

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2012-105285

(P2012-105285A)

(43) 公開日 平成24年5月31日(2012.5.31)

(51) Int.Cl. F I テーマコード (参考)
H04N 7/173 (2011.01) H04N 7/173 610Z 5C164
H04N 7/173 630

審査請求 未請求 請求項の数 14 O L 外国語出願 (全 327 頁)

(21) 出願番号 特願2011-259530 (P2011-259530)
(22) 出願日 平成23年11月9日 (2011.11.9)
(31) 優先権主張番号 12/927, 202
(32) 優先日 平成22年11月9日 (2010.11.9)
(33) 優先権主張国 米国 (US)

(特許庁注：以下のものは登録商標)

1. ウィンドウズ

(71) 出願人 510156402
オンライン インコーポレイテッド
アメリカ合衆国 カリフォルニア州 94
301 パロ アルト リットン アベニ
ュー 181
(74) 代理人 100092093
弁理士 辻居 幸一
(74) 代理人 100082005
弁理士 熊倉 禎男
(74) 代理人 100067013
弁理士 大塚 文昭
(74) 代理人 100086771
弁理士 西島 孝喜
(74) 代理人 100109070
弁理士 須田 洋之

最終頁に続く

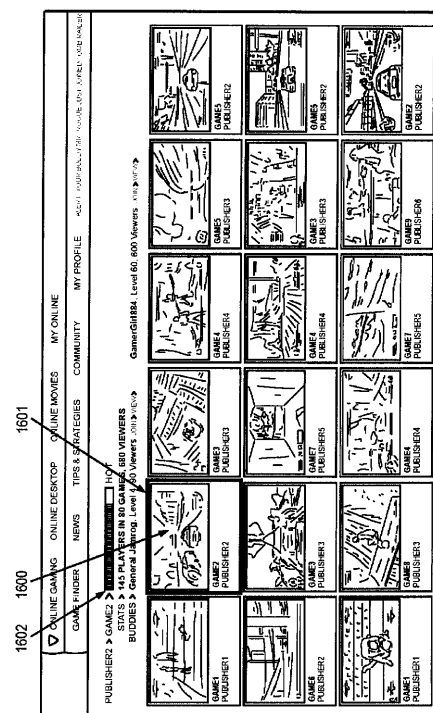
(54) 【発明の名称】 リモートホスト型ビデオ効果のためのシステム及び方法

(57) 【要約】

【課題】 リモートホスト型ビデオ効果のためのシステム及び方法を提供する。

【解決手段】 リモートホスト型ビデオ効果のための方法は、ストリーミング双方向ビデオであるホスティングサービスのアプリケーションフレームバッファをレンダリングすることを含む。アプリケーションフレームバッファの一部のみをその後にローカル装置上に表示する。

【選択図】 図16



【特許請求の範囲】**【請求項 1】**

ストリーミング双方向ビデオであるホスティングサービスのアプリケーションフレームバッファをレンダリングする段階と、

前記アプリケーションフレームバッファの一部分のみをローカル装置上に表示する段階と、
を備えた方法。

【請求項 2】

前記ホスティングサービスは、前記ローカル装置上に表示される一部分のみをストリーミングする、請求項 1 に記載の方法。

10

【請求項 3】

前記ホスティングサービスは、前記アプリケーションフレームバッファ全体をストリーミングし、前記ローカル装置は、前記アプリケーションフレームバッファを切り落とし、前記ローカル装置のビデオウインドウに前記一部分が表示されるようにする、請求項 1 に記載の方法。

【請求項 4】

前記ローカル装置は、前記ビデオウインドウの切り落としに関する第 1 のコントロール入力を前記ホスティングサービスへ送信する、請求項 3 に記載の方法。

【請求項 5】

前記ローカル装置は、前記ビデオウインドウの位置に関する第 2 のコントロール入力を前記ホスティングサービスへ送信する、請求項 4 に記載の方法。

20

【請求項 6】

前記ローカル装置は、前記ビデオウインドウのスケーリングに関する第 3 のコントロール入力を前記ホスティングサービスへ送信する、請求項 5 に記載の方法。

【請求項 7】

前記第 1、第 2 及び第 3 のコントロール入力は、タッチスクリーンユーザインターフェイスのコマンドに応答して発生される、請求項 6 に記載の方法。

【請求項 8】

前記コマンドは、ドラッグコマンドを含む、請求項 7 に記載の方法。

【請求項 9】

前記コマンドは、スワイプコマンドを含む、請求項 7 に記載の方法。

30

【請求項 10】

前記コマンドは、ピンチコマンドを含む、請求項 7 に記載の方法。

【請求項 11】

前記コマンドは、広げコマンドを含む、請求項 7 に記載の方法。

【請求項 12】

前記アプリケーションは、ウェブブラウザを含む、請求項 1 に記載の方法。

【請求項 13】

前記ホスティングサービス及びローカル装置は、インターネットに接続され、前記ホスティングサービスは、前記ローカル装置に比して、インターネットへのより広い帯域巾の接続を有する、請求項 1 に記載の方法。

40

【請求項 14】

前記ホスティングサービス及びローカル装置は、インターネットに接続され、前記ホスティングサービスは、前記ローカル装置に比して、インターネットへのより短い待ち時間の接続を有する、請求項 1 に記載の方法。

【発明の詳細な説明】**【技術分野】****【0001】**

本発明は、一般的に、オーディオ及びビデオメディアを操作し及びそれにアクセスするユーザの能力を改善するデータ処理システムの分野に係る。

50

【0002】

関連出願：本出願は、2009年11月10日に出願された米国プロビジョナル特許出願第61/259,927号の利益を主張するものである。本出願は、2007年12月5日に出願された米国特許出願第11/999,461号の一部継続(CIP)出願であり、更に、これは、2002年12月10日に出願された特許出願第10/315,460のCIP出願であり、その各々は、本CIP出願の譲受人に譲渡されたものである。

【背景技術】

【0003】

記録音声及び動画メディアは、トーマスエジソンの時代から世相を表している。20世紀の初頭に、記録音声メディア(シリンダー及びレコード)及び動画メディア(ニコロデオン及び映画)が広く配給されたが、両技術は、まだ幼時期であった。1920年代後期に、動画が大量市場ベースで音声と結合され、その後、音声を伴うカラー動画となった。ラジオ放送は、大規模な広告支援形態の放送大量市場音声メディアへと徐々に進化した。1940年代中期にテレビ(TV)放送規格が確立されると、テレビは、予め録画された又は生の動画を家庭へ届ける放送大量市場メディアの一形式としてラジオに加わった。

【0004】

20世紀中期までは、米国の家庭の大部分は、記録音声メディアを再生する蓄音器レコードプレーヤ、生放送音声を受信するラジオ、及び生放送オーディオ/ビデオ(A/V)メディアを上映するテレビ受像機を有していた。多くの場合に、これら3つの「メディアプレーヤ」(レコードプレーヤ、ラジオ及びTV)は、共通のスピーカを共有する1つのキャビネットに結合されて、家庭の「メディアセンター」となった。メディアの選択肢は、消費者に制限されるが、メディア「エコシステム」は、かなり安定したものであった。ほとんどの消費者は、「メディアプレーヤ」をどのように使用するか知っており、それらの能力をフルに楽しむことができた。同時に、メディアの発行者(主に動画及びテレビスタジオ、並びに音楽会社)は、一般に普及した海賊版又は「二次販売」即ち中古メディアの再販から悩まされることなく、それらのメディアを劇場及び家庭の両方に配給することができた。典型的に、発行者は、二次販売から収入を得ず、従って、そうでなければ新規販売のために中古メディアの買い手から発行者が得るかもしれない収入が減少した。20世紀の中期に販売された中古レコードは確かにあるが、そのような販売は、レコード発行者に大きな影響を及ぼさなかった。というのは、典型的に大人が一度又は数回見るだけの動画又はビデオ番組とは異なり、音楽トラックは、数百回又は数千回も聞かれるからである。従って、音楽メディアは、動画/ビデオメディアより遥かに「腐敗」しない(即ち、大人の消費者にとって価値が継続する)。レコードを購入すると、消費者がその音楽が好きな場合、消費者は、おそらくそれを長期間保持する。

【0005】

20世紀の中期から今日にかけて、メディアエコシステムは、消費者及び発行者の利益及び損害の両方にとって一連の根本的な変化を受けた。オーディオレコーダー、特に、高品質ステレオサウンド付きのカセットテープが広く導入されると、確かに高度の消費者便宜性があった。しかし、それは、消費者メディアと共に今や広く普及したもの、即ち海賊版の始まりも印した。確かに、大勢の消費者が彼ら自身のレコードを純粹に便宜性のためにテープ記録するのにカセットテープを使用した。次第に多くの消費者(例えば、互いのレコードコレクションに手早くアクセスできる寄宿舎の学生)が海賊版コピーを作成した。又、消費者は、発行者からレコードやテープを買うのではなく、ラジオを経て演奏される音楽もテープ記録した。

【0006】

消費者向けVCRの出現は、新規なVCRがTVショーを録画するようにセットでき、後で見ることができるので、更なる消費者便宜性を導くと共に、映画やTV番組に「オンデマンド」ベースでアクセスできるビデオレンタルビジネスの創設も導いた。1980年代中期以来の大量市場家庭用メディア装置の急速な開発は、かつてない選択肢レベル及び消費者にとっての便宜性を導くと共に、メディア発行市場の急速な膨張も導いた。

【0007】

今日、消費者は、過度のメディア選択肢及び過度のメディア装置に直面し、その多くは、特定の形式のメディア又は特定の発行者に結び付いたものである。熱心なメディア消費者は、家の種々の部屋にTV及びコンピュータに接続された装置を積み重ねて有し、その結果、1つ以上のTV受像機及び／又はパーソナルコンピュータ（PC）へのケーブルの「ネズミの巣穴」、並びにリモートコントローラのグループが生じている。（本出願に関して、「パーソナルコンピュータ」又は「PC」という語は、デスクトップ、Macintosh（登録商標）又は他の非Windows（登録商標）コンピュータ、Windows適合装置、Unix（登録商標）バリエーション、ラップトップ、等を含めて、家庭又はオフィスで使用するのに適した任意の種類のコンピュータを指す。）これらの装置は、ビデオゲームコンソール、VCR、DVDプレーヤ、オーディオサラウンドサウンドプロセッサ／増幅器、衛星セットトップボックス、ケーブルTVセットトップボックス、等を含む。そして、熱心な消費者にとって、互換性の問題があるために、複数の類似機能装置が存在する。例えば、消費者は、HD-DVD及びBlu-ray DVDプレーヤの両方を所有したり、又はMicrosoft Xbox（登録商標）及びSony Playstation（登録商標）の両ビデオゲームシステムを所有したりすることがある。実際に、あるゲームはゲームコンソールのバージョンにわたって互換性がないために、消費者は、Xboxと、その後のバージョン、例えば、Xbox 360（登録商標）との両方を所有することがある。多くの場合、消費者は、どのビデオ入力及びどのリモートコントローラを使用すべきか混乱する。ディスクが正しいプレーヤ（例えば、DVD、HD-DVD、Blu-ray、Xbox又はPlaystation）に入れられ、その装置に対してビデオ及びオーディオ入力を選択され、そして正しいリモートコントローラが見つかった後でも、消費者は、まだ、技術的な挑戦に直面する。例えば、ワイドスクリーンDVDのケースでは、ユーザは、先ず、自分のTV又はモニタスクリーンにおける正しいアスペクト比（例えば、4：3、フル、ズーム、ワイドズーム、シネマワイド、等）を決定し、次いで、セットすることが必要になる。同様に、ユーザは、先ず、正しいオーディオサラウンドサウンドシステムフォーマット（例えば、AC-3、ドルビーデジタル、DTS、等）を決定し、次いで、セットすることが必要になる。多くの場合、消費者は、彼等のテレビ又はオーディオシステムのフル能力でメディアコンテンツを楽しんでいないことに気付かない（例えば、間違ったアスペクト比で押しつぶされた映画を見たり、又はサラウンドサウンドではなくステレオでオーディオを聞いたりしている）。

【0008】

装置の積み重ね体には、インターネットベースのメディア装置が益々追加されている。Sonos（登録商標）デジタルミュージックシステムのようなオーディオ装置は、インターネットからオーディオを直接ストリーミングしている。同様に、Slingbox（登録商標）エンターテインメントプレーヤのような装置は、ビデオを録画し、そしてホームネットワーク又はインターネットを通してそれをストリーミングし、遠隔位置でPCにおいてそれを見ることができる。インターネットプロトコルテレビジョン（IPTV）サービスは、デジタルサブスクライバライン（DSL）又は他のホームインターネット接続を通してケーブルTV型のサービスを提供している。又、最近では、Moxi（登録商標）メディアセンターや、Windows XPメディアセンターエディションを実行するPCのような単一の装置へ複数のメディアファンクションを一体化する努力もなされている。これらの装置は、各々、それが遂行するファンクションに対する便宜性の要素を与えるが、偏在性及びほとんどのメディアへの簡単なアクセス性が欠如している。更に、このような装置は、多くの場合に高価な処理及び／又はローカル記憶が必要なために、製造コストがしばしば数百ドルに達する。更に、これらの近代的な消費者向け電子装置は、典型的に、アイドル状態である間も、多量の電力を消費し、これは、時間と共に経費高となり且つエネルギーリソースを浪費することを意味する。例えば、ある装置は、消費者がそれをターンオフするのを怠るか又は異なるビデオ入力へスイッチした場合に動作し続けることがある。そして、どの装置も完全な解決策ではないので、家庭内の装置の他の積み重ね

体と一体化しなければならず、これでは、依然、ユーザをワイヤのネズミの巣穴及びリモートコントローラの海に残すことになる。

【0009】

更に、多くの新しいインターネットベースの装置は、それが適切に機能するときには、典型的に、メディアを、それが入手できる以上に一般的な形態で提供する。例えば、インターネットを通してビデオをストリーミングする装置は、多くの場合、ビデオ、ゲームの「制作」又は演出家の解説のようなDVDをしばしば付随する双方向「エキストラ」ではなく、ビデオ資料だけをストリーミングする。これは、多くの場合、双方向性をローカルに取り扱う特定の装置に意図された特定のフォーマットで双方向資料が発生されることによるものである。例えば、DVD、HD-DVD及びBlu-rayディスクは、それら自身の特定の双方向フォーマットを有している。普及型フォーマットを全てサポートするように開発されるホームメディア装置又はローカルコンピュータは、おそらく消費者が動作するには法外なほど高価で且つ複雑なものにするレベルの精巧さ及び融通性を要求することになる。

10

【0010】

この問題に加えて、将来的に新規なフォーマットが導入された場合には、ローカル装置は、その新規なフォーマットをサポートするハードウェア能力をもたないことがあり、これは、消費者がアップグレード型のローカルメディア装置を購入しなければならないことを意味する。例えば、高解像度ビデオ又はステレオビデオ（例えば、各目に対して1つのビデオストリーム）が後日に導入された場合には、ローカル装置は、そのビデオをデコードする計算能力をもたないか、又は新たなフォーマットでビデオを出力するためのハードウェアをもたないことがある（例えば、各目に60fpsが与えられて、120fpsのビデオがシャッター付き眼鏡と同期されることでステレオ感が達成されると仮定すれば、消費者のビデオハードウェアが60fpsビデオしかサポートできない場合、このオプションは、アップグレード型ハードウェアを購入しなければ、利用できない）。

20

【0011】

メディア装置の老朽化及び複雑さの問題は、精巧な双方向メディア、特に、ビデオゲームに至ったときには、重大な問題となる。

【0012】

近代的なビデオゲームアプリケーションは、主として、4つの主要な非ポータブルハードウェアプラットフォーム：即ち、Sony Playstation（登録商標）1、2及び3（PS1、PS2及びPS3）、Microsoft Xbox（登録商標）及びXbox 360（登録商標）、Nintendo Gamecube（登録商標）及びWii™、並びにPCベースのゲーム、に分割される。これらプラットフォームの各々は、他のものとは異なり、1つのプラットフォームで実行するように書かれたゲームは、別のプラットフォームでは実行されない。又、装置のある世代から次の世代への互換性の問題もある。ソフトウェアゲーム開発者の大半が、特定のプラットフォームとは独立して設計されたソフトウェアゲームを創作しても、特定のゲームを特定のプラットフォームで実行するためには、特定のプラットフォームで使用するようゲームを適応させるのに、ソフトウェアの専有レイヤ（しばしば「ゲーム開発エンジン」と称される）が必要になる。各プラットフォームは、消費者へ「コンソール」（即ち、TV又はモニタ／スピーカに取り付けられるスタンドアロンボックス）として販売されるか、或いはPCそれ自体である。典型的に、ビデオゲームは、精巧なリアルタイムソフトウェアアプリケーションとして実施されるビデオゲームを含むBlu-ray DVD、DVD-ROM、又はCD-ROMのような光学的メディアで販売される。家庭用ブロードバンドの速度が高まるにつれて、ビデオゲームは、ダウンロードに利用されることが益々増えている。

30

40

【0013】

ビデオゲームソフトウェアとのプラットフォーム互換性を得るための特定の要件は、進歩型ビデオゲームのリアルタイム性及び高計算要件のために極めて厳しいものである。例えば、あるPCから、より速い処理ユニット又はコアをもつ別のPCへの製造アプリケーション

50

ョン（例えば、マイクロソフトワード）の一般的な互換性と同様に、ビデオゲームのある世代から次の世代への（例えば、X B o xからX B o x 3 6 0への、又はP l a y s t a t i o n 2（P S 2）からP l a y s t a t i o n 3（P S 3）への）完全なゲーム互換性が期待される。しかしながら、ビデオゲームでは、このようにならない。ビデオゲームの製造者は、典型的に、あるビデオゲーム世代が発売になるときに所与の価格点に対して考えられる最高の性能を求めているので、前世代システムに対して書かれた多くのゲームが後世代システムでは機能しないような急激なアーキテクチャー変更がシステムに対してしばしばなされる。例えば、X B o xは、プロセッサのx 8 6ファミリーをベースとしたものであり、一方、X B o x 3 6 0は、P o w e r P Cファミリーをベースとしたものである。

