのは、それらがＭＤＳ符号であり、大部分の従来のコンピュータのＣＰＵに対してわずかな計算上のオーバーヘッドをもたらすだけで効率よく符号化され、復号化され得るからである。
【０１０１】
高速（ｎ，ｋ）消去耐性符号を用いる場合、各ピアノードには、ｎの符号化メッセージ空間においてｋ個のキーが割り当てられ、各キーは生成行列Ｇの行索引である。キー割り当ては、サーバによって実行され得る。さらに、メディアをキャッシュするピアの数がｎ／ｋより小さい場合、各ピアに一意のキーセットを割り当てることが可能である。その結果、各ピアが特有の符号化メッセージを保持することが保証され得る。この戦略はいくつかの利点を提供するが、なお、中央調整ノード（サーバなど）を必要とする。
【０１０２】
したがって、別の実施形態では、各ピアにｋ個のランダムキーを選択させることによって中央調整ノードの役割を不要にする。ピアノードの数がｎ／ｋより大きく、またはキーが中央調整ノードなしで割り当てられる場合、いくつかのピアノードは、同じキーを保持し得る。それでもなお、クライアントがｍ個のピアに接続される大部分のメディアストリーミングセッションでは、ｍは、普通、ｎ／ｋよりずっと小さい。したがって、２つのサービングピアが偶然同じキーを保持しており、ゆえに、それらのピアの１つの１つのキーが有用でない確率は小さい。しかしながら、キー競合があったとしても、クライアントは、それが最初にそれらのピアに接続するときに、そのような競合を容易に識別することができる。そのような競合が識別される場合には、クライアントは、単に、ストリーミングセッションの残りの部分で重複したキーの１つを無効化する。したがって、クライアントは、ストリーミングプロセスの間に実際にキー競合に対処する必要がない。
【０１０３】
例えば、Ｓ１およびＳ２が、それぞれ、サービングピア１およびサービングピア２の消去符号化キー空間であり、Ｓ１＝｛１，７，２３，４３，４８｝およびＳ２＝｛３，７，

２８，４９，９９｝であると仮定する。明らかに、キー空間Ｓ１およびＳ２は異なる。しかしながら、キー７は２つのキー空間によって共有されており、したがって、サービングピア１およびサービングピア２は、同じキー、すなわちキー「７」を共有する消去符号化ブロックを保持し得る。したがって、個々の符号化ブロックを要求する前に、それらのサービングピアの一方についてキー「７」が無効化されて、キー「７」によって符号化されたブロックがそれらのピアの一方だけから取り出され、重複キーによって生じる復号化競合が回避されるようにする。しかしながら、メディアストリーミング動作中に１つのサービングピアがオフラインになった場合、別のサービングピアの特定の無効とされた符号化キーは、そのオフラインのサービングピアが、以前に、１つまたは複数の重複キーを使用した結果として競合していた場合には、再有効化され得ることに留意すべきである。

【０１０４】

（６５５３６，１６）リードソロモン符号を用いる場合、各データユニットは、１６ブロックに分割される。事前に割り当てられたキーセットを使って、ピアは、［１６ｐ］消去符号化ブロックをキャッシュすることを選択する。その場合ｐは、式１および２から計算されたパラメータである。ピアに割り当てられたキー、およびその最大サービング帯域幅Ｂは、前述のそのピアの可用性ベクトルを構成する。というのは、クライアントは、そのピアの可用性ベクトルによって提供された情報を使って、何個のどんな消去符号化ブロック（データユニット／ブロックＩＤによる）がそのピアによって保持されているか判定することができるからである。やはり、クライアントは、各ピアが最初に接続される時に、キー競合があればそれを解決する。そのストリーミングセッションの間、次いで、クライアントは、任意のサービングピアノードから任意のｋ個の符号化メッセージを取り出し、関連するデータユニットを復号化することができる。

【０１０５】

さらに、特定のデータユニットを復号化するために任意の１つのサービングピア上にそれらの符号化ブロックの集合全体を格納する必要はない。言い換えると、任意の個々のデータユニットのために任意の個々のサービングピアによって保持されるブロックの数はｋ未満とすることができる。したがって、あらゆる符号化キーでのあらゆる符号化ブロックを計算するのに計算能力を浪費するのではなく、一実施形態では、実際に特定のピアに配信される符号化ブロックだけが生成される。言い換えると、ｊ＜ｋ個のブロックが特定のサービングピア上に格納される場合、その特定のデータユニットにはｊ個のブロックだけが生成されるはずである。

【０１０６】

３．２　Ｐ２ＰネットワークにおけるＰｅｅｒＳｔｒｅａｍｅｒ動作の実装

以下の段落では、前述のＰｅｅｒＳｔｒｅａｍｅｒの動作詳細の説明を考慮して、ＰｅｅｒＳｔｒｅａｍｅｒ動作の実装について説明する。具体的には、以下の段落では、クライアントによるサービングピアの位置検出、取り出したメディア構造に基づくクライアント復号化およびレンダリングのセットアップ、ＰｅｅｒＳｔｒｅａｍｅｒネットワーク接続、ストリーミングビット速度制御、ＰｅｅｒＳｔｒｅａｍｅｒクライアント要求およびピア応答、および最後に、ＰｅｅｒＳｔｒｅａｍｅｒ要求およびステージングキューについて説明する。

【０１０７】

３．２．１　サービングピアの位置検出

前述のように、クライアントによって実行される最初のタスクは、サービングメディアの完全な、または一部のコピーを保持する近隣のサービングピアのリストのＩＰアドレスおよびリスニングポートを獲得することである。さらに、このリストは、メディアストリーミングセッション中にも更新される。前述のように、このリストを獲得する一般的な手法には、１）サーバからリストを取り出すこと、２）知られているサービングピアからリストを取り出すこと、および３）分散ハッシュ表（ＤＨＴ）手法を使って、事前に、そのメディアサーバもサービングピアも知られていないサービングピアを識別することが含まれる。