10

【0014】

従来のアーキテクチャーをエミュレートするための技術を利用できるが、ビデオゲームがリアルタイムアプリケーションであるとすれば、エミュレーションにおいて厳密に同じ挙動を達成することはしばしば不可能である。これは、消費者、ビデオゲームコンソール製造者、及びビデオゲームソフトウェア発行者にとって損害である。消費者については、全てのゲームをプレイできるようにするために、ビデオゲームコンソールの新旧の世代をT Vに接続したままにする必要があることを意味する。コンソールの製造者については、エミュレーションに関連したコスト、及び新規コンソールの採用に手間取ることを意味する。又、発行者については、全ての潜在的な消費者に届くように、新たなゲームの複数のバージョンを発売する必要がある、即ちビデオゲームのブランド（例えば、X B o x、P l a y s t a t i o n）ごとのバージョンを発売するだけでなく、所与のブランドのバージョン（例えば、P S 2及びP S 3）ごとのバージョンをしばしば発売する必要がある。例えば、他のプラットフォームの中でも、X B o x、X B o x 3 6 0、P S 2、P S 3、G a m e c u b e、W i i及びP Cに対して、エレクトロニックアーツ“M a d d e n N F L 0 8”の個別バージョンが開発されている。

20

【0015】

セルラー（セル）電話及びポータブルメディアプレーヤのようなポータブル装置も、ゲーム開発者への挑戦を提示する。次第に、このような装置は、ワイヤレスデータネットワークに接続されており、ビデオゲームをダウンロードできるようになっている。しかし、市場には、広範囲な異なる表示解像度及び計算能力をもつ種々様々なセル電話及びメディア装置が出回っている。又、このような装置は、典型的に、電力消費、コスト及び重量に制約があるので、カリフォルニア州サンタクララのN V I D I Aで製造される装置、等のグラフィックプロセッシングユニット（G P U）のような進歩型グラフィック加速ハードウェアに欠けるものである。その結果、ゲームソフトウェアの開発者は、典型的に、多数の異なる形式のポータブル装置に対して所与のゲームタイトルを同時に開発する。ユーザは、自分の特定のセル電話又はポータブルメディアプレーヤに対して所与のゲームタイトルが得られないことが分かる。

30

【0016】

家庭用ゲームコンソールの場合には、ハードウェアプラットフォームの製造者は、典型的に、彼等のプラットフォームでゲームを発行する能力についてソフトウェアゲーム開発者にロイヤリティを課する。セル電話の電信会社も、典型的に、ゲームをセル電話にダウンロードするためにゲーム発行者にロイヤリティを課する。P Cゲームの場合には、ゲームを発行するために支払われるロイヤリティはないが、ゲームの開発者は、典型的に、生じ得る広範囲のP Cコンフィギュレーション及びインストール問題をサポートするための消費者サービス負担が大きいために、高いコストに直面する。又、P Cは、典型的に、ゲームソフトウェアの海賊行為に対してバリアがほとんどない。というのは、それらは、技術的知識のあるユーザによって再プログラムが容易であり、ゲームの海賊版を容易に作成して（例えば、インターネットを通して）容易に配布することができる。従って、ソフトウェアゲーム開発者については、ゲームコンソール、セル電話及びP Cにおいて発行する上でコストがかかり且つ欠点があることになる。

40

50

【0017】

コンソール及びPCソフトウェアのゲーム発行者にとって、コストは、これだけでは終わらない。小売チャンネルを通してゲームを配布するために、発行者は、利ざやを得るように小売店の販売価格より低い卸売価格を課する。又、発行者は、典型的に、製造コストを支払いそしてゲームを保持する物理的メディアを配布しなければならない。又、発行者は、例えば、ゲームが売れない場合、ゲームの価格が下がった場合、或いは小売店が卸売価格の一部又は全部を払い戻し及び／又はゲームを買い手から引き取らねばならない場合に考えられる不測の事態をカバーするために小売店により「価格保護費用」がしばしば課せられる。更に、小売店は、典型的に、広告チラシでゲームを市場に出す上で助けとなるように発行者に費用を課する。更に、小売店は、ゲームをプレイし終えたユーザからゲームを買い戻して、それを中古ゲームとして販売し、典型的に、中古ゲームの収入をゲーム発行者と分配しないことが増えている。ゲーム発行者に課せられるコスト負担に加えて、しばしばゲームの海賊版が作成されてインターネットを通して配布され、ユーザがそれをダウンロードして無料でコピーする事態になっている。

10

【0018】

インターネットのブロードバンド速度が高まりつつあり、そして米国内及び世界中で、特に、家庭や、インターネット接続PCがレンタルされるインターネット「カフェ」へのブロードバンド接続がより一般的になってきているので、ゲームは、PC又はコンソールへのダウンロードを経て益々配布されてきている。又、マルチプレーヤ及び大規模マルチプレーヤのオンラインゲームをプレイするためにブロードバンド接続が益々使用されてきている（それらは、両方とも、本開示では頭文字“MMOG”で参照される）。これらの変更は、小売配布に関連した問題及びコストをある程度軽減する。オンラインゲームをダウンロードすることは、配布コストが典型的に僅かであり且つ売れ残りメディアからのコストがほとんど又は全くないという点で、幾つかの欠点をゲーム発行者に差し向ける。しかし、ダウンロードされたゲームは、依然、海賊行為を受け、そしてそのサイズ（しばしば数ギガバイトのサイズ）のために、ダウンロードするのに非常に長い時間を要する。更に、ポータブルコンピュータ又はビデオゲームコンソールと共に販売されるような小さなディスク装置は複数のゲームで埋め尽くされる。しかしながら、ゲーム又はMMOGがプレイ可能なゲームのためのオンライン接続を要求する程度まで、海賊版の問題は、軽減されている。というのは、ユーザが、通常、有効なユーザアカウントを有することが要求されるからである。ディスプレイスクリーンを撮影するカメラ又はスピーカからの音声を録音するマイクロホンによりコピーできるリニアメディア（例えば、ビデオ及び音楽）とは異なり、各ビデオゲームの経験は独特なものであり、簡単なビデオ／オーディオ記録を使用してコピーすることができない。従って、著作権法が強力に施行されておらず、海賊行為が横行する地域でも、MMOGは、海賊行為から遮蔽され、それ故、ビジネスをサポートすることができる。例えば、Vivendi SAの“World of Warcraft” MMOGは、世界中にわたって海賊行為で悩まされずに首尾良く展開された。そして、多くのオンライン又はMMOGゲーム、例えば、Linden Labの“Second Life” MMOGは、オンラインツールを使用して資産を買い、売り、そして形成するゲームに内蔵された経済的モデルを通してゲームのオペレータのための収入を発生する。従って、従来のゲームソフトウェアの購入又は契約に加えて、メカニズムを使用して、オンラインゲームを使用するための支払をすることができる。

20

30

40

【0019】

オンラインの性質又はMMOGにより海賊行為をしばしば軽減できるが、オンラインゲームオペレータは、依然、残りの挑戦に直面している。多くのゲームは、オンライン又はMMOGが適切に機能するために実質的なローカル（即ち、家庭内）処理リソースを要求する。ユーザは、性能の低いローカルコンピュータ（例えば、ローエンドラップトップのような、GPUをもたないもの）を有する場合に、ゲームをプレイできないことがある。更に、ゲームコンソールが古くなるにつれて、最新型から徐々に後退し、より進歩したゲームを取り扱うことができなくなる。ユーザのローカルPCがゲームの計算要求を取り扱

50

いできると仮定しても、しばしばインストールの複雑さがある。ドライバの互換性がないことがある（例えば、新たなゲームがダウンロードされる場合に、グラフィックドライバの新たなバージョンがインストールされ、これは、グラフィックドライバの古いバージョンに基づく以前にインストールされたゲームを不動作にすることがある）。コンソールは、より多くのゲームがダウンロードされるにつれて、ローカルディスクスペースを使い果たす。複雑なゲームは、典型的に、バグが見つかって修理されたとき、又はゲームに変更がなされた場合に（例えば、ゲームのレベルがプレイするのに難し過ぎるか又は容易過ぎるとゲーム開発者が分かった場合に）、ゲーム開発者から時間と共にダウンロードされたパッチを受け取る。これらのパッチは、新たなダウンロードを要求する。しかし、時々、全てのユーザが全てのパッチのダウンロードを完了するのではない。他のときには、ダウンロードされたパッチが他の互換性又はディスクスペース消費の問題を導入する。

10

【0020】

又、ゲームのプレイ中に、グラフィック又は挙動情報をローカルPC又はコンソールに与えるために大きなデータダウンロードが要求されることがある。例えば、ユーザがMMOGの部屋に入り、そしてグラフィックデータで作られるか又はユーザのローカルマシンでは得られない挙動をもつシーン又はキャラクタに遭遇する場合には、そのシーン又はキャラクタのデータをダウンロードしなければならない。その結果、インターネット接続が十分に高速でない場合には、ゲームのプレイ中に実質的な遅延を招く。そして、遭遇したシーン又はキャラクタが、ローカルPC又はコンソールを越えるような記憶スペース又は計算能力を要求する場合には、ユーザがゲームを進められないか又は質の低いグラフィックで続けねばならない事態を招く。従って、オンライン又はMMOGゲームは、それらの記憶及び／又は計算の複雑さの要件をしばしば制限する。更に、それらは、ゲーム中のデータ転送の量をしばしば制限する。又、オンライン又はMMOGは、ゲームをプレイできるユーザの市場を狭めることもある。

20

【0021】

更に、技術的知識のあるユーザがゲームのローカルコピーをリバースエンジニアリングし、そしてチートできるようにゲームを変更することが次第に増えてきている。チートは、人間的に可能である以上に速くボタン押しを繰り返すのと同程度に簡単である（例えば、銃を非常に速く撃つように）。ゲーム内資産トランザクションをサポートするゲームでは、チートが、実際の経済的価値の資産を含む詐欺的トランザクションを生じるレベルの精巧さに到達する。オンライン又はMMOG経済的モデルがそのような資産トランザクションに基づくものであるときは、これは、ゲームオペレータに実質的に有害な結果をもたらす。

30

【0022】

新たなゲームを開発するコストは、PC及びコンソールが（例えば、リアルタイムレイトレーシングのようなより現実的なグラフィック、及びリアルタイム物理的シミュレーションのようなより現実的な挙動と共に）益々精巧なゲームを作成できるのにつれて増大している。ビデオゲーム産業の初期には、ビデオゲーム開発は、アプリケーションソフトウェア開発に非常に類似したプロセスであり、即ち開発コストの大部分は、ソフトウェアの開発であり、グラフィック、オーディオ、及び挙動エレメント又は「資産」の開発、例えば、広範囲な特殊効果をもつ動画に対して開発されるものではない。今日、多くの精巧なビデオゲーム開発努力は、ソフトウェア開発よりも、特殊効果に富んだ動画開発に厳密に似ている。例えば、多くのビデオゲームは、3D世界のシミュレーションを与え、そして益々ホトリアリスティックな（即ち、生きた動きの映像写真撮影と同程度に現実的に見えるコンピュータグラフィックの）キャラクタ、プロップ及び環境を発生する。ホトリアリスティックなゲーム開発の最も挑戦的な観点の1つは、生きた動きの人間の顔と見分けがつかないようなコンピュータ発生の人間の顔を創作することである。カリフォルニア州サンフランシスコのMovaにより開発されたContourTMリアリティキャプチャーのような顔捕獲技術は、演技者の顔の正確な形状を運動中に高い解像度で捕獲し追跡する。この技術は、捕獲された生きた動きの顔と実質上区別できない3Dの顔をPC又はゲーム

40

50

コンソール上にレンダリングできるようにする。「ホトリアル」な人間の顔を捕獲してレンダリングすることは、多数の観点で有用である。第1に、非常に認識し易い有名人又はスポーツプレーヤ（多くの場合に高い費用で雇われた）がビデオゲームにしばしば使用され、不完全さがユーザにとって明らかで、視覚体験(viewing experience)を混乱させ又は不快にさせることがある。又、多くの場合に、高度のホトリアリズムを達成するには高度の細部が要求され、即ち、潜在的に、顔が動くときに多角形及び／又はテクスチャがフレームごとに変化するようにして、多数の多角形及び高解像度テクスチャをレンダリングすることが要求される。

【0023】

詳細なテクスチャを伴う多数の多角形のシーンが急速に変化するときには、ゲームをサポートするPC又はゲームコンソールは、ゲームセグメントで発生される所要数のアニメーションフレームに対して十分な多角形及びテクスチャデータを記憶するに足るRAMを有していないことがある。更に、PC又はコンソールに典型的に利用できる単一の光学的ドライブ又は単一のディスクドライブは、通常、RAMより非常に低速であり、そして典型的に、多角形及びテクスチャをレンダリングする際にGPUが受け容れられる最大データレートを維持することができない。現在のゲームは、典型的に、多角形及びテクスチャの大部分をRAMにロードし、これは、所与のシーンの複雑さ及び時間中が主としてRAMの容量により制限されることを意味する。例えば、顔のアニメーションの場合に、これは、ゲームが休止して多角形及びテクスチャ（及び他のデータ）をより多くのフレームに対してロードする前に、PC又はゲームコンソールを、ホトリアルでない低解像度の顔、又は限定された数のフレームについてのみアニメ化できるホトリアルな顔、のいずれかに制限する。

【0024】

“Loading...”に類似したメッセージをPC又はコンソールが表示するときに進行バーがスクリーンを横切ってゆっくり移動するのを見ていることは、複雑なビデオゲームの今日のユーザによって固有の欠点として容認されている。次のシーンがディスク（ここで、「ディスク」とは、特に指示のない限り、不揮発性の光学又は磁気メディア、並びに非ディスクメディア、例えば、半導体「フラッシュ」メモリを指す）からロードする間の遅延は、数秒又は数分を要する。これは、時間の浪費であり、ゲームプレーヤにとってかなりのフラストレーションとなる。上述したように、多くの又は全ての遅延は、多角形、テクスチャ又は他のデータをディスクからロードする時間によるものであるが、PC又はコンソール内のプロセッサ及び／又はGPUがシーンのためのデータを準備する間にロード時間の一部分が費やされる場合もある。例えば、サッカービデオゲームは、プレーヤが、多数の選手、チーム、スタジアム及び天候条件の中から選択を行えるようにする。従って、どんな特定の組合せが選択されるかに基づいて、異なる多角形、テクスチャ及び他のデータ（集合的に「オブジェクト」）がシーンに対して要求される（例えば、異なるチームは、そのユニホームに異なるカラー及びパターンを有する）。多数の又は全ての種々の順列を列挙し、多数の又は全てのオブジェクトを前もって予め計算し、そしてそれらオブジェクトを、ゲームを記憶するのに使用されるディスクに記憶することができる。しかし、順列の数が多い場合には、全オブジェクトに対して要求される記憶の量が、ディスクに適合するには大き過ぎる（ダウンロードするには不可能である）ことがある。従って、既存のPC及びコンソールシステムは、典型的に、所与のシーンの複雑さ及びプレイ時間中の両方について制約があり、複雑なシーンに対する長いロード時間で悩まされている。

【0025】

従来のビデオゲームシステム及びアプリケーションソフトウェアシステムに伴う別の顕著な制限は、それらが、例えば、PC又はゲームコンソールへ処理のためにロードされる必要のある多角形及びテクスチャのような3Dオブジェクトの大きなデータベースを次第に使用してきていることである。上述したように、このようなデータベースは、ディスクにローカルに記憶されるときに長いロード時間を要する。しかしながら、ロード時間は、通常、データベースがリモート位置に記憶されそしてインターネットを通してアクセスさ

れる場合には更に厳しいものとなる。このような状態では、大きなデータベースをダウンロードするのに、数分、数時間、又は数日を要することがある。更に、このようなデータベースは、多くの場合に多大な費用で生成され（例えば、ゲーム、映画又は歴史ドキュメンタリーに使用するための詳細な高いマスト付きの帆船の3Dモデル）、そしてローカルエンドユーザに販売するように意図される。しかしながら、データベースは、ローカルユーザへダウンロードされると、海賊行為を受けるおそれがある。多くの場合、ユーザは、データベースを評価してそれがユーザのニーズに合うか（例えば、ユーザが特定の動きをするときにゲームキャラクタの3Dコスチュームが満足な見掛け又は外観を有するか）どうか調べるだけのためにデータベースのダウンロードを希望する。長いロード時間は、購入することを決断する前に3Dデータベースを評価するユーザにとって妨げになる。

10

【0026】

MMOGでは、特に、ユーザがカスタマイズされたキャラクタを次第に使用できるようなゲームとして、同様の問題が発生する。キャラクタを表示するためのPC又はゲームコンソールの場合に、3D幾何学（多角形、テクスチャ、等）のデータベースへのアクセス及びそのキャラクタに対する挙動（例えば、キャラクタが盾を有する場合に、その盾が槍をそらすに十分な強度であるかどうか）を得ることが必要である。典型的に、MMOGがユーザにより最初にプレイされるときには、キャラクタに対する多数のデータベースが、ゲームの初期コピーと共に予め入手でき、これは、ゲームの光学的ディスクにローカルで入手できるか又はディスクへダウンロードされる。しかし、ゲームが進行するにつれて、データベースがローカルで入手できないキャラクタ又はオブジェクトにユーザが遭遇する場合には（例えば、別のユーザがカスタマイズされたキャラクタを生成した場合には）、そのキャラクタ又はオブジェクトを表示できるまでに、そのデータベースがダウンロードされねばならない。これは、ゲームの実質的な遅延を生じさせる。