【０１０８】
３．２．２　復号化およびレンダリングセットアップ
サービングピアリストを確保した後で、クライアントは、それらのサービングピアのそれぞれに接続しようと試みる。接続されると、クライアントは、各ピアの可用性ベクトルを取り出し、前述のように、キー競合があればそれを解決する。次いで、クライアントは、ピアの１つからメディアヘッダおよびメディア構造の長さを取り出す。両方の長さが取り出された後、メディアヘッダおよびメディア構造のデータユニットのＩＤが構築される。次いで、メディアヘッダおよびメディア構造を、第３．２．６項でさらに詳細に説明するように、Ｐ２Ｐで取り出すことができる。メディアヘッダが取り出されると、クライアントは、メディアがクライアントにストリーミングされる際にそれを復号化し、レンダリングするためにどのデコーダおよびレンダラを初期設定すべきか決定する。
【０１０９】
ＤｉｒｅｃｔＸ（商標）を使って実施された検査済みの実施形態では、このセットアップは、まず、メディアヘッダで提供された情報からＤｉｒｅｃｔＳｈｏｗ（商標）フィルタグラフを構築することによって達成された。本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒはＤｉｒｅｃｔＸ（商標）機能を使った実装形態に限定されるものではなく、ＤｉｒｅｃｔＸ（商標）の使用、および検査済みの実施形態に関連するその説明は、クライアント再生のためのストリーミングメディアの復号化、レンダリングに際してのクライアントコンピュータのセットアップを説明するために提供したにすぎないことに留意すべきである。
【０１１０】
したがって、クライアントセットアップのためのＤｉｒｅｃｔＸ（商標）実装形態を仮定して、クライアントのネットワークコンポーネントがＤｉｒｅｃｔＳｈｏｗ（商標）ネットワークソースフィルタによって表され、その出力が適正なオーディオ／ビデオ復号ＤｉｒｅｃｔＸ（商標）メディアオブジェクト（ＤＭＯ）に供給される。次いで、このＤＭＯがさらに、適当なオーディオ／ビデオレンダリングデバイスに接続される。例えば、クライアントＰｅｅｒＳｔｒｅａｍｅｒメディアストリーミングセッションのＤｉｒｅｃｔＳｈｏｗ（商標）フィルタグラフの一例を図７に示す。この例では、ストリームされるメディアは埋め込み符号化されない。オーディオビットストリームはＷＭＡで圧縮され、ビデオビットストリームはＭＰＥＧ－４で圧縮される。
【０１１１】
ＤｉｒｅｃｔＳｈｏｗ（商標）フレームワークによるＰｅｅｒＳｔｒｅａｍｅｒクライアントセットアップ実装を使用する１つの利点は、それが、ＤｉｒｅｃｔＳｈｏｗ（商標）の下で開発された既存のオーディオ／ビデオエンコーダ／デコーダの膨大なライブラリを利用できることである。例えば、ＤｉｒｅｃｔＳｈｏｗ（商標）を用いると、ＰｅｅｒＳｔｒｅａｍｅｒクライアントは、例えば、ＭＰＥＧ１／２／４、ＷＭＡ／ＷＭＶ、Ｉｎｄｅｏ　Ｖｉｄｅｏなど、あるいはＤｉｒｅｃｔＳｈｏｗ（商標）復号ＤＭＯコンポーネントを備える他の任意のコーデックなどを含む、様々なコーデックによって符号化されたメディアを復号化し、レンダリングすることができる。ＤｉｒｅｃｔＳｈｏｗ（商標）は、復号化されたオーディオ／ビデオを、クライアントのオーディオ／ビデオレンダリングデバイスの機能に自動的にマッチさせることができるように、解像度／色空間変換やインターレース解除(deinterlacing)といったその他のオーディオ／ビデオ処理モジュールも提供することができる。
【０１１２】
さらに、ＤｉｒｅｃｔＳｈｏｗ（商標）は、オーディオ／ビデオトラックの同期を自動的に処理する。例えば、オーディオストリームがストリーム全体の基準クロックを保持している場合、ストリーミングビデオを再生するときに、ＤｉｒｅｃｔＳｈｏｗ（商標）は、リップシンクなどのなどの問題に対処するために、ビデオストリームのシステムタイミングクロックが、オーディオストリームのクロックの可能な限り近い状態にとどまるようにする。最後に、ＤｉｒｅｃｔＳｈｏｗアプリケーションは、本来的に、マルチスレッド

化されている。その結果として、マルチプロセッサＰＣ（またはハイパースレッド処理対応のもの）上では、例えば、ネットワークコンポーネント、オーディオデコーダ、ビデオデコーダ、オーディオ／ビデオレンダリングエンジンなどのといったクライアントの様々なコンポーネントの計算負荷が、複数のプロセッサに分散され得る。これは、クライアントの実行速度を大幅に向上させ、より複雑なオーディオ／ビデオデコーダを使用できるようにする。

【０１１３】

最後に、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒは、ＤｉｒｅｃｔＸ（商標）機能を使った実装形態に限定されるものではなく、ＤｉｒｅｃｔＸ（商標）の使用、および検査済みの実施形態に関連するその説明は、クライアント再生のためのストリーミングメディアの復号化、レンダリングに際してのクライアントコンピュータのセットアップを説明するために提供したにすぎないことに再度留意すべきである。

【０１１４】

３．２．３　ＰｅｅｒＳｔｒｅａｍｅｒネットワークリンクおよびパケット損失管理

Ｗｉｎｄｏｗｓ（登録商標）Ｍｅｄｉａ　ＰｌａｙｅｒやＲｅａｌＰｌａｙｅｒ（登録商標）といった大部分のメディアストリーミングクライアントは、ＵＤＰの上で搬送される、よく知られているリアルタイムトランスポートプロトコル（ＲＴＰ）を使用する。ＵＤＰ／ＲＴＰプロトコルが、通常、メディアストリーミングアプリケーションに選択されるのは、１）ＵＤＰプロトコルはＩＰマルチキャストをサポートし、これは、ＩＰマルチキャスト対応のネットワーク上のノードの集合にメディアを送信する際に効率的であることがあり、２）ＵＤＰプロトコルはどんな再送信機能もデータ速度管理機能も備えないからである。その結果として、ストリーミングサーバおよびクライアントは、メディアパケットのタイムリーな配信を保証するために、前方誤り訂正（ＦＥＣ）など、高度なパケット配信機能を実施することができる。

【０１１５】

しかしながら、前述のよく知られているメディアストリーミング方式とは対照的に、ＰｅｅｒＳｔｒｅａｍｅｒは、クライアントとサービングピアの間のネットワークリンクとしてＴＣＰ接続を使用する。従来のＵＤＰ／ＲＴＰプロトコルではなくＴＣＰ接続を選択する１つの理由は、ＩＰマルチキャストは、現実には、ドメイン間経路指定プロトコル、ＩＳＰビジネスモデル（課金モデル）、配信ツリーに沿った輻輳制御などなどの問題のために、あまり普及していないことである。

【０１１６】

また、多くの市販のメディアプレイヤーのように、ＰｅｅｒＳｔｒｅａｍｅｒクライアントにも、ジッタや輻輳などのネットワーク異常に対処するために、（検査済みの実施形態では４秒間の）ストリーミングメディアバッファを組み込んである。実際、クライアントとサービングピアの間のラウンドトリップ時間（ＲＴＴ）を何倍も上回るストリーミングメディアバッファが与えられたと仮定すると、ＴＣＰ　ＡＲＱ（自動反復要求）機構は、ストリーミングメディアのスムーズな再生を実現するのに十分な時間でのメディアパケットの配信に十分適する。

【０１１７】

一般に、メディアパケット損失に対処するための（多数のよく知られた変形形態を伴う）３つのよく知られた機構がある。例えば、これらの機構には、一般に、ＦＥＣ、選択的パケット再送信、および自動反復要求（ＡＲＱ）が含まれる。これらのパケット損失機構のいずれもＰｅｅｒＳｔｒｅａｍｅｒによって使用され得る。しかしながら、以下で説明するように、特定の機構を使用することには他に優る利点がある。

【０１１８】

具体的には、変動する特性および不明のパケット損失率を伴う消去チャネルとみなすことのできるインターネットチャネルでは、固定されたＦＥＣ方式は、（保護が過多である場合）帯域幅を浪費し、または（保護が過少である場合）失われたパケットを回復することができない。ゆえに、これは、クライアントとピアの間の帯域幅リソースを効率よく利

用しない。したがって、ＲＴＴより何倍も大きいストリーミングバッファを用い、ゆえに、十分な再送信の機会を有する場合には、再送信に基づく誤り保護（選択的再送信やＡＲＱなど）がＦＥＣより好ましい。