20

【0027】

ビデオゲームの精巧さ及び複雑さがあるとすれば、従来のビデオゲームコンソールでのビデオゲーム開発者及び発行者の別の挑戦は、多くの場合、数千万ドルのコストで2ないし3年を掛けてビデオゲームを開発することである。新規なビデオゲームコンソールプラットフォームがほぼ5年毎に一度の割合で導入されると仮定すれば、ゲーム開発者は、新規なプラットフォームが発売されるのと同時にビデオゲームを入手できるようにするために、新規なゲームコンソールの発売よりこれらゲーム年だけ前に開発作業をスタートする必要がある。ほぼ同じ時期に（例えば、互いに1年又は2年以内に）競合する製造者から多数のコンソールが時々発売されるが、調べなければならないものは、各コンソールの人気であり、例えば、どのコンソールがビデオゲームソフトウェアの最大売り上げを生じるかである。例えば、近年のコンソールサイクルでは、Microsoft XBox 360、Sony Play Station 3及びNintendo Wiiがほぼ同じ一般的な時間枠で導入されるようにスケジュールされている。しかし、導入の何年前に、ゲーム開発者は、どのコンソールプラットフォームが他のものより成功するか本質的に「賭け」をし、それに応じて、開発リソースを捧げなければならない。又、動画制作会社も、映画の封切りの充分前に映画の成功の見込みの推定に基づいて限られた制作リソースを配分しなければならない。ビデオゲームに要求される投資レベルの増大を仮定すれば、ゲーム制作は、動画制作と益々類似してきており、ゲーム制作会社は、特定のビデオゲームの将来の成功の推定に基づいて制作リソースを日常的に捧げる。しかし、動画会社とは異なり、この賭けは、単に制作それ自体の成功に基づくものではなく、むしろ、ゲームの実行が意図されたゲームコンソールの成功に基づいたものである。複数のコンソールに対するゲームを一度に発売することでリスクを軽減できるが、この付加的な努力がコストを増大し、ゲームの実際の発売をしばしば遅らせる。

30

40

【0028】

PCにおけるアプリケーションソフトウェア及びユーザ環境は、ユーザに対してより視覚的に訴えるだけでなく、より有効で且つ直感的なものにするために、計算上より集中的で、動的で且つ双方向のものとなる。例えば、新規なWindows VistaTMオペ

50

レーティングシステム、及びMacintosh（登録商標）オペレーティングシステムの次々のバージョンは、視覚アニメーション効果を合体している。Autodesk社からのMaya™のような進歩型グラフィックツールは、最新のCPU及びGPUの限界を押し進める非常に精巧な3Dレンダリング及びアニメーション能力を与える。しかしながら、これら新規なツールの計算上の要件は、そのような製品のユーザ及びソフトウェア開発者にとって多数の実際上の問題を生じさせる。

【0029】

オペレーティングシステム（OS）の視覚表示は、もはや販売されていないが新規なOSで依然アップグレード可能である前世代のコンピュータを含めて、広範囲な種類のコンピュータにおいて機能しなければならないので、OSグラフィック要件は、典型的にGPUを含まないコンピュータを含めたOSのターゲットであるコンピュータの最小公分母によって大きく制限される。これは、OSのグラフィック能力を甚だしく制限する。更に、バッテリー作動式のポータブルコンピュータ（例えば、ラップトップ）は、視覚表示能力を制限する。というのは、CPU又はGPUにおける高い計算上のアクティビティが、典型的に、電力消費を高め、バッテリー寿命を短くするからである。ポータブルコンピュータは、典型的に、プロセッサが利用されないときに消費電力を下げるためにプロセッサのアクティビティを自動的に下げるソフトウェアを含む。あるコンピュータモデルでは、ユーザは、プロセッサのアクティビティを手動で下げることができる。例えば、ソニーのVGNSZ280Pラップトップは、（低性能、長バッテリー寿命のために）片側に“Stamina”と表示され且つ（高性能、短バッテリー寿命のために）他側に“Speed”と表示されたスイッチを含む。ポータブルコンピュータで実行されるOSは、コンピュータがそのピーク性能能力の一部分で実行される場合でも、有用に機能することができねばならない。従って、OSグラフィック性能は、入手可能な最新の計算能力より遥かに下に留まることがしばしばある。

【0030】

Mayaのようなハイエンドの計算に強いアプリケーションは、高性能PCに使用されるという期待でしばしば販売される。これは、典型的に、非常に高い性能と、より高価で且つ低ポータブル性の最小公分母要求を確立する。その結果、このようなアプリケーションは、汎用のOS（又はMicrosoft Officeのような汎用の生産アプリケーション）より非常に限定されたターゲット視聴者を有し、典型的に、汎用のOSソフトウェア又は汎用のアプリケーションソフトウェアより相当に少ない量で販売される。又、多くの場合、見込みのあるユーザがこのような計算に強いアプリケーションを前もって試すことは困難であるから、潜在的な視聴者は更に限定される。例えば、学生がMayaの使い方を学習することを希望し、又はこのようなアプリケーションについて既に知識のある潜在的な買い手が購入（Mayaを実行できるハイエンドコンピュータを買うことも含む）に投資する前にMayaを試すことを希望すると仮定する。学生又は潜在的な買い手は、Mayaのデモバージョンをダウンロードするか又はその物理的メディアコピーを得ることはできるが、Mayaをそのフル能力（例えば、複雑な3Dシーンを取り扱う）で実行できるコンピュータがない場合には、製品の完全なインフォームドアセスメントを行うことができない。これは、このようなハイエンドアプリケーションの視聴者を制限するものである。又、これは、高い販売価格にも貢献する。というのは、開発コストは、通常、汎用アプリケーションより非常に少数の購入にわたって割賦されるからである。

【0031】

又、高価格アプリケーションは、個人や企業がアプリケーションソフトウェアの海賊版コピーを使用する大きな誘因も生み出す。その結果、ハイエンドのアプリケーションソフトウェアは、種々の技術を通して海賊行為を軽減するためにそのようなソフトウェアの発行者による顕著な努力にも関わらず、海賊版がはびこることで悩まされている。海賊版のハイエンドのアプリケーションを使用するときでも、ユーザは、海賊版コピーを実行するために高価な最新のPCに投資する必要性を排除することはできない。従って、海賊版ソフトウェアのユーザは、その実際の小売価格の一部分でソフトウェアアプリケーションを

使用できるが、そのアプリケーションをフルに利用するためには、高価なPCを購入又は入手することが依然要求される。

【0032】

高性能の海賊版ビデオゲームのユーザについても同じことが言える。海賊行為者は、ゲームをそれらの実際の価格の一部分で得ることはできるが、ゲームを適切にプレイするのに必要な高価なコンピューティングハードウェア（例えば、GPU増強型PC、又はXBox 360のようなハイエンドのビデオゲームコンソール）を購入することが依然要求される。ビデオゲームが典型的に消費者にとって気晴らしであると仮定すれば、ハイエンドのビデオゲームシステムのための付加的なコストが法外なものとなる。この事態は、現在の労働者の平均年収が米国に比してかなり低い国（例えば、中国）では、一層悪いものである。その結果、ハイエンドのビデオゲームシステム又はハイエンドのPCを所有するのは、人口の非常に僅かな割合だけとなる。このような国々では、インターネットに接続されたコンピュータを使用するための料金をユーザが支払う「インターネットカフェ」が極めて一般的となる。多くの場合に、このようなインターネットカフェは、プレーヤが計算に強いビデオゲームをプレイできるようにするGPUのような高性能特徴をもたない旧モデル又はローエンドPCしか有していない。これは、中国で非常に成功し、そのインターネットカフェで一般にプレイされるVivendiの“World of Warcraft”のような、ローエンドPCで実行されるゲームの成功における重要なファクタである。それと対照的に、“Second Life”のような計算に強いゲームは、中国のインターネットカフェにインストールされたPCでプレイできる見込みが極めて低い。このようなゲームは、インターネットカフェの低性能PCにしかアクセスできないユーザには、実質上アクセス不能である。

10

20

【0033】

又、ビデオゲームの購入を考えており、先ず、インターネットを経て自分の家にゲームのデモバージョンをダウンロードすることによりそれを試してみたいユーザにとってもバリアが存在する。ビデオゲームのデモは、多くの場合、幾つかの特徴がディスエイブルされるか又はゲームのプレイ量に限度が課せられたゲームの一人前のバージョンである。これは、PC又はコンソールのいずれかにゲームをインストールして実行できるまでにギガバイトのゲームをダウンロードする長いプロセス（おそらく数時間）を伴うことがある。PCの場合には、ゲームに対して必要とされる特殊なドライバ（例えば、DirectX又はOpenGLドライバ）を計算し、正しいバージョンをダウンロードして、それらをインストールし、次いで、PCがゲームをプレイできるかどうか決定することを含む。この後者のステップは、PCが十分な処理（CPU及びGPU）能力、十分なRAM、及び互換性のあるOS（例えば、あるゲームは、Windows XPで実行されるが、Vistaでは実行されない）を有するかどうか決定することを含む。従って、ビデオゲームデモを実行するよう試みる長いプロセスの後に、ユーザは、ユーザのPCコンフィギュレーションを仮定すれば、ビデオゲームデモがおそらくプレイできないことを充分見出すことができる。悪いことに、ユーザが、デモを試すために新規なドライバダウンロードすると、それらのドライババージョンは、ユーザがPCにおいて普通に使用している他のゲーム又はアプリケーションとの互換性がないことがあり、従って、デモをインストールすると、以前に動作できたゲーム又はアプリケーションが不動作になることがある。これらのバリアは、ユーザにフラストレーションを与えるだけでなく、ゲームを市場に出すビデオゲームソフトウェア発行者及びビデオゲーム開発者にとってもバリアとなる。

30

40

【0034】

経済的な非効率さを招く別の問題は、所与のPC又はゲームコンソールが、通常、アプリケーション及び／又はゲームに対するあるレベルの性能要件を受け容れるように設計されていることに関する。例えば、あるPCは、多少のRAMを有し、低速又は高速のCPUを有し、そしてGPUを有する場合には低速又は高速のGPUを有する。あるゲーム又はアプリケーションは、所与のPC又はコンソールの全計算パワーの利点を取り入れるが、多くのゲーム又はアプリケーションは、そうではない。ゲーム又はアプリケーションに

50

についてのユーザの選択が、ローカル P C 又はコンソールのピーク性能能力に達しない場合には、ユーザは、利用されない特徴に関して P C 又はコンソール上で金銭を浪費することになる。コンソールの場合には、コンソールの製造者は、コンソールコストを助成するために必要であった以上に支払ったことになる。

【0035】

ビデオゲームを買って楽しむ際に存在する別の問題は、ユーザがゲームの購入を決める前に他の者がそのゲームをプレイしているのをユーザが見ることができるようにすることに関する。ビデオゲームを記録して後で再生するための多数の従来の解決策が存在する。例えば、米国特許第 5, 558, 339 号は、(同じユーザ又は異なるユーザにより所有された) ビデオゲームクライアントコンピュータでの「ゲームプレイ」中にゲームコントローラのアクションを含むゲーム情報体情報を記録することを教示している。この状態情報を後で使用して、ビデオゲームクライアントコンピュータ (例えば、P C 又はコンソール) 上で幾つか又は全てのゲームアクションを再生することができる。この解決策の顕著な欠点は、ユーザが記録されたゲームを見るために、ユーザは、ゲームをプレイできるビデオゲームクライアントコンピュータを所有すると共に、記録されたゲーム状態が再生されたときにゲームプレイが同一となるようにそのコンピュータ上で実行されるビデオゲームアプリケーションを有していなければならないことである。この他に、ビデオゲームアプリケーションは、記録されたゲームと再生されるゲームとの間に実行の相違がないように書かれねばならない。

【0036】

例えば、ゲームグラフィックは、一般的に、フレームごとに計算される。多くのゲームの場合に、ゲームロジックは、時々、シーンが特に複雑であるかどうか、又は実行速度を下げる他の遅延があるかどうか (例えば、P C において、ゲームアプリケーションから C P U サイクルを取り去る別のプロセスが実行されるかどうか) に基づいて、次のフレームに対して表示されるグラフィックを計算するのに 1 フレームより短い又は長い時間を要する。このようなゲームでは、1 フレーム時間より若干短く (例えば、数 C P U クロックサイクル短く) 計算された「スレッシュホールド」フレームが最終的に発生する。その同じシーンが、全く同じゲーム状態情報を使用して再び計算されたときには、1 フレーム時間より数 C P U クロックサイクル長くかかることが容易に起きる (例えば、内部 C P U バスが外部 D R A M バスと若干位相ずれし、そしてゲーム処理から数ミリ秒の C P U 時間を取り去る別のプロセスによる大きな遅延がなくても、数 C P U サイクルの時間遅延を導入する場合には)。それ故、ゲームが再生されるときには、フレームは、1 つのフレーム時間ではなく、2 つのフレーム時間で計算される。幾つかの挙動は、ゲームがどれほど頻繁に新たなフレームを計算するか (例えば、いつゲームがゲームコントローラから入力をサンプルするか) に基づいている。ゲームが表示されている間には、異なる挙動に対する時間基準のこの相違は、ゲームのプレイに影響しないが、再生されたゲームが異なる結果を生じさせることを招く。例えば、バスケットボールの弾道は、一定の 60 f p s 速度で計算されるが、ゲームコントローラの入力が、計算されたフレームの速度に基づいてサンプリングされる場合には、計算されたフレームの速度は、ゲームが記録されたときは 53 f p s であるが、ゲームが再生されるときは 52 f p s となり、これは、バスケットボールがバスケットへ入ることが阻止されるかどうかに差を生じさせ、異なる結果を招くことになる。従って、ゲーム状態を使用してビデオゲームを記録するには、同じゲーム状態情報を使用した再生が全く同じ結果を生じさせるよう保証するために非常に入念なゲームソフトウェア設計が要求される。

【0037】

ビデオゲームを記録するための別の従来の解決策は、P C 又はビデオゲームシステムのビデオ出力を (例えば、V C R、D V D レコーダへ、又は P C 上のビデオ捕獲ボードへ) 単に記録することである。次いで、ビデオは、巻き戻して再生することもできるし、或いは記録されたビデオを、典型的に、圧縮した後にインターネットへアップロードすることもできる。この解決策の欠点は、3 D ゲームシーケンスが再生されるときに、ユーザは、

シーケンスが記録された視点のみからシーケンスを見ることに限定されることである。換言すれば、ユーザは、シーンの視点を変更することができない。

【0038】

更に、家庭用PC又はゲームコンソールで再生される記録されたゲームシーケンスの圧縮ビデオがインターネットを通して他のユーザに利用されるときには、ビデオがリアルタイムで圧縮された場合でも、その圧縮されたビデオをリアルタイムでインターネットにアップロードすることは不可能である。その理由は、インターネットに接続された世界中の多数の家庭が、極めて非対称的なブロードバンド接続を有する（例えば、DSL及びケーブルモデムは、典型的に、アップストリーム帯域巾よりも遥かに広いダウンストリーム帯域巾を有する）からである。圧縮された高解像度ビデオシーケンスは、多くの場合に、ネットワークのアップストリーム帯域巾容量より広い帯域巾を有し、それらをリアルタイムでアップロードするのを不可能にする。従って、ゲームシーケンスが再生された後、インターネットの別のユーザがゲームを見ることが出来るまでに著しい遅延が生じる（おそらく数分又は数時間）。この遅延は、ある状況（例えば、以前に生じたゲームプレーヤの成績を見る）では許容できるが、ゲーム（例えば、チャンピオンプレーヤによりプレイされるバスケットボールトーナメント）を生で見る能力、又はゲームが生でプレイされるとき

10

【0039】

別の従来の解決策は、テレビジョン制作員の管理下でのみ、視聴者が、テレビ受像機でビデオゲームを生で見ることが出来るようにする。米国及び他の国々における幾つかのチャンネルは、ビデオゲーム視聴チャンネルを提供し、テレビ視聴者は、ビデオゲームチャンネル上で何人かのビデオゲームユーザ（例えば、トーナメントでプレイする最高評価プレーヤ）を見ることが出来る。これは、テレビジョンチャンネル用のビデオ配信及び処理装置へビデオゲームシステム（PC及び／又はコンソール）のビデオ出力をフィードすることにより達成される。これは、多数のカメラがバスケットボールコートで異なるアングルから生映像を送る生のバスケットボールゲームをテレビチャンネルが放送するときとは異なる。従って、テレビチャンネルは、それらのビデオ／オーディオ処理を活用して、種々のビデオゲームシステムからの出力を操作するように装置に作用することができる。例えば、テレビチャンネルは、異なるプレーヤの状態を示すテキストをビデオゲームからのビデオの上に重畳することができ（生のバスケットボールゲーム中のオーバーレイテキストであるかのように）、そしてテレビチャンネルは、ゲーム中に生じるアクションについて論じることのできるコメンテータからの音声をかぶせて録音することができる。更に、ビデオゲーム出力を、ゲームの実際のプレーヤのビデオを記録するカメラと合成することができる（例えば、ゲームに対する感情的応答を示す）。

20

30

【0040】

この解決策に伴う1つの問題は、生放送の興奮を得るために、このような生のビデオフィードがテレビチャンネルのビデオ配信及び処理装置にリアルタイムで得られねばならないことである。しかしながら、上述したように、これは、ビデオゲームシステムが家庭から実行されるとき、特に、放送の一部分が、ゲームプレーヤの実世界ビデオを捕獲するカメラからの生ビデオを含む場合には、しばしば不可能である。更に、トーナメント状態において、上述したように、家庭内のゲーム遊技者がゲームを変更し及びチートする問題もある。これらの理由で、テレビチャンネルを経てのこのようなビデオゲーム放送は、プレーヤ及びビデオゲームシステムが共通の位置（例えば、テレビスタジオ又は競技場）に集合し、そこで、テレビ制作装置が多数のビデオゲームシステム及び潜在的な生カメラからのビデオフィードを受け容れることができるように、しばしば構成される。

40

【0041】

このような従来のビデオゲームテレビチャンネルは、ビデオゲームの世界におけるアクション及び実世界におけるアクションの両方に関して、例えば、ビデオゲームプレーヤが「アスリート」として描かれるようにして、生のスポーツイベントに類似した体験である非常に興奮させる上映をテレビの視聴者に与えることができるが、これらのビデオゲーム