【０１１９】

ＡＲＱおよび選択的再送信を考察すると、ＴＣＰプロトコルを使用するインターネットチャネルでは、多数のパケットが再送信されるよう選択されていない場合には、選択的再送信がＡＲＱより優れている。非埋め込み符号化メディアでは、失われたパケットは、普通、個々のパケットを復号化、再生できないことを含めて、重大な再生劣化をもたらす。したがって、失われたパケットは、ほとんどの場合再送信される。逆に、埋め込み符号化メディアを用いる場合、失われたパケットは、そのメディアが再生されるのを妨げない可能性がある。しかしながら、ランダムパケットを損失すると、やはり、いくつかの派生パケットが使用不能になる。その結果、最上拡張階層パケットだけが、再送信されるよう選択できなくなる。

【０１２０】

選択的再送信と比べると、ＡＲＱは、パケットが要求されると、それらが最上拡張階層に属するものであっても、常にそれらを再送信する。それでもなお、ＡＲＱ方式は、後続のメディアパケットの最上拡張階層パケットを要求しないよう選択することができ、ゆえに、選択的送信方式と同じ帯域幅利用度および知覚メディア再生品質を達成する。その結果として、ネットワーク状況が急速に変動しない限り、ＴＣＰプロトコルによって用いられるＡＲＱ機構は、メディアストリーミングにおけるパケット損失を処理するのに十分である。

【０１２１】

ＴＣＰをネットワークプロトコルとして使用することは、前述のような従来のメディアストリーミング方式に優るいくつかの他の利点も提供する。例えば、ＴＣＰを用いれば、フロー制御、スループット推定、輻輳制御および回避、キープアライブなどを明示的に処理する必要がない。これらの問題すべてがＴＣＰプロトコルによって自動的に処理される。また、ＴＣＰプロトコルは、ピアがオフラインになるのを検出することもでき、ピアとクライアントの間の接続リンクのシャットダウンを安全に処理することもできる。

【０１２２】

３．２．４　埋め込み符号化を用いたＰｅｅｒＳｔｒｅａｍｅｒストリーミングビット速度制御

非埋め込み符号化メディアは、クライアント側でのメディア再生の劣化を防ぐために、好ましくは常に、そのメディアのビット速度でストリーミングされる。しかしながら、埋め込み符号化メディアのストリーミングビット速度は、ストリーミングセッション中に変動し得る。

【０１２３】

したがって、一実施形態では、各埋め込み符号化メディアパケットごとのストリーミングビット速度R_{recv}が、まず、式５、６、７によって以下のように計算される。

【０１２４】

$$R_{raw}=Th\cdot(1+T_{rft}-T_{staging})+B_{staging}-B_{outstanding} \quad \text{式５}$$

$$R_{filter}=(1-\alpha)R_{filter}+\alpha R_{raw} \quad \text{式６}$$

$$R_{recv}=min(R_{min},R_{inst}) \quad \text{式７}$$

式中、Ｔｈは複数のサービングピアの合計サービング帯域幅であり、$T_{staging}$は（検査済みの実施形態では２．５秒のデフォルトの）目標ステージングバッファサイズであり、T_{rft}は（検査済みの実施形態では１．０秒のデフォルトの）所望の要求完了時間であり、$B_{staging}$はステージングキュー中の受信パケットの長さであり、$B_{outstanding}$は受け取られる未処理の応答の長さであり、R_{min}は（必須データユニットだけを伴う）基本階層ビット速度であり、αは低域制御パラメータである。

【０１２５】

次いで、式５～７の結果を使い、以下の第３．２．６項でさらに詳細に説明する、合計

サービング帯域幅Ｔｈおよびステージングおよび要求キュー状況に従って、ストリーミングビット速度R_{recv}が制御される。ストリーミングビット速度が求められると、クライアントは、ストリーミングビット速度R_{recv}を下回るビット速度を持つデータユニットを求める要求だけを発行する。

【０１２６】

関連する一実施形態では、より高度な戦略を使い、データユニットのひずみ貢献も考慮することによって、ビット速度R_{recv}を制御する。しかしながら、これは、クライアントがデータユニットのひずみ（または速度ひずみ勾配）にアクセスできることを必要とし、それがメディア構造に含められ、クライアントに送られなければならない。しかしながら、メディア構造中の既存の情報とは異なり、データユニットのひずみは、復号化の際には必要とされず、ゆえに、追加のオーバーヘッドであるとみなされる。その結果として、これは、クライアントに送られるオーバーヘッドの量と速度制御の正確さの間のトレードオフになる。

【０１２７】

３．２．５　ＰｅｅｒＳｔｒｅａｍｅｒデータブロック要求および応答

クライアントデータブロック要求およびそのピアによる応答の寿命(life)を、一般的に、図８に示す。具体的には、図８に示すように、クライアントは、要求を生成し、それを、アウトバウンドＴＣＰ接続を介して特定のサービングピアに送信する。さらに、ネットワーク配信では、ＴＣＰは、その要求を、同じピアに発行された前の要求とまとめることができる。前の要求が送信中に失われた場合、ＴＣＰは、その要求の再送信も処理する。

【０１２８】

パケット要求がピアに配信された後、それはそのサービングピアのＴＣＰ受信バッファに格納される。次いで、そのピアは、それらの要求を一度に１つずつ処理する。各要求ごとに、ピアは、そのディスクまたはメモリ記憶から（使用される符号化に応じて、消去符号化されていることも、されていないこともある）要求されたブロックを読み取り、要求されたコンテンツをクライアントに返送する。サービングピアからクライアントへのＴＣＰソケットがブロックされている、すなわち、それ以上の帯域幅が利用できない場合、サービングピアは、そのＴＣＰ接続が空くまで、それ以上のクライアント要求をブロックする。

【０１２９】

要求がクライアントによって発行された時刻とその応答がクライアントによって受け取られた時刻の間隔は、要求完了時間（ＲＦＴ）として定義される。要求は、普通、その応答よりずっと小さく、要求を処理するのに必要な諸動作、例えば、ディスク読取りなどは、通常、そのコンテンツを返送するのに使用されるネットワーク配信時間に比べると些細なものである。したがって、要求のＲＦＴ、T'_{rft}は、式８によって以下のように計算される。

【０１３０】

【数３】

$$T'_{rft} = (B_{i,\,outstanding} + B_{cur}) / Th_i \qquad \text{式８}$$

【０１３１】

式中、Th_iはピアｉのサービング帯域幅であり、$B_{i,outstanding}$はその要求の前の受け取られていない応答の長さであり、B_{cur}は要求されたコンテンツの長さである。したがって、ＲＦＴは、ピアのサービング帯域幅、要求のサイズおよびピアからの受け取られていないコンテンツのサイズの関数として求められる。

【０１３２】

要求されたコンテンツパケットがクライアントに到着すると、それは直ちにステージングキューに移動される。ステージングキューでは、複数のピアからの（消去符号化ブロックを含み得る）データブロックが組み合わされ、復号化されてデータユニットになり、それらがさらに組み合わされてメディアパケットになる。クライアントは、ステージングキ

ューから定期的に配信されたメディアパケットを取り出し、それらを対応するオーディオ／ビデオデコーダにプッシュする。メディアパケットがデコーダによって伸張された後、クライアント再生装置（表示モニタ、スピーカなど）上でのストリーミング再生のために、圧縮されていないオーディオ／ビデオデータストリームがオーディオ／ビデオレンダリング装置に送られる。