50

システムは、プレーヤが互いに物理的に非常に接近した状態にしばしば限定される。そして、テレビチャンネルが放送されて以来、各放送チャンネルは、テレビチャンネルの制作員により選択された1つのビデオストリームしか示すことができない。これらの制限や、放送時間、制作装置及び制作員の高いコストのために、テレビチャンネルは、典型的に、トップトーナメントでプレイする最高評価プレーヤしか示さない。

【0042】

更に、ビデオゲームのフルスクリーン映像を全テレビ視聴者に放送する所与のテレビチャンネルは、一度に1つのビデオゲームしか示さない。これは、テレビ視聴者の選択肢を甚だしく制限する。例えば、テレビ視聴者は、所与の時間に映されるゲームに関心がないことがある。別の視聴者は、所与の時間にテレビチャンネルによって特集されない特定のプレーヤのゲームプレイを見ることにしか関心がない。他のケースでは、視聴者は、専門プレーヤがゲームにおける特定のレベルをどのように取り扱うか見ることにしか関心がない。更に他の視聴者は、制作チーム等により選択されたものとは異なる、ビデオゲームを見る視点をコントロールすることを希望する。手短に言えば、テレビ視聴者は、多数の異なるテレビチャンネルが見られる場合でも、テレビネットワークの特定の放送によって受け容れられないビデオゲームを見ることに無数の好みをもっている。上述した全ての理由で、従来のビデオゲームテレビチャンネルは、テレビ視聴者にビデオゲームを示す上で著しい制限がある。

【0043】

従来のビデオゲームシステム及びアプリケーションソフトウェアシステムの別の欠点は、それらが複雑であり、且つエラー、クラッシュ及び／又は意図されない及び望ましからぬ挙動（集合的に「バグ」）により通常悩まされていることである。ゲーム及びアプリケーションは、典型的に、発売までにデバッグ及びチューニングプロセス（しばしば「ソフトウェアクオリティアシュアランス」又はSQAと称される）に通されるが、ほぼ常に、ゲーム又はアプリケーションがその分野の広範囲な視聴者に発売されると、バグが突然現れる。不都合なことに、ソフトウェア開発者は、発売後に多くのバグを識別して追跡することは困難である。ソフトウェア開発者がバグに気付くことは困難である。バグについて学習したときでも、何がバグを生じさせたか識別するために利用できる情報は、限られた量しかない。例えば、ユーザは、ゲーム開発者の顧客サービスラインに電話し、ゲームをプレイするときに、スクリーンがフラッシュを開始し、次いで、濃い青色に変化し、PCがフリーズすることを示すメッセージを残す。これは、SQAチームに、バグを追跡するのに有用な非常に僅かな情報しか与えない。オンライン接続されたあるゲーム又はアプリケーションは、時々、幾つかのケースでは、多くの情報を与えることができる。例えば、ゲーム又はアプリケーションを「クラッシュ」について監視するために、時々「ウォッチドッグ」プロセスを使用することができる。ウォッチドッグプロセスは、ゲーム又はアプリケーションプロセスがクラッシュしたときにその状態（例えば、スタックの状態、メモリ使用状態、ゲーム又はアプリケーションがどれほど進行したか、等）に関する統計情報を収集し、次いで、その情報を、インターネットを経てSQAチームへアップロードすることができる。しかし、複雑なゲーム又はアプリケーションでは、このような情報は、クラッシュ時にユーザが何を行ったか正確に決定するために解読するのに非常に長時間を要する。従って、どんな事象シーケンスがクラッシュを招いたか決定することが不可能である。

【0044】

PC及びゲームコンソールに関連した更に別の問題は、消費者に相当不便をかけるサービスの問題を受けることである。又、これらサービスの問題は、PC又はゲームコンソールの製造者に影響を及ぼす。というのは、製造者は、壊れたPC又はコンソールを安全に運搬するために特殊なボックスを送付し、PC又はコンソールが保証期間内である場合には修理のコストを負う必要があるからである。又、ゲーム又はアプリケーションソフトウェアの発行者は、PC及び／又はコンソールが修理の状態にあることによる販売の損失（又はオンラインサービス使用）によっても影響を受ける。

【0045】

図1は、Sony Playstation（登録商標）3、Microsoft X Box 360（登録商標）、Nintendo WiiTM、Windowベースのパーソナルコンピュータ、又はApple Macintoshのような従来のビデオゲームシステムを示す。これらシステムは、各々、プログラムコードを実行するための中央処理ユニット（CPU）、典型的に、進歩型グラフィックオペレーションを遂行するためのグラフィック処理ユニット（GPU）、並びに外部装置及びユーザと通信するための複数の形態の入力／出力（I/O）を備えている。簡単化のために、これらのコンポーネントは、単一のユニット100として一緒に結合されて示されている。又、図1の従来のビデオゲームシステムは、光学的メディアドライブ104（例えば、DVD-ROMドライブ）
10
、ビデオゲームのプログラムコード及びデータを記憶するためのハードドライブ103、マルチプレーヤゲームをプレイし、ゲーム、パッチ、デモ又は他のメディアをダウンロードするためのネットワーク接続105、CPU/GPU100により現在実行されているプログラムコードを記憶するためのランダムアクセスメモリ（RAM）101、ゲームプレイ中にユーザからの入力コマンドを受け取るためのゲームコントローラ106、及びディスプレイ装置102（例えば、SDTV/HDTV又はコンピュータモニタ）を含むように示されている。

【0046】

図1に示す従来のシステムは、多数の制限で悩まされている。第1に、光学的ドライブ104及びハードドライブ103は、RAM101に比して非常に低速のアクセス速度を
20
有する傾向がある。RAM101を通して直接機能するときには、CPU/GPU100は、実際に、プログラムコード及びデータがハードドライブ103又は光学ドライブ104から直接読み取られるときに可能であるより遥かに多くの多角形を毎秒処理することができる。というのは、RAM101は、一般的に、帯域巾が非常に広く、ディスクメカニズムの比較的長いシーク遅延で悩まされないからである。しかし、これら従来システムでは、限定された量のRAMしか設けられていない（例えば、256-512Mバイト）。それ故、ビデオゲームの次のシーケンスのためのデータでRAM101が周期的に埋められる“Loading...”シーケンスがしばしば要求される。

【0047】

あるシステムは、ゲームプレイと同時にプログラムコードのローディングを重畳させるように試みているが、これは、既知の事象シーケンスがあるときしか行うことができない（例えば、車を運転して道路を下る場合に、車を運転しながら、路傍にある接近してくるビルの幾何学形状をロードすることができる）。複雑で及び／又は急速なシーンの変化については、この形式の重畳は、通常、機能しない。例えば、ユーザが戦いの最中であり、そしてその瞬間に視野内のオブジェクトを表すデータでRAM101が完全に埋められる
30
ケースでは、RAM101に現在ロードされていないオブジェクトを見るためにユーザが視野を素早く左へ動かした場合に、アクションの不連続状態が生じる。というのは、ハードドライブ103又は光学的メディア104から新たなオブジェクトをRAM101へロードするに十分な時間がないからである。

【0048】

図1のシステムでは、ハードドライブ103及び光学的メディア104の記憶容量に制限があるために別の問題が生じる。比較的大きな記憶容量（例えば、50ギガバイト以上）をもつディスク記憶装置を製造することはできるが、現在ビデオゲームで遭遇する幾つかのシーンについて十分な記憶容量をまだ与えることができない。例えば、上述したように、サッカーのビデオゲームは、世界中の多数のチーム、プレーヤ及びスタジアムの中からユーザが選択を行えるようにする。各チーム、各プレーヤ及び各スタジアムに対して、世界中の3D曲面を特徴付けるには非常に多数のテクスチャマップ及び環境マップが要求される（例えば、各チームが独特のジャージを有し、その各々が独特のテクスチャマップを要求する）。
40

【0049】

10

20

30

40

50

この後者の問題に対処するために使用される1つの技術は、ユーザによってテクスチャ及び環境マップが選択されたときにゲームがそれらを予め計算することである。これは、映像の解凍、3Dマッピング、シェーディング、データ構造の編成、等を含めて、多数の計算上強いプロセスを含むことがある。その結果、ビデオゲームがこれらの計算を遂行する間にユーザにとって遅延が生じ得る。この遅延を減少するための1つの方法は、原理的に、ゲームが最初に開発されたときに、チーム、プレーヤ名簿及びスタジアムの各順列を含めて、これら計算の全てを遂行することである。従って、ゲームの発売バージョンは、光学的メディア104、又はインターネット上の1つ以上のサーバーに記憶された全てのこの前処理されたデータを、ユーザが選択を行うときにインターネットを通してハードドライブ103へダウンロードされる所与のチーム、プレーヤ名簿、スタジアム選択のための選択された前処理されたデータと共に含む。しかしながら、実際上の問題として、ゲームプレイにおいて考えられる各順列のこのような予めロードされるデータは、多分、今日の光学的メディア装置の容量を遥かに越えるテラバイトのデータになってしまう。更に、所与のチーム、プレーヤ名簿、スタジアム選択のためのデータは、多分、数百メガバイト以上のデータになってしまう。例えば、10Mbpsの家庭用ネットワーク接続では、ネットワーク接続105を通してこのデータをダウンロードするのに、データをローカルで計算する以上の時間を要することになる。

10

【0050】

従って、図1に示す従来のゲームアーキテクチャーは、複雑なゲームの主要シーン移行間にユーザに著しい遅延を受けさせる。

20

【0051】

図1に示す従来の解決策に伴う別の問題は、年々、ビデオゲームがより進歩する傾向にあり、より高いCPU/GPUパワーを要求することである。従って、限定された量のRAMを仮定しても、ビデオゲームハードウェアの要件は、これらシステムに得られる処理パワーのピークレベルを越えてしまう。その結果、ユーザは、歩調を合わせるためにゲームハードウェアを数年毎にアップグレードする必要がある（又は新たなゲームを質の低いレベルでプレイする）。ビデオゲームが絶えず進歩する傾向の1つの結果として、家庭で使用するためのビデオゲームプレイマシンは、典型的に、それらのコストが、通常、サポートできる最高性能ゲームの要件により決定されるので、経済的に効率が悪い。例えば、XBox 360は、高性能CPU、GPU、及び数百メガバイトのRAMを必要とする“Gears of War”のようなゲームをプレイするのに使用されるか、又はXBox 360は、数キロバイトのRAM及び非常に低性能のCPUしか要求しない1970年代のゲームであるPac Manをプレイするのに使用される。実際に、XBox 360は、Pac Manゲームを一度に多数同時にホストするに十分な計算パワーを有している。

30

【0052】

ビデオゲームマシンは、典型的に、一週間のほとんどの時間中オフにされている。13年以上経った現役ゲームに関するニールセン・エンターテインメント・スタディ2006年7月号によれば、平均で、現役のゲームは、毎週14時間をコンソールビデオゲームのプレイに費やしており、即ち一週間の合計時間の12%に過ぎない。これは、平均でビデオゲームコンソールが88%の時間アイドル状態であり、高価なリソースの使用効率が悪いことを意味する。購入価格を下げるためにビデオゲームコンソールが製造者によりしばしば助成されると仮定すれば（その助成金が将来のビデオゲームソフトウェアの購入からのロイヤリティにより還元されることを期待して）、これは、特に意義深いことである。

40

【0053】

又、ビデオゲームコンソールは、ほとんどの消費者電子装置に関連したコストを負っている。例えば、システムの電子装置及びメカニズムは、エンクロージャ内に収容する必要がある。製造者は、修理保証を提供する必要がある。システムを販売する小売店は、システムの販売及び/又はビデオゲームソフトウェアの販売において利ぎやを収集する必要がある。これらのファクタは、全てビデオゲームコンソールのコストに追加され、これは、製造者によって助成されねばならないか、消費者へ回されるか、又はその両方である。

50

【0054】

更に、ビデオゲーム産業にとって海賊版が主たる問題である。実質上全ての主要ビデオゲームシステムに使用されるセキュリティメカニズムは、年々「クラック」されて、ビデオゲームの無断コピーが取られている。例えば、XBox 360のセキュリティシステムは、2006年7月にクラックされ、ユーザは、今や、不正コピーをオンラインでダウンロードすることができる。ダウンロード可能なゲーム（例えば、PC又はMac用のゲーム）は、特に海賊行為を受け易い。海賊行為の取り締まりが弱い世界のある地域では、スタンドアロンビデオゲームソフトウェアのための成功が見込める市場が本質的にない。というのは、ユーザが海賊版コピーを合法的コピーと同程度に容易にそのコストのほんの一部分で買えるからである。又、世界の多くの地域では、ゲームコンソールのコストが収入の高い割合であり、海賊版を取り締まっても、最新のゲームシステムを買う余裕がある人は僅かに過ぎない。

10

【0055】

更に、中古ゲーム市場は、ビデオゲーム産業の収入を減少させる。ユーザは、ゲームに飽きると、ゲームを店に売ることができ、店は、ゲームを他のユーザに再販売する。この無許可であるが一般的な慣習は、ゲーム発行者の収入を著しく減少させる。同様に、数年毎にプラットフォームの移行があるときには、通常、50%の販売量低下が生じる。これは、ユーザが、新規バージョンプラットフォームが発売されようとしていることを知ると、旧型プラットフォーム用のゲームの購入を止める（例えば、Playstation3が発売されようとしているときに、ユーザは、Playstation2のゲームの購入を止める）からである。新規のプラットフォームに関連した販売の損失及び開発コストの増加が相まって、ゲーム開発者の利益に非常に大きな影響が及ぶことになる。

20

【0056】

又、新規なゲームコンソールは、非常に高価である。XBox 360、Nintendo Wii、及びSony Playstation3は、全て、数百ドルで小売される。ハイパワーのパーソナルコンピュータゲームシステムは、その価格が\$8000までである。これは、特に、ハードウェアが数年後に旧式となり且つ多くのシステムが子供のために購入されることを考えると、ユーザにとって著しい投資を表す。

【0057】

以上の問題に対する1つの解決策は、ゲームプログラムコード及びデータがサーバーにホストされそしてオンデマンドでクライアントマシンへデジタルブロードバンドネットワークを経てストリーミングされる圧縮されたビデオ及びオーディオとして配信されるオンラインゲームである。フィンランドのG-cluster（現在では、日本のソフトバンクブロードメディアの子会社）のような幾つかの会社は、これらのサービスをオンラインで提供する。同様のゲームサービスが、ホテル内にあるようなローカルネットワークにおいて利用でき、DSL及びケーブルテレビジョンプロバイダーによって提供されている。これらシステムの主たる欠点は、待ち時間(latency)、即ちオペレータの「ヘッドエンド」に典型的に配置されたゲームサーバーへ信号が進み且つそこから進むのに要する時間の問題である。高速アクションビデオゲーム（「ツイッチ」ビデオゲームとしても知られている）は、ユーザがゲームコントローラでアクションを遂行する時間と、ディスプレイスクリーンが更新されてユーザアクションの結果を示す時間との間に非常に短い待ち時間を要求する。ゲームが「即座」に応答するという感覚をユーザがもつようにするため、短い待ち時間が必要とされる。ユーザは、ゲームの形式及びユーザの熟練度レベルに基づいて異なる待ち時間間隔で満足することができる。例えば、ゆっくりしたカジュアルなゲーム（バックギャモンのような）又は低速アクションの役割を演じるゲームについては100msの待ち時間が許容できるが、高速アクションゲームでは、待ち時間が70又は80msを越えると、ユーザは、ゲームにおいて演技が不十分となり、受け容れられない。例えば、高速反応時間を要求するゲームでは、待ち時間が50から100msへ増加するにつれて、精度が鋭く低下する。

30

40

【0058】

50

ゲーム又はアプリケーションサーバーが、近傍のコントロールされたネットワーク環境にインストールされるか、又はユーザへのネットワーク経路が予想可能で及び／又は帯域巾ピークを許容できる環境にインストールされるときには、最大待ち時間と、待ち時間の一貫性の両方について、待ち時間をコントロールすることが遥かに容易である（例えば、ユーザがネットワークを通してストリーミングするデジタルビデオから一定の動きを観察するように）。このようなレベルのコントロールは、ケーブルTVネットワークのヘッドエンドとケーブルTV加入者の家庭との間、又はDSL中央オフィスからDSL加入者の家庭までの間、或いはサーバー又はユーザからの商業的オフィスのローカルエリアネットワーク（LAN）環境において、達成することができる。又、保証された帯域巾及び待ち時間を有する会社間に特殊グレードのポイント対ポイントプライベート接続を得ることができる。しかし、一般的なインターネットに接続されたサーバーセンターにおいてゲームをホストし、次いで、ブロードバンド接続を経てユーザへ圧縮ビデオをストリーミングするゲーム又はアプリケーションシステムでは、多数のファクタから待ち時間を招き、従来システムの展開に甚だしい制限を生じさせる。

10

【0059】

典型的なブロードバンド接続の家庭では、ユーザは、ブロードバンドサービスのためにDSL又はケーブルモデムをもつことができる。このようなブロードバンドサービスは、通常、ユーザの家庭と一般的なインターネットとの間に25ms程度（時々はそれ以上の）の往復待ち時間を被る。更に、インターネットを経てサーバーセンターへデータをルーティングすることからも往復遅延を被る。インターネットを通しての待ち時間は、データが与えられるルート及びそれがルーティングされるときに被る遅延に基づいて変化する。ルーティング遅延に加えて、ほとんどのインターネットを相互接続する光ファイバを経て進む光の速度のために、往復遅延も被る。例えば、1000マイルごとに、光ファイバを通る光の速度及び他のオーバーヘッドのために、約22msの往復待ち時間を被る。