【０１３３】

一実施形態では、図８に示すバッファを使って、パケット損失やジッタなどのネットワーク異常に対処する。（しかしながら、ＤｉｒｅｃｔＳｈｏｗ（商標）実装形態を使用するとき、圧縮されていないオーディオ／ビデオバッファはＤｉｒｅｃｔＳｈｏｗフィルタグラフの制御下にあり、プログラム不能である）。ＰｅｅｒＳｔｒｅａｍｅｒの検査済みの一実施形態では、ステージングバッファのサイズは、$T_{staging}$＝２．５秒に設定され、所望のＲＦＴはT_{rft}＝１．０秒に設定され、圧縮オーディオ／ビデオバッファは０．５秒に設定された。その結果として、この検査済みの実施形態では、ＰｅｅｒＳｔｒｅａｍｅｒクライアントの合計バッファは約４秒になる。

【０１３４】

消去符号化が使用される実施形態では、各データブロック要求は、ある一定のデータユニットの消去符号化ブロックのグループの要求として定式化される。この消去符号化ブロックグループは、開始ブロック索引および要求されるブロック数で識別可能である。データユニットは、３２ビットＩＤによって識別可能である。ゆえに、要求は、

Ｄａｔａ＿Ｕｎｉｔ＿ＩＤ［３２］，Ｓｔａｒｔ＿Ｉｎｄｅｘ［４］，Ｎｕｍｂｅｒ＿ｏｆ＿Ｂｌｏｃｋｓ［４］　　式９

の形を取り、式中、括弧内の数は、各コンポーネントのビット数である。

【０１３５】

したがって、式９で示すように、消去符号化ブロックの場合には、各要求は５バイト長である。他方、要求されるコンテンツのサイズは、１２８から２０４８バイトまでに及ぶ（データユニットの長さＬ＝２０４８、ｋ＝１６）。その結果、要求のサイズは、応答の約０．２４から３．９１％にすぎない。したがって、クライアントが要求を送信するのに費やされるアップロード帯域幅の量は、要求されるコンテンツに比べて非常に小さい。

【０１３６】

３．２．６　ＰｅｅｒＳｔｒｅａｍｅｒ要求およびステージングキュー

前述のように、ＰｅｅｒＳｔｒｅａｍｅｒクライアントは、（消去符号化されている可能性のある）受信データブロックを保持するために単一のステージングキューを維持し、そこからデータブロックが組み立てられてデータユニットになり、次いで、メディアパケットになる。クライアントは、各ピアに送られた未完了の要求を保持するために、各サービングピアごとの別個の要求キューも維持する。これらの要求およびステージングキューの一例を図９に示す。

【０１３７】

ステージングキューは、ＰｅｅｒＳｔｒｅａｍｅｒクライアントの主ストリーミングバッファである。すべての受信コンテンツは、まず、ステージングキューに置かれる。要求キューは、１）スループット制御および負荷均衡化を実行する、２）各サービングピアによって返送された応答を識別する、および３）切断されたピアを処理するという３つの目的を果たす。

【０１３８】

要求キューの第１の機能は、サービングピア間で負荷を均衡化させることである。メディアが消去符号化されている場合、データユニットを求める要求は、各グループが１つのピアに宛先指定されている、消去符号化ブロックの複数のグループの要求に分割される。これらの要求は、以下の動作によって生成される。データユニットを要求すると、クライアントは、まず、各ピアの可用性ベクトルをチェックし、そのデータユニットについて各ピアによって保持されている消去符号化ブロック数（a_i）を計算する。すべてのオンラインのピアによって保持されている合計ブロック数がｋ未満である場合、そのデータユニ

ットは取り出し不能である。その取り出し不能データユニットが非必須である（すなわち、埋め込み符号化メディアの基本階層ではない）場合、クライアントは、単に、そのデータユニットをスキップするだけである。

【０１３９】

逆に、その取り出し不能データユニットが必須である、すなわち、非埋め込み符号化メディアパケット、または埋め込み符号化メディアパケットの基本階層に属する場合、クライアントは、そのストリーミングメディアのダウンロードおよび再生を開始することができない。したがって、一実施形態では、クライアントは、欠けているブロックを供給するためにより多くのピアがオンラインになるのを待つ。別の実施形態では、クライアントは、そのメディア全体をスキップし、次のオーディオ／ビデオデコーダに対して、それが欠けているとマークする。結果として、レンダリングされるメディアにはギャップまたはスキップが生じることになる。しかしながら、１つの必須データユニットがそのピアクラスタから取り出し不能である場合、それ以外の後続の必須データユニットも取り出し不能である可能性が非常に高い。したがって、通常は、ユーザにより良い再生体験を提供するために、そのデータが利用可能になるまでクライアントに待たせる方がよい。

【０１４０】

個々のデータユニットが取り出し可能であること、すなわち、

$$\Sigma_i a_i \geqq k \qquad \text{式１０}$$

を確認した後、クライアントは、各ピアの要求キュー中での利用可能な空間をチェックする。各ピアのＲＦＴを、システム定数T_{rft}程度に維持することが望ましい。検査済みの一実施形態では、約１．０秒程度のT_{rft}が良い結果をもたらした。（短すぎる要求キューを使用すると、クライアントからピアまでの帯域幅を効率よく利用することができないことに留意されたい。）

特に、クライアントによって送信される要求パケットが失われ、または遅延した場合、サービングピアは、送信するものがない状態のままに置かれる可能性があり、これはそのサービング帯域幅を無駄にする。逆に、過度に長い要求キューを使用すると、クライアントが、ピアの１つの切断などの変化に迅速に適応することが妨げられる可能性がある。さらに、すべてのピアでの要求キューがＲＦＴにおいて同じ長さである場合、要求キューの容量は、そのサービング帯域幅に比例し、すなわち、$Th_i \cdot T_{rft}$になる。

【０１４１】

例えば、T_{rft}が１．０秒であると仮定すると、１６ｋｂｐｓのサービング帯域幅を持つピアは、２ＫＢの未完了要求をその要求キュー中で待機させることができ、１Ｍｂｐｓのサービング帯域幅を持つピアは、１２８ＫＢの未完了要求を待機させることができる。ゆえに、個々のピアに要求することのできる消去符号化ブロックの数は、以下のように、その要求キューに残されている空間によって上限が決まる。

【０１４２】

$$e_i = \min(\alpha_i, (Th_i \cdot T_{rft} - B_{i,outstanding}) / bk) \qquad \text{式１１}$$

式中、e_iは、ピアｉに要求することのできる消去符号化ブロックの数であり、ｂｋは、消去符号化ブロックのサイズである。

【０１４３】

式１１は、クライアントが、T_{rft}より大きい期待されるＲＦＴを持つ要求を決して送信しないことを保証する。クライアントが、十分な現在利用可能な消去符号化ブロックを見つけられない場合、すなわち、

$$\Sigma_i e_i < k \qquad \text{式１２}$$

である場合、クライアントは、サービングピアの要求キューがクリアされるまで待機する。データユニット要求は、$\Sigma_i e_i \geqq k$であるときにだけ形成され、各ピアに送信される。特定のピアに要求されるブロックの実数（b_i）は、以下のように計算される。

【０１４４】

【数４】

$$\begin{cases} \sum_i b_i = k, \\ b_i = \min(e_i, c \cdot Th_i) \end{cases}$$

式１３

【０１４５】

式中、ｃは$\Sigma_i b_i = k$を満たす定数である。

【０１４６】

一般に、前述の手順は、各ピアに、そのサービング帯域幅Th_iに比例してサービング負荷を割り振る（式１３）。また、これは、クライアントが、個々のサービングピアに、そのサービングピアによって実際にキャッシュされ、または格納されているブロック数を上回るブロックを要求しないことも保証する。最後に、この手順は、式１１に示すように、要求のＲＦＴがT_{rft}を上回らないことも保証する。