20

【0060】

インターネットを経てストリーミングされるデータのデータレートのために、付加的な待ち時間が生じる。例えば、ユーザが「6Mbps DSLサービス」として販売されるDSLサービスを有する場合に、実際に、ユーザは、おそらく、せいぜい5Mbps未満のダウンストリームスループットしか得ず、そしておそらく、デジタルサブスクライバラインアクセスマルチプレクサ（DSLAM）におけるピークロード時間中の混雑のような種々のファクタのために、接続の質低下を周期的に見ることになる。又、ケーブルモデムシステムネットワークにおける隣接部又は他のどこかを経てループされるローカル共有同軸ケーブルに混雑が生じる場合には、「6Mbpsケーブルモデムサービス」として販売される接続に使用されるケーブルモデムのデータレートを遥かに低く減少する同様の問題が発生する。4Mbpsの一定レートでのデータパケットがサーバーセンターからこのような接続を経てユーザデータグラムプロトコル（UDP）フォーマットで一方向としてストリーミングされる場合には、全てがうまく機能すれば、データパケットが、付加的な待ち時間を被ることなく通過するが、混雑（又は他の障害）が存在し且つユーザへデータをストリーミングするのに3.5Mbpsしか利用できない場合には、典型的な状態において、パケットがドロップされてデータロスを生じるか、又はパケットを送ることができるまで混雑点にパケットがキューイングされて付加的な待ち時間を招くか、のいずれかである。異なる混雑点は、遅延パケットを保持する異なるキューイング容量を有し、従って、あるケースでは、混雑を通過できないパケットが直ちにドロップされる。他のケースでは、数メガビットのデータがキューイングされ、最終的に、送出される。しかし、ほぼ全てのケースにおいて、混雑点におけるキューは、容量限界があり、これらの限界を越えると、キューがオーバーフローし、パケットがドロップされる。従って、付加的な待ち時間（又は更に悪い場合は、パケットのロス）を被るのを回避するために、ゲーム又はアプリケーションサーバーからユーザへのデータレート容量を越えないようにする必要がある。

30

40

【0061】

又、サーバーにおいてビデオを圧縮しそしてクライアント装置においてビデオを解凍す

50

るのに要求される時間によっても待ち時間を被る。更に、サーバー上で実行されているビデオゲームが、表示されるべき次のフレームを計算する間にも待ち時間を被る。現在入手できるビデオ圧縮アルゴリズムは、高いデータレート又は長い待ち時間のいずれかで悩まされている。例えば、モーション J P E G は、待ち時間が短いことを特徴とするイントラフレーム専用ロッシー (intra frame-only lossy) 圧縮アルゴリズムである。各ビデオフレームは、互いのビデオフレームと独立して圧縮される。クライアント装置は、圧縮されたモーション J P E G ビデオのフレームを受け取ると、そのフレームを直ちに解凍して表示し、待ち時間を非常に短くする。しかし、各フレームは、別々に圧縮されるので、アルゴリズムは、連続するフレーム間の類似性を利用することができず、その結果、イントラフレーム専用ビデオ圧縮アルゴリズムは、非常に高いデータレートで悩まされている。例え

10

ば、60 f p s (フレーム/秒) 640 x 480 モーション J P E G ビデオは、40 M b p s (メガビット/秒) 以上のデータを要求する。このように低い解像度のビデオウインドウに対するこのように高いデータレートは、多くのブロードバンドアプリケーションにおいて (及び確実にほとんどの消費者インターネットベースのアプリケーションに対して) 法外に高価なものとなる。更に、各フレームは独立して圧縮されるので、ロッシー圧縮から生じ得るフレーム内の欠陥は、おそらく、連続するフレームの異なる場所に現れる。これは、ビデオが解凍されたときに、動く視覚上の欠陥として視聴者に見える。

【0062】

マイクロソフト社からの M P E G 2、H. 264、又は V C 9 のような他の圧縮アルゴリズムは、従来の構成で使用されたときに、高い圧縮比を達成できるが、長い待ち時間が犠牲となる。このようなアルゴリズムは、インターフレーム及びイントラフレームの圧縮を使用する。周期的に、このようなアルゴリズムは、フレームのイントラフレーム専用圧縮を遂行する。このようなフレームは、キーフレーム (典型的に “I” フレームと称される) として知られている。次いで、これらのアルゴリズムは、典型的に、I フレームを手前のフレーム及び連続するフレームの両方と比較する。手前のフレーム及び連続するフレームを独立して圧縮するのではなく、このアルゴリズムは、I フレームから手前のフレーム及び連続するフレームへと映像の何が変化したか決定し、次いで、それらの変化を、I フレームに先行する変化については “B” フレームとして、及び I フレームに続く変化については “P” フレームとして記憶する。これは、イントラフレーム専用圧縮よりも非常にゆっくりしたデータレートを生じさせる。しかし、これは、典型的に、長い待ち時間を犠牲とする。I フレームは、典型的に、B 又は P フレームより非常に大きく (しばしば 10 倍以上)、その結果、所与のデータレートで送信するのに比例的に長い時間を要する。

20

30

【0063】

例えば、I フレームが、B 及び P フレームのサイズの 10 倍であり、単一の I イントラフレームごとに 29 個の B フレーム + 30 個の P フレーム = 59 個のインターフレームがあるか、又は「フレームグループ (G O P)」ごとに合計 60 個のフレームがある状態を考える。従って、60 f p s では、毎秒 1 つの 60 フレーム G O P がある。送信チャンネルが 2 M b p s の最大データレートを有すると仮定する。チャンネルに最高クオリティのビデオを得るために、圧縮アルゴリズムは、2 M b p s のデータストリームを発生し、上述した比を仮定すれば、これは、2 メガビット (M b) / (59 + 10) = 30, 394

40

ビット/イントラフレーム及び 303, 935 ビット/I フレームを生じさせる。圧縮されたビデオストリームが解凍アルゴリズムによって受信されるときには、ビデオを着実にプレイするために、各フレームを規則的な間隔 (例えば、60 f p s) で解凍して表示する必要がある。この結果を得るために、フレームが送信待ち時間を受ける場合には、全てのフレームを少なくともその待ち時間だけ遅延する必要がある。従って、最悪のケースのフレーム待ち時間は、各ビデオフレームに対する待ち時間を定義する。I フレームは、最も大きなものであるから、最長の送信待ち時間を導入し、I フレームを解凍して表示できるまでに I フレーム全体を受信しなければならない (又はインターフレームが I フレームに依存する)。チャンネルデータレートが 2 M b p s であると仮定すれば、I フレームを送信するのに、303, 935 / 2 M b = 145 m s を要することになる。

50

【0064】

送信チャンネルの帯域巾の大部分を使用する上述したインターフレームビデオ圧縮システムは、フレームの平均サイズに対してIフレームのサイズが大きいために長い待ち時間を受けることになる。又は、換言すれば、従来のインターフレーム圧縮アルゴリズムは、イントラフレーム専用圧縮アルゴリズムより低いフレーム当たり平均データレートを達成するが（例えば、2Mbps対40Mbps）、大きなIフレームのために、依然、高いフレーム当たりピークデータレートで悩まされている（例えば、303, 935*60=18.2Mbps）。前記分析は、P及びBの両フレームがIフレームより非常に小さいと仮定していることを銘記されたい。これは、一般的には、真であるが、高い映像複雑さが手前のフレーム、高モーション又はシーンの変化に相関していないフレームについては真ではない。このような状態では、P又はBフレームがIフレームと同程度の大きさになる（P又はBフレームがIフレームより大きくなると、精巧な圧縮アルゴリズムが、典型的に、Iフレームを「強制(force)」し、そしてP又はBフレームをIフレームに置き換える）。従って、デジタルビデオストリームには、いつでも、Iフレームサイズのデータレートピークが生じる。従って、圧縮されたビデオでは、平均ビデオデータレートが送信チャンネルのデータレート容量に接近するとき（ビデオに対する高いデータレート要求を仮定すると、しばしばそうである）、Iフレーム或いは大きなP又はBフレームからの高いピークデータレートが長いフレーム待ち時間を生じさせる。

10

【0065】

もちろん、以上の説明は、GOPにおける大きなB、P又はIフレームによって生じる圧縮アルゴリズムの待ち時間を特徴付けるに過ぎない。Bフレームが使用される場合は、待ち時間がより長くなる。その理由は、Bフレームを表示できるまでに、Bフレーム及びIフレーム後の全てのBフレームを受信しなければならないからである。従って、各Iフレームの前に5個のBフレームがあるBBBBBIPPPPPBBBBBIPPPPPのような画像グループ（GOP）シーケンスにおいて、最初のBフレームは、その後のBフレーム及びIフレームが受信されるまでビデオデコンプレッサによって表示することができない。従って、ビデオが60fps（即ち、16.67ms/フレーム）でストリーミングされる場合には、最初のBフレームを解凍できるまでに、チャンネル帯域巾がどれほど速くても、5個のBフレーム及びIフレームは、受信するのに16.67*6=100msを要し、これは、ちょうど、5個のBフレームである。30個のBフレームをもつ圧縮されたビデオシーケンスは、極めて一般的である。そして、2Mbpsのような低いチャンネル帯域巾では、Iフレームのサイズにより生じる待ち時間の影響が、主として、Bフレームが到着するのを待機することによる待ち時間の影響に加えられる。従って、2Mbpsチャンネルにおいて、非常に多数のBフレームがある状態では、従来のビデオ圧縮技術を使用して、500ms以上の待ち時間を越えるのは極めて容易である。Bフレームが使用されない（所与のクオリティレベルに対して低い圧縮比を犠牲にして）場合には、Bフレームの待ち時間を被らないが、上述したピークフレームサイズにより生じる待ち時間は、依然、被る。

20

30

【0066】

この問題は、まさに多数のビデオゲームの性質により激化される。上述したGOP構造を使用するビデオ圧縮アルゴリズムは、主として、受動的な視聴に意図された生のビデオ又は動画資料で、使用上最適化されている。典型的に、カメラ（実際のカメラであるか、コンピュータ発生アニメーションの場合のバーチャルカメラであるかに関わらず）、及びシーンは、比較的安定である。というのは、単に、カメラ又はシーンがあまりに急に動く場合に、ビデオ又は映画資料が、（a）典型的に見るのに不快であり、そして（b）それを見る場合に、カメラが突然グイと動くときに、通常、視聴者がアクションを厳密に追従できないからである（例えば、バースディケーキのキャンドルに息を吹きかける子供を撮影するときにカメラがパンプされ、そして突然ケーキから離れて戻るようにグイと動かされる場合に、視聴者は、典型的に、子供及びケーキに集中し、カメラが突然動くときの短い中断は無視する）。ビデオのインタビュー又はビデオ遠隔会議のケースでは、カメラを

40

50

固定位置に保持して、全く動かさず、非常に僅かなデータピークしか生じさせない。しかし、3D高アクションビデオゲームは、一定の動きにより特徴付けられる（例えば、レースの期間中に全フレームが迅速な動きにあるような3Dレースを考えるか、又はバーチャルカメラが常にグイと動くようなファーストパーソンシューター(first-person shooter)を考える）。このようなビデオゲームは、これらの突然の動きの間に何が起きたかユーザがはっきり見る必要があるところの大きなそして頻繁なピークを伴うフレームシーケンスを生じさせる。従って、3D高アクションビデオゲームでは、圧縮欠陥は、ほとんど許容できない。従って、多くのビデオゲームのビデオ出力は、それらの性質により、非常に高く且つ頻繁なピークをもつ圧縮ビデオストリームを発生する。

【0067】

高速アクションビデオゲームのユーザが長い待ち時間に対してあまり寛容度がないと仮定し、又、上述した全ての待ち時間の原因を仮定すれば、今日まで、インターネットを経てビデオをストリーミングするサーバーホスト型ビデオゲームには制限があった。更に、高度の双方向性を要求するアプリケーションのユーザは、そのアプリケーションが一般的なインターネット及びストリームビデオでホストされる場合には、同様に制限で悩まされている。このようなサービスは、ホスティングサーバーが、商業的な設定において、ヘッドエンド（ケーブルブロードバンドの場合）、又は中央オフィス（デジタルサブスクライバライン(DSL)の場合）、或いはLAN（又は特殊グレードのプライベート接続）内に直接設定されて、クライアント装置からサーバーへのルート及び距離が待ち時間を最小にするように制御され、且つ待ち時間を被ることなくピークを受け容れ得るようなネットワークコンフィギュレーションを要求する。LAN（典型的に100Mbpsから1Gbps定格）及び充分な帯域巾の賃貸ラインは、典型的に、ピーク帯域巾要求をサポートすることができる（例えば、18Mbpsのピーク帯域巾は、100MbpsのLAN容量の小部分である）。

【0068】

又、ピーク帯域巾要求は、特殊な受け容れがなされる場合には、住居用ブロードバンドインフラストラクチャーにより受け容れることができる。例えば、ケーブルTVシステムでは、デジタルビデオトラフィックに、大きなIフレームのようなピークを取り扱うことのできる専用帯域巾が与えられる。そしてDSLシステムでは、高速DSLモデムを準備して、高いピークを許すことができるか、又は高いデータレートを取り扱うことのできる特殊グレードの接続を準備することができる。しかし、一般的なインターネットに取り付けられる従来のケーブルモデム及びDSLインフラストラクチャーは、圧縮ビデオのピーク帯域要件に対して遥かに低い寛容度しか有していない。従って、クライアント装置から長距離にあるサーバーセンターにおいてビデオゲーム又はアプリケーションをホストし、次いで、インターネットを経て従来の住居用ブロードバンド接続を通して圧縮ビデオ出力をストリーミングするオンラインサービスは、特に、非常に短い待ち時間を要求するゲーム及びアプリケーション（例えば、ファーストパーソンシューター及び他のマルチユーザ双方向アクションゲーム、又は高速応答時間を要求するアプリケーション）に関して、顕著な待ち時間及びピーク帯域巾制限で悩まされている。

【0069】

本開示は、添付図面及び以下の詳細な説明から、より完全に理解されよう。しかしながら、ここに開示した要旨は、本発明を例示するものに過ぎず、ここに示す特定の実施形態に限定されるものではない。

【図面の簡単な説明】

【0070】

【図1】従来のビデオゲームシステムのアーキテクチャーを示す。

【図2a】一実施形態による高レベルシステムアーキテクチャーを示す。

【図2b】一実施形態による高レベルシステムアーキテクチャーを示す。

【図3】クライアントとサーバーとの間の通信に対する実際のデータレート、定格データレート、及び要求されるデータレートを示す。

10

20

30

40

50

- 【図 4 a】一実施形態により使用されたホスティングサービス及びクライアントを示す。
- 【図 4 b】クライアントとホスティングサービスとの間の通信に関連した例示的待ち時間を示す。
- 【図 4 c】一実施形態によるクライアント装置を示す。
- 【図 4 d】別の実施形態によるクライアント装置を示す。
- 【図 4 e】図 4 c のクライアント装置のブロック図である。
- 【図 4 f】図 4 d のクライアント装置のブロック図である。
- 【図 5】一実施形態により使用できるビデオ圧縮の一形式を例示する。
- 【図 6 a】別の実施形態に使用できるビデオ圧縮の一形式を例示する。
- 【図 6 b】低い複雑さ、低アクションのビデオシーケンスの送信に関連したデータレートのピークを示す。 10
- 【図 6 c】高い複雑さ、高アクションのビデオシーケンスの送信に関連したデータレートのピークを示す。
- 【図 7 a】一実施形態に使用されるビデオ圧縮技術を示す。
- 【図 7 b】一実施形態に使用されるビデオ圧縮技術を示す。
- 【図 8】一実施形態に使用される付加的なビデオ圧縮技術を示す。
- 【図 9 a】本発明の一実施形態に使用されるフレームレート処理技術を示す。
- 【図 9 b】本発明の一実施形態に使用されるフレームレート処理技術を示す。
- 【図 9 c】本発明の一実施形態に使用されるフレームレート処理技術を示す。
- 【図 10 a】映像タイルをパケット内に効率的にパックする一実施形態を示す。 20
- 【図 10 b】映像タイルをパケット内に効率的にパックする一実施形態を示す。
- 【図 11 a】順方向エラー修正技術を使用する実施形態を示す。
- 【図 11 b】順方向エラー修正技術を使用する実施形態を示す。
- 【図 11 c】順方向エラー修正技術を使用する実施形態を示す。
- 【図 11 d】順方向エラー修正技術を使用する実施形態を示す。
- 【図 12】圧縮のためのマルチコア処理ユニットを示す一実施形態を示す。
- 【図 13 a】実施形態による地理的ポジショニング及びホスティングサービス間の通信を示す。
- 【図 13 b】実施形態による地理的ポジショニング及びホスティングサービス間の通信を示す。 30
- 【図 14】クライアントとホスティングサービスとの間の通信に関連した待ち時間を例示する。
- 【図 15】ホスティングサービスのサーバーセンターアーキテクチャーを示す。
- 【図 16】複数の生のビデオウインドウを含むユーザインターフェイスの一実施形態のスクリーンショットを例示する。
- 【図 17】特定のビデオウインドウを選択した後の図 16 のユーザインターフェイスを示す。
- 【図 18】特定のビデオウインドウをフルスクリーンサイズへズームインした後の図 17 のユーザインターフェイスを示す。
- 【図 19】マルチプレーヤゲームのスクリーンにオーバーレイされる共同ユーザビデオデータを例示する。 40
- 【図 20】ホスティングサービスにおけるゲームプレーヤのためのユーザページを例示する。
- 【図 21】3D 双方向広告を例示する。
- 【図 22】生の演技の表面捕獲からのテクスチャ処理表面を有するホトリアルな映像を発生するための一連のステップを例示する。
- 【図 23】リニアメディアコンテンツの選択を許すユーザインターフェイスページを例示する。
- 【図 24】ウェブページが生である前に経過する時間長さを接続速度に対して示すグラフである。 50