【０１４７】

要求キューの第２の機能は、各サービングピアによって返送されるコンテンツを識別することである。前述のように、ＰｅｅｒＳｔｒｅａｍｅｒクライアントおよびピアはＴＣＰを介してやりとりし、これは、データ送信の順序を保存し、パケット配信を保証する。さらに、各ピアは、着信要求を順次処理する。その結果、返送されるコンテンツを明確に識別する必要がなくなる。というのは、それが、各ピアごとの要求キュー中で待機している最初の要求のためのものに他ならないからである。

【０１４８】

前述の要求キューの第３の機能について、この要求キューは、切断されたピアの要求の宛先変更にも使用される。例えば、特定のサービングピアがクライアントから切断されるたびに、ＴＣＰプロトコルによって切断イベントがピックアップされ、次いで、この切断がクライアントに報告される。次いで、クライアントは、切断されたピアのキューで待機中のすべての未完了要求を残りのピアの１つまたは複数に動的に再び割り当てる。要求を再割り当てする手順は、最初に要求を割り当てる手順に非常によく似ている。唯一の例外は、要求再割り当てにおいては、切断されたピアにすでに要求されたブロック数を考慮しなければならないことである。

【０１４９】

最後に、消去符号化ブロックがクライアントに到着するたびに、それらは直ちにＴＣＰソケットから引き離される。到着したコンテンツを待機中の要求と組み合わせた後、完了した要求が要求キューから除去される。次いで、識別された消去符号化ブロックがステージングキューに置かれる。その結果、ステージングキューのサイズが増大する。ステージングキューが所定のサイズ$T_{staging}$に達した場合、それ以上のメディアパケット／データユニットの要求は送信されない。あるデータユニットのすべての消去符号化ブロックが受け取られると、そのデータユニットは消去復号化され、作動可能とマークされる。メディアパケットは、その要求データユニットすべてが作動可能であれば作動可能になる。オーディオ／ビデオデコーダは、定期的に、ステージングキューから「作動可能」メディアパケットを取り出す。これは、ステージングキューのサイズを縮小させ、新しいメディアパケット要求の生成をトリガすることができる。

【０１５０】

次いで、前述のメディアストリーミング動作が、そのメディアファイルの再生が完了するまで、またはそのメディアをストリーミングするのに利用可能なピアの数が不足するようなときまで、あるいはユーザがそのストリーミングセッションを終了するまで続行される。

【０１５１】

３．３　ＰｅｅｒＳｔｒｅａｍｅｒ動作

前述の、図２から図９に関連する各プロセスを、図１０の全体動作流れ図に示す。図１０には、全体として、ＰｅｅｒＳｔｒｅａｍｅｒのいくつかの動作実施形態を示す例示的

動作流れ図が示されている。図１０に破線または点線で表す任意のボックスおよびボックス間の相互接続は、本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒの代替実施形態を表すこと、およびこれらの代替実施形態のいずれかまたはすべては、以下で説明するように、本明細書全体にわたって説明する他の代替実施形態と組み合わせて使用され得ることに留意すべきである。

【０１５２】

具体的には、図１０に示すように、メディアストリーミング動作の前に、（ピア２２０の１つとすることもできる）サーバ２００は、ストリーミングされるメディアを符号化する（１０００）。前述のように、ＰｅｅｒＳｔｒｅａｍｅｒは、例えば、ＭＰＥＧ１／２／４、ＷＭＡ、ＷＭＶといった、いくつかの従来のコーデックのいずれかと共に動作することができる。また、符号化プロセス１０００の間に、サーバ２００は、前述のメディアヘッダも、メディア構造を含むコンパニオンファイルも生成する。

【０１５３】

前述のように、一実施形態では、メディアが符号化される（１０００）と、符号化されたメディアパケットが、いくつかの固定サイズのデータユニットに分割される（１００５）。さらに、符号化メディアと同様に、メディアヘッダおよびメディア構造も、符号化メディアパケットを分割するのに使用されたのと同じ固定サイズのいくつかのデータユニットに分割される。前述のように、この情報を固定長のデータユニットに分割すること（１００５）により、クライアントとサービングピアの両方が、メディアストリーミング動作の前にメモリブロックを事前に割り振ることができるようになり、ストリーミングプロセス中の、計算上高くつくメモリ割り振り動作が回避される。さらに、より小さいデータユニットの使用により、ストリーミング動作中のクライアントデータユニット要求を満たすための、各サービングピアにより消費される正確な帯域幅の量に対するクライアントのよりきめ細かい制御が可能になる。

【０１５４】

符号化メディア、メディアヘッダ、およびメディア構造をより小さいデータユニットに分割すること（１００５）に加えて、一実施形態では、別の符号化階層を使って、サービングピアが本来的に低信頼である典型的なＰ２Ｐ環境でさらなる冗長性が提供される。具体的には、前述のように、一実施形態では、キーベースの高速消去耐性符号化プロセス１０１０を使って、各データユニットが、さらにいくつかのデータブロックに分割される。

【０１５５】

そのような符号化１０１０の使用は、ピアの１つまたは複数が、個々のデータユニットを再構築するのに必要なデータブロックを持つと同時に、それらのピアのどれが必要なデータを含むか識別するクライアント側での要求を簡略化することを保証する。さらに、前述のように、一実施形態では、各サービングピア２２０によって使用される消去耐性符号化キーがサーバ２００によって各ピアに自動的に割り当てられる。しかしながら、別の実施形態では、各サービングピア２２０は、単に、ランダムに１つの消去耐性符号化キーを選択するだけである。その場合、これらのキーは、各ピア２２０がクライアント２１０によって最初に接続されるときにそのクライアントによって取り出される前述の可用性ベクトルと共に含まれる。ランダムなキーの実施形態では、クライアントは、所与のデータユニットについてキー競合がある場合には、１つまたは複数のピアのそのキーを無効化する。

【０１５６】

メディアが最初に符号化され（１０００）、データユニットに分割され（１００５）、おそらくさらに消去符号化される（１０１０）と、結果として生じるデータユニットまたはデータブロックが様々なサービングピア２２０に分散される（１０１５）。この分散（１０１５）は、符号化メディアの各ブロックまたはパケットが、単に、全体として、または部分的に、いくつかのピアに提供され、次いでそこで、Ｐ２Ｐネットワークに加わったクライアントによって呼び出されたときの将来のストリーミング動作のためにキャッシュされ、または格納されるという意味で、意図的なものといえる。

【０１５７】
代替としては、前述のように、クライアント２１０が特定のメディアファイルをストリーミングするたびに、回復されるメディアパケットが符号化動作１０００後のメディアパケットになる。それらをデータユニットに分割し（１００５）、おそらくさらに消去符号化する（１０１０）ことができ、クライアントは、そこにストリーミングされたコンテンツの少なくとも一部を、ローカルメモリまたは記憶装置内に維持することができる。次いで、クライアントは、将来のストリーミング動作での（前述のピアリスト３１０中の）サービングピア２２０として識別される。この実施形態の１つの利点は、個々のメディアファイルの各部分を含むピアの数が最初は小さく、そのため、サービング要求を満たすサーバ自体の側での需要が増大するが、時間が経過し、より多くのクライアントがそのメディアをストリーミングするにつれて、それらのクライアントが、その後のストリーミング要求でのピアとして働くことができるようになることである。その結果として、ストリーミングされるメディアの全部または一部の最初のキャッシュを保持するようにサービングピア２２０を明示的に選択する必要がなくなる。その結果、ストリーミングされるメディアの最初のキャッシュを受け入れることをいとわないピアを識別しようとすることに関するサーバ側での任意の要求はさらに少なくなる。