【図 2 5 a】クライアント装置からホスティングサービスへのフィードバックチャンネルを使用する本発明の実施形態を示す。

【図 2 5 b】クライアント装置からホスティングサービスへのフィードバックチャンネルを使用する本発明の実施形態を示す。

【図 2 6 a】首尾良く受信されるように最後に知られたタイル／フレームに基づきタイル／フレームをエンコードする実施形態を示す。

【図 2 6 b】首尾良く受信されるように最後に知られたタイル／フレームに基づきタイル／フレームをエンコードする実施形態を示す。

【図 2 7 a】ゲーム又はアプリケーションの状態が第 1 のホスティングサービス又はサーバーから第 2 のホスティングサービス又はサーバーへポートされる実施形態を示す。

10

【図 2 7 b】ゲーム又はアプリケーションの状態が第 1 のホスティングサービス又はサーバーから第 2 のホスティングサービス又はサーバーへポートされる実施形態を示す。

【図 2 8】異なるデータを使用してゲーム又はアプリケーションの状態がポートされる一実施形態を示す。

【図 2 9】クライアント装置に一時的デコーダを使用する本発明の一実施形態を示す。

【図 3 0】本発明の一実施形態により「I タイル」が「R フレーム」間にどのように散在されるか示す。

【図 3 1 a】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

【図 3 1 b】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

20

【図 3 1 c】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

【図 3 1 d】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

【図 3 1 e】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

【図 3 1 f】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

【図 3 1 g】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

30

【図 3 1 h】生のストリーム及び／又は 1 つ以上の H Q ストリームを発生する本発明の実施形態を示す。

【発明を実施するための形態】

【0 0 7 1】

以下の説明では、本開示を完全に理解するために、装置の形式、システムコンフィギュレーション、通信方法、等の特定の細部について述べる。しかしながら、当業者であれば、ここに述べる実施形態を具現化するのに、これらの特定の細部は必要ないことが明らかであろう。

【0 0 7 2】

40

図 2 a - b は、ビデオゲーム及びソフトウェアアプリケーションが、契約サービスのもとでインターネット 2 0 6（或いは他のパブリック又はプライベートネットワーク）を経てユーザの家屋 2 1 1（「ユーザの家屋」とは、ユーザが所在する場所を意味し、移動装置を使用する場合には屋外も含む）においてホスティングサービス 2 1 0 によりホストされそしてクライアント装置 2 0 5 によりアクセスされるような 2 つの実施形態の高レベルアーキテクチャーを示す。クライアント装置 2 0 5 は、内部又は外部ディスプレイ装置 2 2 2 を有してインターネットにワイヤード又はワイヤレス接続される M i c r o s o f t W i n d o w s 又は L i n u x（登録商標）ベースの P C 又は A p p l e 社の M a c i n t o s h コンピュータのような汎用コンピュータでもよいし、又はビデオ及びオーディオをモニタ又は T V 受像機 2 2 2 へ出力するセットトップボックス（インターネット

50

にワイヤード又はワイヤレス接続される)のような専用クライアント装置でもよいし、或いはおそらくインターネットにワイヤレス接続される移動装置でもよい。

【0073】

これらの装置は、いずれも、それ自身のユーザ入力装置(例えば、キーボード、ボタン、タッチスクリーン、トラックパッド又は慣性感知棒、ビデオ捕獲カメラ及び/又は運動追跡カメラ、等)を有してもよいし、又は有線又は無線で接続された外部入力装置221(例えば、キーボード、マウス、ゲームコントローラ、慣性感知棒、ビデオ捕獲カメラ、及び/又は運動追跡カメラ、等)を使用してもよい。以下に詳細に述べるように、ホスティングサービス210は、高パワーのCPU/GPU処理能力を伴うものを含めて、種々の性能レベルのサーバーを含む。ゲームのプレイ中、又はホスティングサービス210におけるアプリケーションの使用中に、家庭又はオフィス用のクライアント装置205は、ユーザからのキーボード及び/又はコントローラ入力を受け取り、次いで、インターネット206を通してホスティングサービス210へコントローラ入力を送信し、このホスティングサービス210は、それに応答してゲームプログラムを実行し、そしてゲーム又はアプリケーションソフトウェアのためのビデオ出力の連続フレーム(一連のビデオ映像)を発生する(例えば、ユーザがボタンを押して、スクリーン上のキャラクタを右へ移動するように指令する場合には、ゲームプログラムが、右へ移動するキャラクタを示す一連のビデオ映像を生成する)。この一連のビデオ映像は、次いで、短待ち時間のビデオコンプレッサを使用して圧縮され、次いで、ホスティングサービス210は、インターネット206を通して短待ち時間のビデオストリームを送信する。家庭又はオフィス用クライアント装置は、次いで、圧縮されたビデオストリームをデコードし、そしてモニタ又はTVにおいてその解凍されたビデオ映像をレンダリングする。その結果、クライアント装置205のコンピューティング及びグラフィックハードウェア要件が著しく緩和される。クライアント205は、キーボード/コントローラ入力をインターネット206へ転送し、そしてインターネット206から受け取った圧縮されたビデオストリームをデコードし解凍するための処理パワーを有するだけでよく、これは、実質上、パーソナルコンピュータが、今日、そのCPUにおいてソフトウェアで実行できることである(例えば、ほぼ2GHzで実行されるインテル社のCore Duo CPUは、H.264及びWindows Media VC9のようなコンプレッサを使用してエンコードされた720p HDTVを解凍することができる)。そして、クライアント装置のケースでは、専用のチップも、このような規格に対するビデオ解凍を、リアルタイムで、遥かに低いコストにおいて、且つ近代的なPCに要求されるような汎用CPUより遥かに低い消費電力で遂行することができる。特に、コントローラ入力を転送しそしてビデオを解凍する機能を遂行するために、家庭用クライアント装置205は、特殊なグラフィック処理ユニット(GPU)、光学的ドライブ又はハードドライブ、例えば、図1に示す従来のビデオゲームシステムを要求しない。

【0074】

ゲーム及びアプリケーションソフトウェアが、より複雑になり且つよりホトリアリスティックになるにつれて、それらは、高性能CPU、GPU、より多くのRAM、及びより大きくて高速のディスクドライブを要求し、且つホスティングサービス210のコンピューティングパワーは、アップグレードし続けるが、エンドユーザは、家庭又はオフィス用のクライアントプラットフォーム205を更新することが要求されない。というのは、その処理要件は、所与のビデオ解凍アルゴリズムでの表示解像度及びフレームレートに対して一定のままだからである。従って、図2a-bに示すシステムには、今日見られるハードウェア制限及び互換性の問題が生じない。

【0075】

更に、ゲーム及びアプリケーションソフトウェアは、ホスティングサービス210のサーバーでのみ実行されるので、ゲーム又はアプリケーションソフトウェアのコピー(光学的メディアの形態又はダウンロードされたソフトウェアとして)がユーザの家庭又はオフィスに存在することはない(ここで使用する「オフィス」とは、特に指示のない限り、例

えば、学校の教室を含めて、非住居設定を含むものとする）。これは、ゲーム又はアプリケーションソフトウェアが不法にコピーされる（海賊版作成される）おそれを著しく軽減すると共に、海賊版作成されるゲーム又はアプリケーションにより貴重なデータベースが使用されるおそれを軽減する。実際に、家庭又はオフィスでの使用に実際的でないゲーム又はアプリケーションソフトウェアをプレイするために特殊なサーバーが要求される（例えば、非常に高価で、大きく又はノイズを発生する装置が要求される）場合には、ゲーム又はアプリケーションソフトウェアの海賊版コピーが得られたとしても、家庭又はオフィスで動作することはできない。

【0076】

一実施形態では、ホスティングサービス210は、ソフトウェア開発ツールをゲーム又はアプリケーションソフトウェア開発者（一般的に、ソフトウェア開発会社、ゲーム又は映画スタジオ、或いはゲーム又はアプリケーションソフトウェア発行者を指す）220に与え、この開発者は、ホスティングサービス210において実行できるゲームを設計するようにビデオゲームを設計する。そのようなツールは、スタンドアローンPC又はビデオゲームコンソールでは通常利用できないホスティングサービスの特徴を開発者が利用できるようにする（例えば、複雑な幾何学形状の非常に大きなデータベースへの高速アクセス（「幾何学形状」とは、特に指示のない限り、ここでは、多角形、テクスチャ、リギング、照明、挙動、並びに3Dデータベースを定義する他のコンポーネント及びパラメータを指すものとする））。

【0077】

このアーキテクチャーのもとでは、異なるビジネスモデルが考えられる。1つのモデルのもとで、ホスティングサービス210は、図2aに示すように、エンドユーザから契約料金を収集し、そして開発者220にロイヤリティを支払う。図2bに示す別の具現化においては、開発者220がユーザから契約料金を直接収集し、そしてゲーム又はアプリケーションコンテンツをホストするためにホスティングサービス210に支払をする。これらの基礎的な原理は、オンラインゲーム又はアプリケーションホスティングを提供するための特定のビジネスモデルに限定されない。

【0078】

圧縮ビデオ特性

上述したように、ビデオゲームサービス又はアプリケーションソフトウェアサービスをオンラインで提供することに伴う1つの顕著な問題は、待ち時間である。（ユーザにより入力装置が操作される点から、ディスプレイ装置に応答が表示される点までの）70から80msの待ち時間が、高速応答時間を要求するゲーム及びアプリケーションの上限である。しかしながら、図2a及び2bに示すアーキテクチャーの環境では多数の実際的及び物理的な制約があるために、これを達成することが非常に困難である。

【0079】

図3に示すように、ユーザがインターネットサービスに契約するときに、その接続は、典型的に、ユーザの家庭又はオフィスへの公称最大データレート301によって定格付けされる。プロバイダーのポリシー及びルーティング装置の能力に基づいて、その最大データレートは、若干厳格に施行できるが、典型的に、実際に得られるデータレートは、多数の異なる理由の1つで、低いものとなる。例えば、DSL中央オフィス又はローカルケーブルモデムループに非常に多くのネットワークトラフィックがあるか、又はケーブルにノイズが生じてパケットをドロップさせるか、或いはプロバイダーが最大ビット数/月/ユーザを確立することがある。現在、ケーブル及びDSLサービスに対する最大ダウンロードデータレートは、典型的に、数百キロビット/秒(Kbps)から30Mbpsの範囲である。セルラーサービスは、典型的に、数百Kbpsのダウンロードデータに制限される。しかしながら、ブロードバンドサービスの速度及びブロードバンドサービスに契約するユーザの数は、時間と共に急激に増加する。現在、ある分析では、米国のブロードバンド加入者の33%が2Mbps以上のダウンロードデータレートを有することが推定される。例えば、ある分析では、2010年までに、米国のブロードバンド加入

者の 85%以上が 2Mbps 以上のデータレートを有することが推定される。

【0080】

図 3 に示すように、実際に利用可能な最大データレート 302 は、時間と共に変動し得る。従って、短い待ち時間のオンラインゲーム又はアプリケーションソフトウェアコンテキストでは、時々、特定のビデオストリームに対する実際に利用できるデータレートを予想することが困難である。所与の数値のフレーム／秒 (fps) における所与のレベルのクオリティをある量のシーンの複雑さに対し所与の解像度 (例えば、640x480@60fps) において維持するためにデータレート 303 が要求され、そして (図 3 にピークで示されるように) 実際に利用できる最大データレート 302 より上に動きが上昇する場合には、多数の問題が発生し得る。例えば、あるインターネットサービスは、単にパケットをドロップし、ユーザのビデオスクリーンにデータロス及び映像歪／ロスを招くことになる。他のサービスは、付加的なパケットを一時的にバッファし (即ち、キューイングし)、そして利用可能なデータレートでクライアントへパケットを与えて、待ち時間の増加、即ち多数のビデオゲーム及びアプリケーションにとって受け容れられない結果を招くことになる。最終的に、あるインターネットサービスプロバイダーは、データレートの増加を、悪意のある攻撃、例えば、サービス拒否攻撃 (ネットワーク接続を無能力化するためにハッカーにより使用される良く知られた技術) とみなし、そしてユーザのインターネット接続を指定期間中切断する。従って、ここに述べる実施形態は、ビデオゲームに要求されるデータレートが、利用可能な最大データレートを越えないことを保証するステップをとる。

【0081】

ホスティングサービスアーキテクチャー

図 4 a は、一実施形態によるホスティングサービス 210 のアーキテクチャーを示す。ホスティングサービス 210 は、単一サーバーセンターに配置することもできるし、又は複数のサーバーセンターにわたり分散することもできる (あるサーバーセンターへの経路が他のサーバーセンターより短い待ち時間となる短待ち時間接続をユーザに与え、ユーザ間に負荷バランスを与え、そして 1 つ以上のサーバーセンターがフェイルする場合に冗長性を与えるために)。ホスティングサービス 210 は、最終的に、数百又は数千或いは数百万のサーバー 402 を含み、非常に大きなユーザベースにサービスすることができる。ホスティングサービスコントロールシステム 401 は、ホスティングサービス 210 に対する全体的なコントロールを与え、ルーター、サーバー、ビデオ圧縮システム、ビリング及びアカウンティングシステム、等に指令する。一実施形態では、ホスティングサービスコントロールシステム 401 は、ユーザ情報、サーバー情報及びシステム統計情報のためのデータベースを記憶するのに使用される RAID アレイに結合された分散処理 Linux ベースシステムにおいて具現化される。以上の説明では、ホスティングサービス 210 により具現化される種々のアクションは、他の特定システムに起因しない限り、ホスティングサービスコントロールシステム 401 により開始されコントロールされる。

【0082】

ホスティングサービス 210 は、インテル、IBM、ヒューレットパッカード、等から現在入手できるもののような多数のサーバー 402 を含む。或いは又、サーバー 402 は、コンポーネントのカスタム構成でアッセンブルすることもできるし、又は全サーバーが単一チップとして具現化されるように最終的に一体化することもできる。この図は、図示明瞭化のために、少数のサーバー 402 しか示していないが、実際の展開では、1 つ程度のサーバー 402 があってもよいし、又は数百万以上のサーバー 402 があってもよい。サーバー 402 は、全てが同様に構成されてもよいし (幾つかの構成パラメータの例として、同じ CPU 形式及び性能で、GPU をもったりもたなかったりして、そして GPU をもつ場合には、同じ GPU 形式及び性能で、又、同じ数の CPU 及び GPU で、又、同じ量及び形式／速度の RAM で、並びに同じ RAM 構成で)、又はサーバー 402 の種々のサブセットが同じ構成を有してもよいし (例えば、サーバーの 25% をある仕方で構成することができ、50% を異なる仕方で構成することができ、そして 25% を更に別の仕方

で構成することができ）、或いは各サーバー４０２が異なるものでもよい。

【００８３】

一実施形態では、サーバー４０２は、ディスクレスであり、即ちそれ自身のローカル大量記憶装置（光学又は磁気記憶装置、或いは半導体ベースの記憶装置、例えば、フラッシュメモリ又は同様の機能を果たす他の大量記憶手段）を有するのではなく、各サーバーは、高速バックプレーン又はネットワーク接続を通して共有大量記憶装置にアクセスする。一実施形態では、この高速接続は、一連の独立ディスク冗長アレイ（RAID）４０５に接続された記憶エリアネットワーク（SAN）４０３であり、装置間の接続がギガビットイーサネット（登録商標）を使用して具現化される。当業者に明らかなように、SAN ４０３は、多数のRAIDアレイ４０５を一緒に合成して、最終的に広い帯域巾を生じさせ、即ち現在のゲームコンソール及びPCに使用されるRAMから得られる帯域巾に接近するか又はそれを潜在的に越えるのに使用される。そして、磁気メディアのような回転メディアをベースとするRAIDアレイは、著しいシーク時間アクセス待ち時間をしばしば有するが、半導体記憶装置をベースとするRAIDアレイは、相当に短いアクセス待ち時間で具現化することができる。別の構成では、サーバー４０２の幾つか又は全部が、それら自身の大量記憶装置の幾つか又は全部をローカルで与える。例えば、サーバー４０２は、そのオペレーティングシステムのような頻繁にアクセスされる情報、及びビデオゲーム又はアプリケーションのコピーを短待ち時間のローカルフラッシュベースの記憶装置に記憶するが、幾何学形状又はゲーム状態情報の大きなデータベースに時々アクセスするには、SANを使用して、回転メディアをベースとするRAIDアレイ４０５に大きなシーク待ち時間でアクセスする。

10

20

【００８４】

更に、一実施形態では、ホスティングサービス２１０は、以下に詳細に説明する短待ち時間のビデオ圧縮ロジック４０４を使用する。このビデオ圧縮ロジック４０４は、ソフトウェア、ハードウェア又はその組合せで具現化することができる（その幾つかの実施形態を以下に説明する）。ビデオ圧縮ロジック４０４は、オーディオ及びビジュアル資料を圧縮するためのロジックを含む。

【００８５】

動作に際し、ユーザの家屋２１１において、キーボード、マウス、ゲームコントローラ又は他の入力装置４２１を経てビデオゲームをプレイするか又はアプリケーションを使用する間に、クライアント４１５のコントロール信号ロジック４１３は、ユーザにより作動されたボタン押圧（及び他の形式のユーザ入力）を表す（典型的にUDPパケットの形態の）コントロール信号４０６a-bを送信する。所与のユーザからのコントロール信号は、適当なサーバー（又は複数のサーバーがユーザの入力装置に応答する場合には複数のサーバー）４０２へルーティングされる。図４aに示すように、コントロール信号４０６aは、SANを経てサーバー４０２へルーティングされる。それとは別に又はそれに加えて、コントロール信号４０６bは、ホスティングサービスネットワーク（例えば、イーサネットベースのローカルエリアネットワーク）を経てサーバー４０２へ直接ルーティングされる。それらがどのように送信されるかに関わらず、サーバー（１つ又は複数）は、コントロール信号４０６a-bに応答してゲーム又はアプリケーションソフトウェアを実行する。図４aに示されていないが、ファイアウォール（１つ又は複数）及び／又はゲートウェイ（１つ又は複数）のような種々のネットワークコンポーネントは、ホスティングサービス２１０の縁（例えば、ホスティングサービス２１０とインターネット４１０との間）において、及び／又はインターネット４１０と家庭又はオフィスクライアント４１５との間のユーザの家屋２１１の縁において到来するトラフィック及び出て行くトラフィックを処理することができる。実行されたゲーム又はアプリケーションソフトウェアのグラフィック及びオーディオ出力、即ちビデオ映像の新たなシーケンスは、短待ち時間のビデオ圧縮ロジック４０４へ供給され、このロジックは、ここに述べるような短待ち時間のビデオ圧縮技術に基づいてビデオ映像のシーケンスを圧縮し、そして圧縮されたビデオストリームを、典型的に、圧縮された又は非圧縮のオーディオと共に、インターネット４１０を経