【０１５８】
どちらの場合も、メディアがサービングピア２２０に分散される（１０１５）と、クライアント２１０は、それらのサービングピアへのストリーミング要求を開始することができる。さらに、前述のように、サーバ２００は、クライアント２１０へのストリーミングのためのサービングピア２２０としても働くことができる。再度、前述の説明を考察すると、個々のメディアファイルの最初のストリーミングはより大きなサーバ２００の関与を必要とし得るが、時間が経過し、より多くのクライアント２１０がそのメディアをストリーミングする（次いで、サービングピアとして働くために利用可能になる）につれて、実際にサービングピアとして働くサーバ側での要求は低下し、または無くなりさえする。

【０１５９】
この時点で、クライアント２１０はストリーミングセッションを開始し、まず、利用可能なサービングピア２２０のリスト３１０を取り出す。前述のように、リスト３１０は、サーバ２００から直接、ピア２２０の１つから、あるいは従来のＤＨＴ法３１５を用いて可能なサービングピアを識別することによって取り出される。クライアント２１０は、ピアリスト３１０を取り出すと、次いで、各サービングピア２２０に接続し、各ピアから可用性ベクトルを取り出す（１０２５）。さらに、一実施形態では、クライアント２１０は、進行中のストリーミング動作中に、ピアリスト３１０への更新１０３０の有無を定期的にチェックする（１０３０）。そのような定期的チェック（１０３０）の１つの利点は、大規模なＰ２Ｐネットワークでは、任意の所与の時点において、複数のサービングピアがオンラインになり、オフラインになる可能性があることである。したがって、クライアント２１０が更新されたピアリスト３１０を持つようにすることは、そのクライアントが、現在そのクライアントにメディアをストリーミングしているピア２２０の損失または劣化に応答できるようにする。リスト３１０の定期的チェック（１０３０）が新規のピア２２０のリストへの追加を指示するたびに、クライアント２１０は、また、その新規のピアに接続し、その新規のピアの可用性ベクトルを取り出す（１０２５）。

【０１６０】
クライアント２１０は、各ピア２２０の可用性ベクトルを取り出す（１０２５）と、次いで、クライアントとそれらのピアの間のネットワーク接続を介してそれらのピアの１つまたは複数にその情報に対応するデータユニットを要求することによって、サービングピアの１つまたは複数からストリーミングされるメディアのメディアヘッダおよびメディア構造を取り出す（１０３５）。

【０１６１】
前述のように、メディアヘッダは、一般に、そのメディアを記述する全体的情報、例えば、メディア中のチャネル数、各チャネルの属性および特性（オーディオサンプリング速

度、ビデオ解像度／フレーム速度）、使用されるコーデック、メディアの著者／著作権保持者などを含む。したがって、メディアストリーミングセッションの開始時にメディアヘッダを取り出せば、クライアント２２０は、その後に受け取るパケットを復号化し（１０７０）、レンダリングする（１０７５）ための必要なツールを、そのストリーミングセッション中にそれらのパケットを受け取る前にセットアップし、または初期設定する（１０４０）ことができる。

【０１６２】

さらに、特定のストリーミングメディアのメディア構造を取り出した（１０３５）後、クライアントは、そのメディア構造を分析し、そのストリーミングプロセス中に要求される必要のあるストリーミングメディアのデータユニットのデータユニットＩＤを計算する（１０４５）。次いで、クライアント２１０は、サービングピア２２０の１つまたは複数に、それらのデータユニットを１つずつ要求する（１０５０）。

【０１６３】

さらに、前述のように、消去符号化が符号化キーのランダムピア選択と組み合わせて使用される実施形態では、クライアント２１０は、キー競合をうまく処理する（１０５５）ために、ピア２２０の１つまたは複数で重複するキーを無効化する。関連する一実施形態では、ＰｅｅｒＳｔｒｅａｍｅｒは埋め込み符号化メディアを使用し、その場合、各ピア２２０ごとのデータ要求（およびストリーミングビット速度）は、利用可能なサービング帯域幅およびクライアント２１０キュー状況に従って処理される（１０６０）。この場合、進行中のデータユニットを求める要求（１０５０）は、様々なサービングピアの利用可能な帯域幅に基づいて最小の速度ひずみを提供するパケットに対応する。どちらの場合も、前述のように、欠けている、または遅れているデータユニットは、埋め込み符号化が使用されているか、それとも非埋め込み符号化が使用されているか、各ピアの接続状況、および欠けている、または遅れているデータユニットを要求し、受け取るための残り時間に応じて、同じ、または代替のピア２２０に再度要求される（１０５０）。

【０１６４】

最後に、クライアント２２０要求（１０５０）に従って特定のメディアパケットを構成するデータユニットのすべてが取り出されると、それらのデータパケットは、再組み立てされて元のメディアパケットになる（１０６５）。次いで、再組み立てされたメディアパケットは、復号化され（１０７０）、レンダリングされ（１０７５）、再生のために、従来の表示装置３５５またはスピーカ２６０、あるいはその両方に提供される。

【０１６５】

以上のＰｅｅｒＳｔｒｅａｍｅｒの説明は、図示および説明のために提示したものである。この説明では、網羅的であることも、本発明を開示の通りの形に限定することも意図されていない。前述の教示を考慮すれば多くの変更および変形が可能である。さらに、前述の代替実施形態のいずれかまたはすべてを、所望の任意の組合せで使って、別のＰｅｅｒＳｔｒｅａｍｅｒの混成実施形態を形成することもできることに留意すべきである。本発明の範囲は、この詳細な説明によってではなく、添付の特許請求の範囲によって限定されることが意図されている。

【図面の簡単な説明】

【０１６６】

【図１】本明細書で説明する、「ＰｅｅｒＳｔｒｅａｍｅｒ」を実施する例示的システムを構成する汎用コンピューティングデバイスを示す全体システム図である。

【図２】本明細書で説明する、受信側主導のメディアストリーミングのための例示的ピアツーピア（Ｐ２Ｐ）ネットワークを示す図である。

【図３】本明細書で説明する、ＰｅｅｒＳｔｒｅａｍｅｒを実施するプログラムモジュールを示す例示的アーキテクチャの流れ図である。

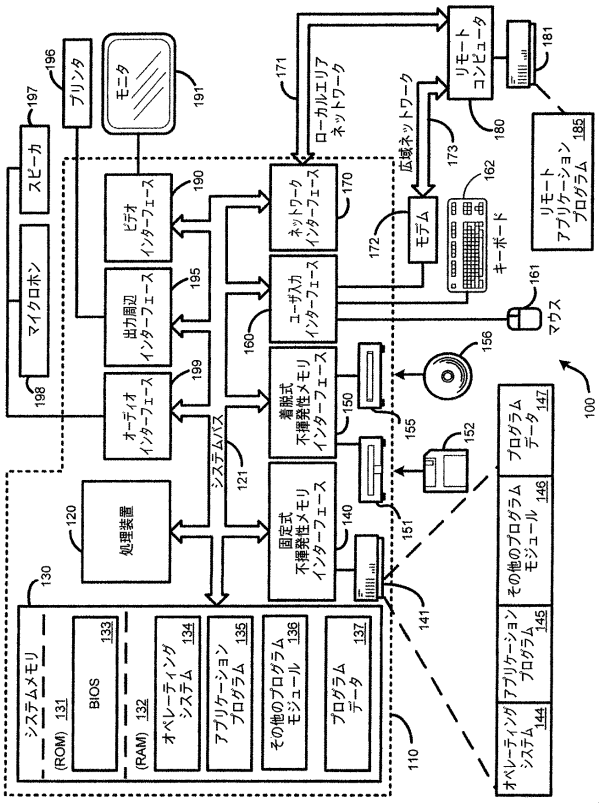
【図４】本明細書で説明する、ストリーミングメディアファイルのファイル形式を示す図である。