30

40

50

て（又は、以下に述べるように、一般的なインターネットをバイパスする最適な高速ネットワークサービスを経て）クライアント 4 1 5 へ返送する。クライアント 4 1 5 における短待ち時間のビデオ解凍ロジック 4 1 2 は、次いで、ビデオ及びオーディオストリームを解凍し、その解凍されたビデオストリームをレンダリングし、そして典型的に、その解凍されたオーディオストリームをディスプレイ装置 4 2 2 においてプレイする。或いは又、オーディオは、ディスプレイ装置 4 2 2 とは個別のスピーカで再生してもよいし、そうでなくてもよい。入力装置 4 2 1 及びディスプレイ装置 4 2 2 は、図 2 a 及び 2 b では独立した装置として示されているが、ポータブルコンピュータ又は移動装置のようなクライアント装置内に一体化されてもよいことに注意されたい。

【0086】

家庭又はオフィスクライアント 4 1 5（図 2 a 及び 2 b において家庭又はオフィスクライアント 2 0 5 として既に述べた）は、非常に安価で且つ低電力の装置で、計算又はグラフィック性能が非常に限定されていると共に、ローカルの大量記憶装置が非常に限定されたものであるか又はそれを全く有していない。対照的に、S A N 4 0 3 及び複数の R A I D 4 0 5 に結合された各サーバー 4 0 2 は、非常に高い性能のコンピューティングシステムであり、そして実際に、複数のサーバーが並列処理構成で協働的に使用される場合は、保持することのできる計算及びグラフィック処理パワーの量にほぼ制限がなくなる。そして、ユーザにとって知覚的に短待ち時間のビデオ圧縮 4 0 4 及び短待ち時間のビデオ圧縮 4 1 2 のために、サーバー 4 0 2 の計算パワーがユーザに与えられる。ユーザが入力装置 4 2 1 のボタンを押すと、ディスプレイ 4 2 2 上の映像は、ゲーム又はアプリケーションソフトウェアがローカルで実行されるかのように、知覚的に意義のある遅延を伴わずに、ボタンの押圧に応答して更新される。従って、短待ち時間のビデオ解凍及びコントロール信号ロジック 4 1 3 を具現化する非常に低性能のコンピュータ又は安価なチップである家庭又はオフィスクライアント 4 1 5 では、ローカルで利用できると思われるリモート位置から効果的な任意の計算パワーがユーザに与えられる。これは、最も進歩したプロセッサ集中の（典型的に新規な）ビデオゲーム及び最高性能のアプリケーションをプレイするパワーをユーザに与える。

【0087】

図 4 c は、非常に基本的で且つ安価な家庭又はオフィスクライアント装置 4 6 5 を示す。この装置は、図 4 a 及び 4 b からの家庭又はオフィスクライアント 4 1 5 の実施形態である。これは約 2 インチの長さである。これは、パワーオーバーイーサネット（P o E）でイーサネットケーブルとインターフェイスするイーサネットジャック 4 6 2 を有し、そこから、インターネットへの電力及び接続が得られる。ネットワークアドレ스트ランスレーション（N A T）をサポートするネットワーク内で N A T を実行することができる。オフィス環境では、多数の新規なイーサネットスイッチは、P o E を有し、そして P o E をオフィスのイーサネットジャックへ直接持っていく。このような状況では、壁ジャックからクライアント 4 6 5 へのイーサネットケーブルが要求されるだけである。利用できるイーサネット接続が（例えば、D S L 又はケーブルモデムをもつが、P o E はもたない家庭において）電力を搬送しない場合には、非給電イーサネットケーブル及び P o E 付き出力イーサネットを受け容れる安価な壁「ブリック」（即ち電源）が利用できる。

【0088】

クライアント 4 6 5 は、キーボード、マウス、ゲームコントローラ及び／又はマイクロホン及び／又はヘッドセットのようなブルーーツース入力装置 4 7 9 とインターフェイスするブルーーツースワイヤレスインターフェイスに結合された（図 4 a の）コントロール信号ロジック 4 1 3 を含む。又、クライアント 4 6 5 の一実施形態は、1 2 0 f p s ビデオをサポートできるディスプレイ装置 4 6 8 に結合されるビデオを 1 2 0 f p s で出力し、そしてシャッター付き眼鏡 4 6 6 に信号を送って（典型的に赤外線により）、各次々のフレームで片方の目に、次いで、他方の目に交互にシャッター作動することができる。ユーザが認知する効果は、ディスプレイスクリーンを「跳び出す」ステレオ 3 D 映像である。このようなオペレーションをサポートする 1 つのこのようなディスプレイ装置 4 6 8 は、S

10

20

30

40

50

amsung HL-T5076Sである。各目のビデオストリームは個別であるから、一実施形態では、独立した2つのビデオストリームがホスティングサービス210により圧縮され、フレームは、時間的にインターリーブされ、そしてフレームは、クライアント465内の独立した2つの解凍プロセスとして解凍される。

【0089】

又、クライアント465は、到来するビデオ及びオーディオを解凍してHDMI（高鮮明度マルチメディアインターフェイス）を通して出力する短待ち時間のビデオ解凍ロジック412と、SDTV（標準鮮明度テレビジョン）又はHDTV（高鮮明度テレビジョン）468に差し込まれてTVにビデオ及びオーディオを与えるか又はHDMIをサポートするモニタ468に差し込まれるコネクタ463とを含む。ユーザのモニタ468がHDMIをサポートしない場合には、HDMI対DVI（デジタルビジュアルインターフェイス）を使用することができるが、オーディオが失われる。HDMI規格のもとでは、ディスプレイの能力（例えば、サポートされる解像度、フレームレート）464がディスプレイ装置468から通信され、この情報は、次いで、インターネット接続462を通してホスティングサービス210へ返送され、従って、ディスプレイ装置に適したフォーマットで圧縮ビデオをストリーミングすることができる。

【0090】

図4dは、より多くの外部インターフェイスを有する以外は図4cに示す家庭又はオフィスクライアント装置465と同じである家庭又はオフィスクライアント装置475を示す。又、クライアント475は、電力に対してPoEを受け容れることもできるし、又は壁に差し込まれる外部電源アダプタ（図示せず）から延びることもできる。クライアント475のUSB入力を使用すると、ビデオカメラ477は、圧縮されたビデオをクライアント475へ供給し、これは、クライアント475によってホスティングサービス210へアップロードされ、以下に述べるように使用される。カメラ477に内蔵されるのは、以下に述べる圧縮技術を使用した短待ち時間のコンプレッサである。

【0091】

インターネット接続としてイーサネットコネクタを有するのに加えて、クライアント475は、インターネットへの802.11gワイヤレスインターフェイスも有する。両インターフェイスは、NATをサポートするネットワーク内でNATを使用することができる。

【0092】

又、ビデオ及びオーディオを出力するためにHDMIコネクタを有するのに加えて、クライアント475は、アナログ出力を含む（且つ標準アダプタケーブルでVGA出力を与える）デュアルリンクDVI-Iコネクタも有する。又、これは、複合ビデオ及びS-ビデオのためのアナログ出力も有する。

【0093】

オーディオに対して、クライアント475は、左右のアナログステレオRCAジャックを、そしてデジタルオーディオ出力に対して、TOSLINK出力を有する。

【0094】

又、入力装置479へのブルーーツワイヤレスインターフェイスに加えて、入力装置にインターフェイスするためのUSBジャックも有する。

【0095】

図4eは、クライアント465の内部アーキテクチャの一実施形態を示す。図示された装置の全部又は幾つかを、フィールドプログラマブルロジックアレー、カスタムASIC、或いはカスタム設計又は既製の多数の個別装置で具現化することができる。

【0096】

PoEを伴うイーサネット497がイーサネットインターフェイス481に取り付けられる。PoEを伴うイーサネット497から電力499が導出され、クライアント465内の残りの装置に接続される。バス480は、装置間の通信のための共通バスである。

【0097】

フラッシュ 476 からの小さなクライアントコントロールアプリケーションを実行するコントロール CPU 483（ほとんどの場合に、RAM が埋設された 100 MHz の MIPS R4000 シリーズ CPU のような小型の CPU で充分である）は、ネットワーク（即ち、イーサネットインターフェイス）のためのプロトコルスタックを具現化し、又、ホスティングサービス 210 と通信し、そしてクライアント 465 内の全ての装置を構成する。又、これは、入力装置 469 とのインターフェイスを取り扱い、そして必要に応じて、「順方向エラー修正」で保護して、パケットをユーザコントローラデータと共にホスティングサービス 210 へ返送する。又、コントロール CPU 483 は、パケットトラフィックを監視する（例えば、パケットが失われるか遅延された場合、それらの到着にタイムスタンプも押す）。この情報は、ホスティングサービス 210 へ返送され、従って、ネットワーク接続を常時監視し、それに依りて何を送信するか調整することができる。フラッシュメモリ 476 は、最初に、製造時に、コントロール CPU 483 のコントロールプログラムがロードされると共に、特定のクライアント 465 ユニットにとって独特のシリアル番号もロードされる。このシリアル番号は、ホスティングサービス 210 がクライアント 465 ユニットを独特に識別できるようにする。

10

【0098】

ブルーツースインターフェイス 484 は、クライアント 465 の内部にあるそのアンテナを通して入力装置 469 へワイヤレス通信する。

【0099】

ビデオデコンプレッサ 486 は、ここに述べるビデオ解凍を具現化するように構成された短待ち時間のビデオデコンプレッサである。非常に多数のビデオ解凍装置が、既製品として、又は FPGA 又はカスタム ASIC へ一体化できる知的プロパティ (IP) 設計として、存在する。H. 264 デコーダのための IP を提供する 1 つの会社は、NSW オーストラリアのオーシャンロジックオブマンリー (Ocean Logic of Manly) である。IP を使用する効果は、ここで使用する圧縮技術が圧縮規格に従わないことである。ある標準的なデコンプレッサは、ここに述べる圧縮技術を受け容れるように構成されるに充分なほど融通性があるが、あるものはそうではない。しかし、IP では、デコンプレッサを必要に応じて設計し直す上で完全な融通性がある。

20

【0100】

ビデオデコンプレッサの出力は、ビデオ出力サブシステム 487 に結合され、これは、ビデオを HDMI インターフェイス 490 のビデオ出力に結合する。

30

【0101】

オーディオ解凍サブシステム 488 は、入手可能な標準的オーディオデコンプレッサを使用して具現化されるか、又は IP として具現化することもでき、或いはオーディオの解凍は、例えば、(Vorbis.com において入手できる) Vorbis オーディオデコンプレッサを具現化できるコントロールプロセッサ 483 内で具現化することもできる。

【0102】

オーディオの解凍を具現化する装置は、オーディオ出力サブシステム 489 へ結合され、これは、オーディオを HDMI インターフェイス 490 のオーディオ出力へ結合する。

40

【0103】

図 4f は、クライアント 475 の内部アーキテクチャーの一実施形態を示す。明らかに、このアーキテクチャーは、付加的なインターフェイスと、壁に差し込む電源アダプタからの任意の外部 DC 電力を除き、クライアント 465 と同じであり、その外部 DC 電力は、そのように使用される場合、イーサネット PoE 497 から到来する電力に取って代わる。クライアント 465 と共通の機能は、以下に説明を繰り返さず、付加的な機能を以下に説明する。

【0104】

CPU 483 は、付加的な装置と通信し、それを構成する。

【0105】

50

W i F i サブシステム 4 8 2 は、そのアンテナを通してイーサネット 4 9 7 への代替物としてワイヤレスインターネットアクセスを与える。W i F i サブシステムは、カリフォルニア州サンタクララのアセロスコミュニケーションズ(Atheros Communications)を含めて幅広い製造者から入手できる。

【0106】

U S B サブシステム 4 8 5 は、ワイヤード U S B 入力装置 4 7 9 に対してブルー투스通信の代替物を与える。U S B サブシステムは、極めて標準的で、F P G A 及び A S I C について入手容易であり、且つビデオ解凍と同様に、他の機能を遂行する既製装置にしばしば内蔵される。

【0107】

ビデオ出力サブシステム 4 8 7 は、クライアント 4 6 5 内よりも広範囲なビデオ出力を発生する。これは、H D M I 4 9 0 ビデオ出力を与えるのに加えて、D V I - I 4 9 1、S - ビデオ 4 9 2 及び複合ビデオ 4 9 3 を与える。又、D V I - I 4 9 1 インターフェイスがデジタルビデオに対して使用されるときには、ディスプレイ能力 4 6 4 がディスプレイ装置からコントロール C P U 4 8 3 へ返送され、ディスプレイ装置 4 7 8 の能力をホスティングサービス 2 1 0 に通知できるようにする。ビデオ出力サブシステム 4 8 7 により与えられる全てのインターフェイスは、極めて標準的なインターフェイスであり、多数の形態で入手容易である。

【0108】

オーディオ出力サブシステム 4 8 9 は、デジタルインターフェイス 4 9 4 (S/P D I F 及び/又は T o s l i n k) を通してオーディオをデジタルで出力すると共に、ステレオアナログインターフェイス 4 9 5 を通してオーディオをアナログ形態で出力する。

【0109】

往復待ち時間の分析

もちろん、以上の段落を理解するために、入力装置 4 2 1 を使用したユーザのアクションと、そのアクションの結果をディスプレイ装置 4 2 0 で見るときとの間の往復待ち時間は、7 0 - 8 0 m s 以下でなければならない。この待ち時間は、ユーザの家屋 2 1 1 内の入力装置 4 2 1 からホスティングサービス 2 1 0 へそしてユーザの家屋 2 1 1 へ戻って、ディスプレイ装置 4 2 2 へ至る経路における全てのファクタを考慮しなければならない。図 4 b は、信号が進行しなければならない種々のコンポーネント及びネットワークを示し、これらコンポーネント及びネットワークの上には、実際の具現化で予想できる待ち時間を列挙した時間線がある。図 4 b は、重要な経路ルーティングだけを示すように簡略化されていることに注意されたい。システムの他の特徴のために使用されるデータの他のルーティングは、以下で説明する。双頭矢印(例えば、矢印 4 5 3)は、往復待ち時間を表し、単頭矢印(例えば、矢印 4 5 7)は、片道待ち時間を表し、そして「~」は、近似尺度を示す。列挙した待ち時間を達成できない実世界の状況があるが、米国内における多くのケースでは、ユーザの家屋 2 1 1 への D S L 及びケーブルモデム接続を使用し、次の段落で述べる環境においてこれら待ち時間を達成できることを指摘しなければならない。又、インターネットへのセルラーワイヤレス接続は、図示されたシステムにおいて確実に機能するが、ほとんどの現在の U S セルラーデータシステム(E V D O のような)は、非常に長い待ち時間を被り、図 4 b に示す待ち時間を達成できないことにも注意されたい。しかしながら、これらの基礎的な原理は、このレベルの待ち時間を具現化できる将来のセルラー技術において具現化することができよう。更に、現在の U S セルラーデータシステムを経て被る待ち時間が、ユーザに気付くものであるが、ゲーム又はアプリケーションにとって許容できるようなゲーム及びアプリケーションシナリオ(例えば、チェスのような速いユーザ反応時間を要求しないゲーム)もある。

【0110】

ユーザの家屋 2 1 1 における入力装置 4 2 1 からスタートして、ユーザが入力装置 4 2 1 を操作すると、ユーザコントロール信号がクライアント 4 1 5 へ送信され(これは、セットトップボックスのようなスタンドアローン装置でもよいし、或いは P C 又は移動装置

10

20

30

40

50

のような別の装置で実行されるソフトウェア又はハードウェアでもよい）、そして（一実施形態ではUDPフォーマットで）パケット化され、そのパケットには、ホスティングサービス210に到着するための行先アドレスが与えられる。又、パケットは、コントロール信号がどのユーザから到来するか指示する情報も含む。次いで、コントロール信号のパケット（1つ又は複数）は、ファイアウォール／ルーター／NAT（ネットワークアドレストランスレーション）装置443を通してWANインターフェイス442へ転送される。WANインターフェイス442は、ユーザのISP（インターネットサービスプロバイダー）によりユーザの家屋211に与えられるインターフェイス装置である。WANインターフェイス442は、ケーブル又はDSLモデム、WiMaxトランシーバー、ファイバートランシーバー、セルラーデータインターフェイス、インターネットプロトコル・オーバー・パワーラインインターフェイス、又はインターネットへの多数のインターフェイスの他のものでよい。更に、ファイアウォール／ルーター／NAT装置443（及び潜在的にWANインターフェイス442）は、クライアント415に一体化されてもよい。その一例は、家庭又はオフィスクライアント415の機能を具現化するためのソフトウェアと、ある規格（例えば、802.11g）によりインターネットへワイヤレスでルーティング及び接続する手段とを含む移動電話である。