【図５】本明細書で説明する、ＰｅｅｒＳｔｒｅａｍｅｒによって検査済みの実施形態で

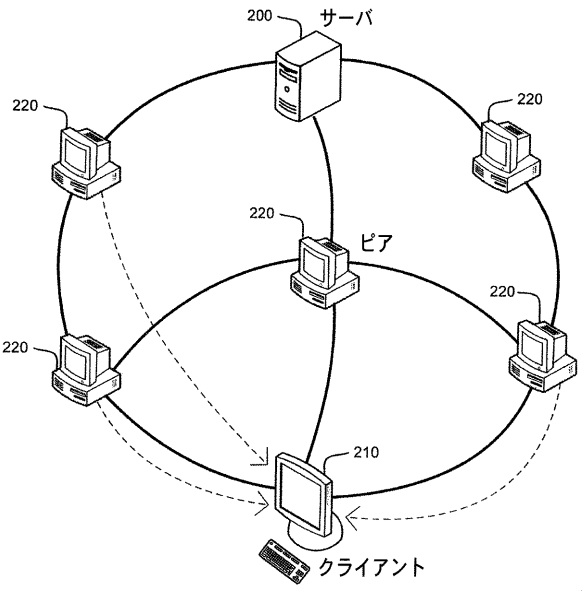
使用された「データユニット」を示す図である。
【図６】本明細書で説明する、８個のデータユニットに分割されている埋め込み符号化メディアパケットの部分キャッシングを示す図である。
【図７】クライアントＰｅｅｒＳｔｒｅａｍｅｒメディアストリーミングセッションのＤｉｒｅｃｔＳｈｏｗ（商標）フィルタグラフの一例を示す図である。
【図８】本明細書で説明するＰｅｅｒＳｔｒｅａｍｅｒの要求およびステージングキュー、ならびにストリーミングメディアの復号化、レンダリングおよび再生を表す、システムバッファが破線で示されたアーキテクチャシステム図である。
【図９】到着データユニットでのＰｅｅｒＳｔｒｅａｍｅｒクライアントステージングキュー、および各サービングピアごとのＰｅｅｒＳｔｒｅａｍｅｒクライアント要求キューを示すブロック図である。
【図１０】本明細書で説明する、ＰｅｅｒＳｔｒｅａｍｅｒの一実施形態の一般的動作を示す動作流れ図である。
【符号の説明】
【０１６７】
１２０　処理装置
１３０　システムメモリ
１３４　オペレーティングシステム
１４４　オペレーティングシステム
１４５　アプリケーションプログラム
１３６　その他のプログラムモジュール
１４６　その他のプログラムモジュール
１３７　プログラムデータ
１４７　プログラムデータ
１４０　固定式不揮発性メモリインターフェース
１５０　着脱式不揮発性メモリインターフェース
１６０　ユーザ入力インターフェース
１６１　マウス
１６２　キーボード
１７０　ネットワークインターフェース
１７１　ローカルエリアネットワーク
１７２　モデム
１７３　広域ネットワーク
１８０　リモートコンピュータ
１８５　リモートアプリケーションプログラム
１９０　ビデオインターフェース
１９１　モニタ
１９５　出力周辺インターフェース
１９６　プリンタ
１９７　スピーカ
１９８　マイクロホン
１９９　オーディオインターフェース
２００　サーバ
２１０　クライアント
２２０　サービングピア
３００　メディア符号化モジュール
３０５　ピア位置検出モジュール
３１０　ピアリスト
３１５　分散ハッシュ表
３２０　可用性ベクトル取り出しモジュール

３２５　メディアヘッダ／メディア構造分析モジュール
３３０　クライアント構成モジュール
３３５　データユニットＩＤ計算モジュール
３４０　データユニット要求モジュール
３４５　データユニット処理モジュール
３５０　復号化／レンダリング／再生モジュール
８０５　ＴＣＰ送信バッファ（サーバ）
８１０　ＴＣＰ受信バッファ（サーバ）
８１５　ステージングキュー（クライアント）
８２０　ＴＣＰ送信バッファ（クライアント）
８３０　圧縮オーディオ
８４０　圧縮ビデオ
８４５　オーディオデコーダ
８５０　ビデオデコーダ
８５５　圧縮されていないオーディオ
８６０　圧縮されていないビデオ
８６５　オーディオレンダリング
８７０　オーディオレンダリング

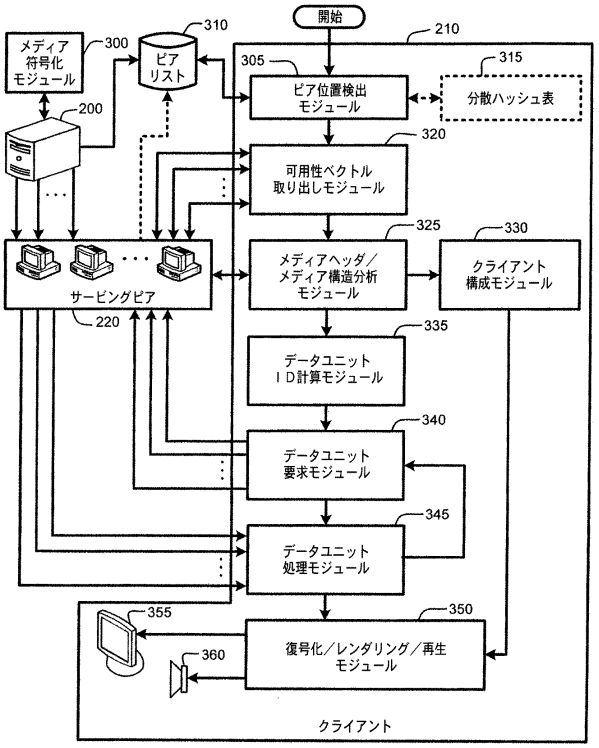
【図１】



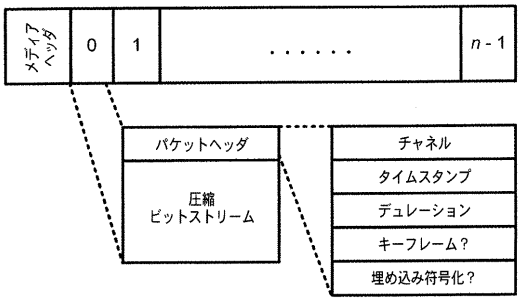
【図２】



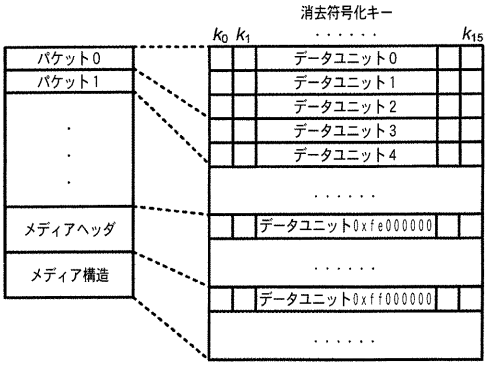
【図３】



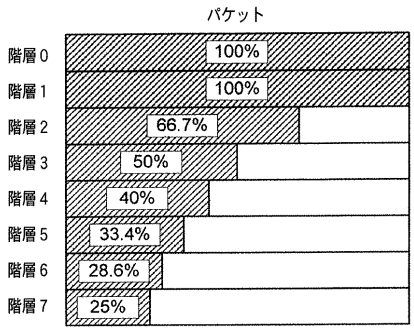
【図４】



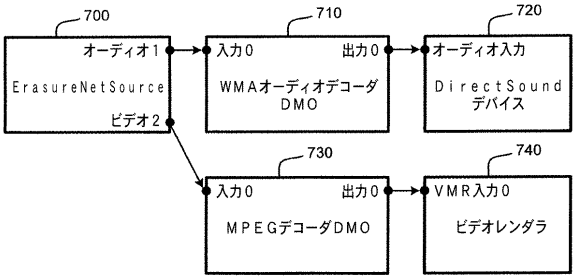
【図５】



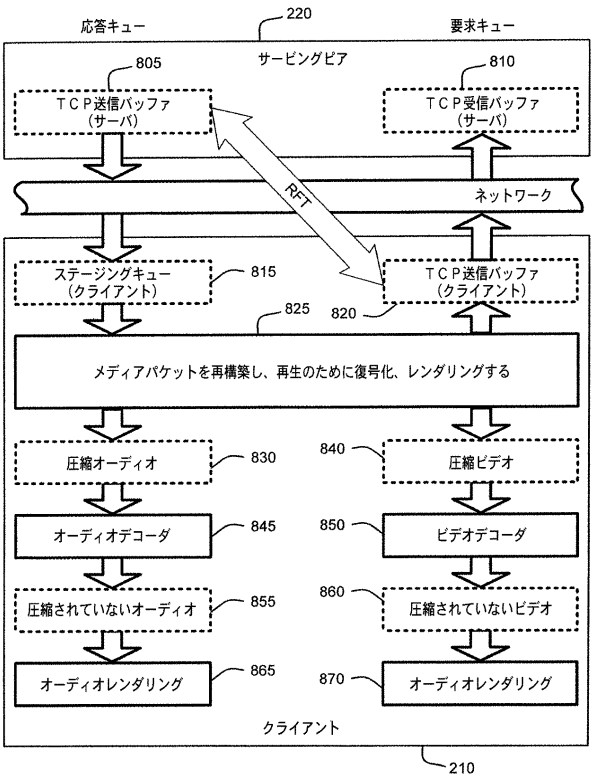
【図６】



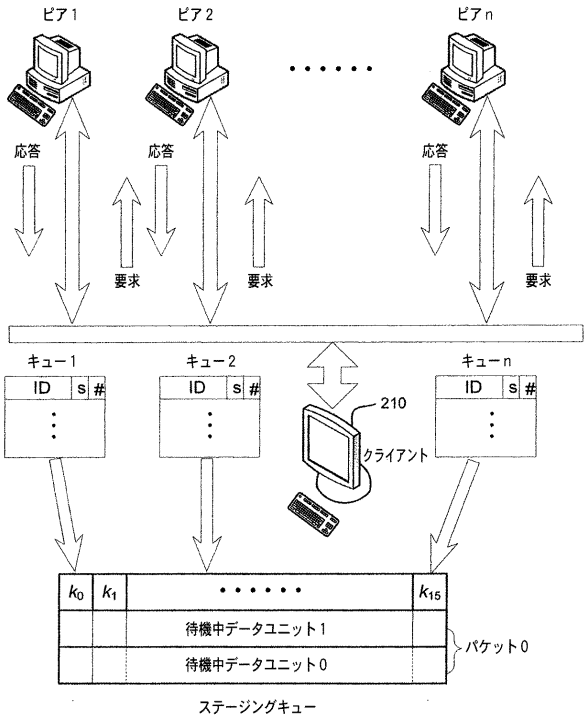
【図７】



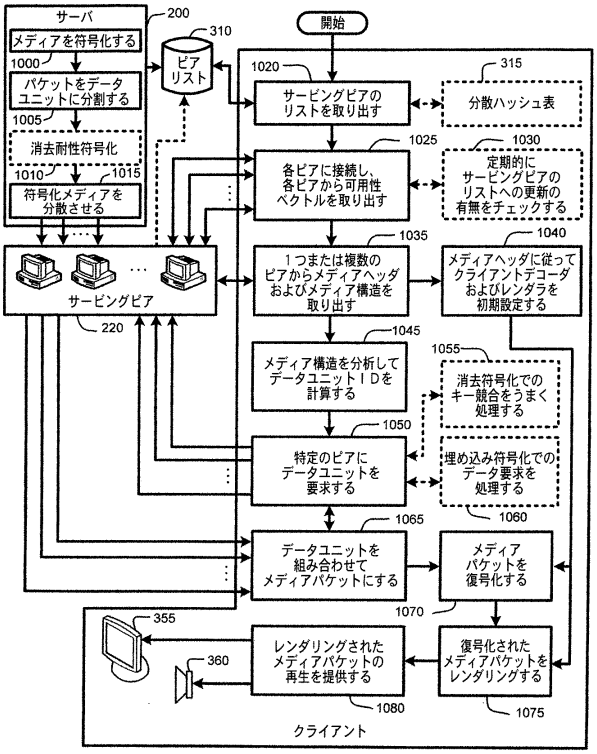
【図８】



【図９】


ピア1
ピア2
ピアn
応答
要求
キュー1
キュー2
キューn
ID
s
#
210
クライアント
k_0
k_1
k_{15}
待機中データユニット1
待機中データユニット0
パケット0
ステージングキュー

【図１０】


サーバ
200
メディアを符号化する
1000
パケットをデータユニットに分割する
1005
消去耐性符号化
1010
1015
符号化メディアを分散させる
310
ピアリスト
サービングピア
220
開始
1020
サービングピアのリストを取り出す
315
分散ハッシュ表
1025
各ピアに接続し、各ピアから可用性ベクトルを取り出す
1030
定期的にサービングピアのリストへの更新の有無をチェックする
1035
1つまたは複数のピアからメディアヘッダおよびメディア構造を取り出す
1040
メディアヘッダに従ってクライアントデコーダおよびレンダラを初期設定する
1045
メディア構造を分析してデータユニットIDを計算する
1055
消去符号化でのキー競合をうまく処理する
1050
特定のピアにデータユニットを要求する
埋め込み符号化でのデータ要求を処理する
1060
1065
データユニットを組み合わせてメディアパケットにする
メディアパケットを復号化する
1070
355
360
レンダリングされたメディアパケットの再生を提供する
復号化されたメディアパケットをレンダリングする
1080
1075
クライアント

フロントページの続き

審査官　古河　雅輝

(56)参考文献　特開２００４－０８０１４５（ＪＰ，Ａ）
国際公開第０３／１０５４２１（WO，Ａ１）
特開２００３－１６９０８９（ＪＰ，Ａ）
特開２００４－２３６３１６（ＪＰ，Ａ）
特開２００１－２３０８０９（ＪＰ，Ａ）
特開２００２－２０４２７８（ＪＰ，Ａ）
特開２００３－３０９６００（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１３／００
Ｇ０６Ｆ　１５／００
Ｈ０４Ｌ　１２／００－１２／２６
Ｈ０４Ｌ　１２／５０－１２／６６
Ｈ０４Ｎ　７／１７３