【0111】

WANインターフェイス442は、次いで、コントロール信号をユーザのインターネットサービスプロバイダー（ISP）のための「存在点(point of presence)」441と称されるものへルーティングし、これは、ユーザの家屋211に接続されたWANポートと、一般的インターネット又はプライベートネットワークとの間のインターフェイスをなすファシリティである。存在点の特性は、提供されるインターネットサービスの性質に基づいて変化する。DSLの場合、これは、典型的に、DSLAMが配置される電話会社の中央オフィスとなる。ケーブルモデムの場合、これは、典型的に、ケーブルマルチシステムオペレータ(MSO)ヘッドエンドとなる。セルラーシステムの場合、これは、典型的に、セルラータワーに関連したコントロールルームとなる。しかし、存在点の性質が何であれ、これは、コントロール信号パケット（1つ又は複数）を一般的インターネット410へルーティングする。コントロール信号パケット（1つ又は複数）は、次いで、最も多くはファイバートランシーバーインターフェイスであるものを通して、ホスティングサービス210へのWANインターフェイス441へルーティングされる。WAN441は、次いで、コントロール信号パケットをルーティングロジック409（これは、イーサネットスイッチ及びルーティングサーバーを含めて、多数の異なる仕方で具現化される）へルーティングし、これは、ユーザのアドレスを評価し、そしてコントロール信号を所与のユーザに対する正しいサーバー402へルーティングする。

【0112】

サーバー402は、次いで、コントロール信号を、サーバー402で実行されるゲーム又はアプリケーションソフトウェアに対する入力として取り上げ、そしてそのコントロール信号を使用して、ゲーム又はアプリケーションの次のフレームを処理する。次のフレームが発生されると、サーバー402からビデオコンプレッサ404へビデオ及びオーディオが出力される。ビデオ及びオーディオは、サーバー402から種々の手段を経てコンプレッサ404へ出力される。先ず、コンプレッサ404は、サーバー402に組み込むことができ、従って、圧縮をサーバー402内でローカルに具現化することができる。又は、ビデオ及び／又はオーディオは、パケット化形態で、イーサネット接続のようなネットワーク接続を経て、サーバー402とビデオコンプレッサ404との間のプライベートネットワークであるネットワークへ、又はSAN403のような共有ネットワークを通して、出力することができる。或いは、ビデオは、サーバー402から、DVI又はVGAコネクタのようなビデオ出力コネクタを通して出力され、次いで、ビデオコンプレッサ404により捕獲されてもよい。又、オーディオは、サーバー402からデジタルオーディオとして（例えば、TOSLINK又はS/PDIFコネクタを経て）出力されるか又はアナログオーディオとして出力されてもよく、アナログオーディオは、ビデオコンプレッサ

404内のオーディオ圧縮ロジックによりデジタル化されエンコードされる。

【0113】

ビデオコンプレッサ404は、ビデオフレーム、及びサーバー402からのそのフレーム時間中に発生されたオーディオを捕獲すると、以下に述べる技術を使用してビデオ及びオーディオを圧縮する。ビデオ及びオーディオが圧縮されると、アドレスと共にパケット化されて、ユーザのクライアント415へ返送され、WANインターフェイス441へルーティングされ、このWANインターフェイスは、ビデオ及びオーディオパケットを一般的なインターネット410を経てルーティングし、このインターネットは、ビデオ及びオーディオパケットをユーザのISP存在点441へルーティングし、この存在点は、ビデオ及びオーディオパケットをユーザの家屋のWANインターフェイス442へルーティングし、このインターフェイスは、ビデオ及びオーディオパケットをファイアウォール／ルーター／NAT装置443へルーティングし、次いで、この装置は、ビデオ及びオーディオパケットをクライアント415へルーティングする。

【0114】

クライアント415は、ビデオ及びオーディオを解凍し、次いで、ビデオをディスプレイ装置422（又はクライアントの内蔵ディスプレイ装置）に表示すると共に、オーディオをディスプレイ装置422又は個別の増幅器／スピーカへ或いはクライアントに内蔵された増幅器／スピーカへ送信する。

【0115】

上述した全プロセスに知覚的に遅れがないことをユーザが認知するためには、往復遅延が70又は80ms未満である必要がある。前記往復経路における待ち時間遅延のあるものは、ホスティングサービス210及び／又はユーザのコントロールのもとにあり、他のものは、そうではない。それでも、非常に多数の実世界のシナリオの分析及びテストに基づき、おおよその測定値は次のようになる。

【0116】

コントロール信号451を送信するための片道送信時間は、典型的に、1ms未満であり、ユーザの家屋452を通る往復ルーティングは、典型的に、イーサネット上に入手容易な消費者グレードのファイアウォール／ルーター／NATスイッチを使用して約1msで達成される。ユーザISPは、その往復遅延453が広く変化するが、DSL及びケーブルモデムプロバイダーでは、典型的に、10ないし25msである。一般的なインターネット410における往復待ち時間は、トラフィックがどのようにルーティングされるか及びルートに欠陥があるかどうか（これらの問題は以下で述べる）に基づいて大きく変化するが、典型的に、一般的なインターネットは、非常に最適なルートを与え、そして待ち時間は、行先までの距離が与えられると、主として光ファイバを通る光の速度によって決定される。以下に更に述べるように、ユーザの家屋211から離れてホスティングサービス210を発することが予想される最も遠いおおよその距離として1000マイルを確立した。1000マイル（往復2000マイル）では、インターネットを通る信号の実際の通過時間は、約22msである。ホスティングサービス210へのWANインターフェイス441は、典型的に、待ち時間を無視できる商業的グレードファイバの高速インターフェイスである。従って、一般的インターネットの待ち時間454は、典型的に、1ないし10msである。ホスティングサービス210を通る片道ルーティング455の待ち時間は、1ms未満である。サーバー402は、典型的に、ゲーム又はアプリケーションのための新たなフレームを1フレーム時間（60fpsでは16.7msである）未満で計算し、従って、使用するために合理的な最大片道待ち時間は、16msである。ここに述べるビデオ圧縮及びオーディオ圧縮アルゴリズムの最適なハードウェア具現化では、圧縮457を1msで完了することができる。あまり最適でない形態では、圧縮に6ms程度を要する（もちろん、もっと最適でない形態は、更に長い時間を要するが、このような具現化は、往復の全待ち時間に影響を及ぼし、70－80msの待ち時間目標を維持するには他の待ち時間を短縮する必要がある（例えば、一般的なインターネットを通る許容距離を短縮することができる））。インターネット454、ユーザISP453及びユーザ家屋

10

20

30

40

50

ルーティング 4 5 2 の往復待ち時間は、既に考慮されており、従って、残りは、ビデオ解凍 4 5 8 の待ち時間であり、これは、ビデオ解凍 4 5 8 が専用のハードウェアで具現化されるか、又は（P C 又は移動装置のような）クライアント装置 4 1 5 においてソフトウェアで具現化されるかに基づいて、又、ディスプレイのサイズ及び解凍 C P U の性能に基づいて、変化する。典型的に、解凍 4 5 8 は、1 ないし 8 m s を要する。

【0 1 1 7】

従って、実際上見られる最悪のケースの待ち時間を全部加算することにより、図 4 a に示すシステムのユーザにより経験されると予想できる最悪のケースの往復待ち時間を決定することができる。それらは、 $1 + 1 + 2.5 + 2.2 + 1 + 1.6 + 6 + 8 = 80 \text{ ms}$ である。そして、正に、実際に（以下に述べる注意と共に）、これは、図 4 a に示すシステムの
10
プロトタイプバージョンを使用し、既製品の W i n d o w s P C をクライアント装置として及び米国内の家庭用 D S L 及びケーブルモデム接続を使用して分かったおおよその往復待ち時間である。もちろん、最悪のケースより良好のシナリオでは、非常に短い待ち時間が得られるが、それは、広く使用される商業的サービスの開発に依存することができない。

【0 1 1 8】

一般的なインターネットを経て図 4 b に列挙する待ち時間を得るには、図 4 a のビデオコンプレッサ 4 0 4 及びクライアント 4 1 5 のビデオデコンプレッサ 4 1 2 が、非常に特定の特性をもつパケットストリームを発生し、ホスティングサービス 2 1 0 からディスプレイ装置 4 2 2 までの全経路を経て発生されるパケットシーケンスが、遅延や過剰なパ
20
ケットロスを受けず、且つ特に、W A N インターフェイス 4 4 2 及びファイアウォール／ルーター／N A T 4 4 3 を通るユーザのインターネット接続を経てユーザに利用できる帯域巾の制約内に一貫して入るようにする必要がある。更に、ビデオコンプレッサは、通常のインターネット及びネットワーク送信において生じる不可避なパケットロス及びパケット並べ替えを許容できるに十分なほど頑丈なパケットストリームを生成しなければならない。

【0 1 1 9】

短待ち時間のビデオ圧縮

前記目標を達成するために、一実施形態は、待ち時間を減少し且つビデオを送信するためのピーク帯域巾要件を緩和するというビデオ圧縮の新規な解決策をとる。これらの実施
30
形態を説明する前に、図 5 及び図 6 a - b を参照して、現在のビデオ圧縮技術の分析を行う。もちろん、これらの技術は、これらの技術で要求されるデータレートを取り扱うに十分な帯域巾がユーザに与えられる場合には、基礎的な原理に基づいて使用することができる。オーディオの圧縮は、ビデオの圧縮と同時に且つ同期して具現化されることを述べる以外、ここでは扱わないことに注意されたい。このシステムの要件を満足する従来のオーディオ圧縮技術が存在する。

【0 1 2 0】

図 5 は、ビデオを圧縮するための 1 つの特定の従来技術であって、特定の圧縮アルゴリズムを使用する圧縮ロジック 5 2 0 により各個々のビデオフレーム 5 0 1 - 5 0 3 を圧縮して、一連の圧縮されたフレーム 5 1 1 - 5 1 3 を発生する従来技術を示している。この
40
技術の一実施形態は、「モーション J P E G」であり、各フレームが、離散的コサイン変換（D C T）に基づき、ジョイント・ピクチャー・エクスパート・グループ（J P E G）圧縮アルゴリズムに従って圧縮されるというものである。種々の異なる形式の圧縮アルゴリズムを使用してもよいが、これらの基礎的な原理（例えば、J P E G - 2 0 0 0 のようなウェーブレットベースの圧縮アルゴリズム）に依然適合させるようにする。

【0 1 2 1】

この形式の圧縮の 1 つの問題は、各フレームのデータレートを下げるが、連続フレーム間の類似性を利用してビデオストリーム全体のデータレートを下げるものでないことである。例えば、図 5 に示すように、映像の所与のクオリティに対し、 $640 \times 480 \times 24$ ビット／ピクセル = $640 * 480 * 24 / 8 / 1024 = 900$ キロバイト／フレーム
50

(KB/フレーム)のフレームレートを仮定すれば、モーションJPEGは、10のファクタでストリームを圧縮するだけであり、90KB/フレームのデータストリームを生じさせる。60フレーム/秒では、これは、 $90\text{KB} \times 8\text{ビット} \times 60\text{フレーム/秒} = 42.2\text{Mbps}$ のチャンネル帯域巾を要求し、これは、今日の米国内のほぼ全ての家庭用インターネット接続に対して遥かに広い帯域巾となり、且つ多数のオフィス用インターネット接続に対しても非常に広い帯域巾となる。実際に、このような広い帯域巾で一定のデータストリームを要求し、且つオフィスLAN環境においても1人のユーザにしか役立たないとすれば、 100Mbps のイーサネットLAN帯域巾の大きな割合を消費し、LANをサポートするイーサネットスイッチに大きな負担を掛けることになる。従って、動画ビデオに対する圧縮は、(以下に述べるような)他の圧縮技術と比べたときに、非効率的である。更に、ロッシェ圧縮アルゴリズムを使用するJPEG及びJPEG-2000のような単一フレーム圧縮アルゴリズムは、静止映像では気付かない圧縮欠陥(例えば、シーンにおける密集した葉の中の欠陥は、その密集した葉がどのように見えるべきか目にとって厳密に分からないので、欠陥として見えないことがある)を生じる。しかし、シーンが動くと、欠陥が静止映像では気付かないシーンのエリアにあるにも関わらず、欠陥がフレームごとに変化することが目で検出されるので、欠陥が目立つようになる。その結果、余白的アナログTV受信中に見える「スノー」ノイズに見掛けが類似した「バックグラウンドノイズ」がフレームのシーケンスにおいて知覚される。もちろん、この形式の圧縮は、ここに述べる幾つかの実施形態で依然使用できるが、一般的に述べると、シーンにおけるバックグラウンドノイズを回避するため、所与の知覚上のクオリティに対して高いデータレート(即ち、低い圧縮比)が要求される。

10

20

【0122】

H.264、又はWindows Media VC9、MPEG2及びMPEG4のような他の形式の圧縮は、全て、連続するフレーム間の類似性を利用しているので、ビデオストリームの圧縮がより効率的である。これらの技術は、全て、ビデオを圧縮するための同じ一般的技術に依存している。従って、H.264規格について述べるが、他の種々の圧縮アルゴリズムにも同じ一般の原理が適用される。圧縮H.264のためのx264オープンソースソフトウェアライブラリー、及び解凍H.264のためのFFmpegオープンソースソフトウェアライブラリーを含めて、非常に多数のH.264コンプレッサ及びデコンプレッサを利用することができる。

30

【0123】

図6a及び6bは、従来の圧縮技術を例示するもので、一連の非圧縮のビデオフレーム501-503、559-561が、圧縮ロジック620により、一連の「Iフレーム」611、671、「Pフレーム」612-613、及び「Bフレーム」670へと圧縮される。図6aの縦軸は、一般的に、エンコードされて得られる各フレームのサイズを表す(が、フレームは適切なスケールで描かれていない)。上述したように、Iフレーム、Bフレーム及びPフレームを使用するビデオコーディングは、当業者に良く知られている。簡単に述べると、Iフレーム611は、完全非圧縮のフレーム501のDCTベース圧縮である(上述した圧縮されたJPEG映像と同様)。Pフレーム612-613は、一般的に、Iフレーム611よりサイズが著しく小さい。というのは、手前のIフレーム又はPフレームのデータの利点を取り入れるからであり、即ち手前のIフレーム又はPフレーム間の変化を示すデータを含むからである。Bフレーム670は、Pフレームと同様であるが、Bフレームは、後続する基準フレームにおけるフレームと、先行する基準フレームにおける潜在的なフレームとを使用する。

40

【0124】

以下の説明では、望ましいフレームレートが60フレーム/秒であり、各Iフレームが約160Kbであり、平均Pフレーム及びBフレームが16Kbであり、そして毎秒新たなIフレームが発生されると仮定する。この1組のパラメータで、平均データレートは、 $160\text{Kb} + 16\text{Kb} \times 59 = 1.1\text{Mbps}$ となる。このデータレートは、家庭及びオフィスへの多くの現在のブロードバンドインターネット接続に対する最大データレート内

50

に充分入る。又、この技術は、イントラフレーム専用のエンコーディングからバックグラウンドノイズの問題を回避する傾向がある。というのは、P及びBフレームは、フレーム間の差を追跡し、圧縮欠陥がフレームごとに現れたり消えたりする傾向がなく、前記バックグラウンドノイズの問題を緩和するからである。

【0125】

前記形式の圧縮に伴う1つの問題は、平均データレートが比較的低い（例えば、1.1 Mbps）が、単一のIフレームが送信に多数のフレーム分の時間を要することである。例えば、従来技術を使用すると、60フレームごとに160 KbpsのIフレームの状態
10
で1.1 Mbpsでビデオをストリーミングするのに、典型的に、2.2 Mbpsのネットワーク接続（例えば、図3aから2.2 Mbpsピークの利用可能な最大データレート302をもつDSL又はケーブルモデム）で充分である。これは、ビデオを解凍するまで解凍キューに1秒のビデオを入れておくことにより達成される。1秒で、1.1 Mbpsのデータが送信され、これは、利用可能なデータレートが周期的に50%程度下がると仮定しても、2.2 Mbpsの利用可能な最大データレートによって容易に受け容れられる。不都合なことに、この従来の解決策は、1秒のビデオバッファが受信器にあるので、ビデオについて1秒の待ち時間を生じさせる。このような遅延は、多数の従来の用途（例えば、リニアビデオの再生）では充分であるが、70-80 msを越える待ち時間を許容できない高速アクションのビデオゲームに対しては遥かに長い待ち時間である。

【0126】

1秒のビデオバッファを排除する試みがなされたが、高速アクションビデオゲームのための充分な待ち時間短縮が得られていない。一例として、上述したBフレームの使用は、Iフレームに先行する全てのBフレーム及びIフレームの受信を必要とする。PフレームとBフレームとの間で59個の非Iフレームがおおよそ分割されると仮定すれば、少なくとも29個のBフレームがあり、そしてBフレームを表示できるまでにIフレームが受信される。従って、チャンネルの利用可能な帯域巾に関わらず、各々1/60秒巾の29+1=30フレームの遅延、即ち500 msの待ち時間が必要になる。明らかに、これは、遥かに長過ぎる。
20

【0127】

従って、別の解決策は、Bフレームを排除し、そしてI及びPフレームだけを使用することである。（その1つの結果として、所与のクオリティレベルに対してデータレートが増加することになるが、この例における一貫性のために、各Iフレームが160 Kbであり、平均Pフレームが16 Kbのサイズであり、従って、データレートが依然1.1 Mbpsであると仮定し続ける。）この解決策は、Bフレームにより導入される不可避な待ち時間を排除する。というのは、各Pフレームのデコーディングが、以前に受信したフレームに依存するだけだからである。この解決策に伴う問題は、Iフレームが平均Pフレームより非常に大きいために、ほとんどの家庭及び多くのオフィスにおいて典型的である狭帯域巾チャンネルでは、Iフレームの送信が実質的な待ち時間を増大することである。これが図6bに示されている。Iフレームに要求されるピークデータレート623が、利用可能な最大データレート622を（及び定格最大データレート621も）遥かに越えるIフレームを除いて、ビデオストリームのデータレート624は、利用可能な最大データレート621より低い。Pフレームにより要求されるデータレートは、利用可能な最大データレート未満である。2.2 Mbpsの利用可能な最大データレートのピークがその2.2 Mbpsのピークレートに着実に保たれるとしても、Iフレームを送信するのに160 Kb/2.2 Mb=71 msを要し、利用可能な最大データレート622が50%（1.1 Mbps）下がった場合には、Iフレームを送信するのに142 msを要する。従って、Iフレームを送信する際の待ち時間は、71から142 msのどこかに入る。この待ち時間は、図4bに示された待ち時間に加算され、最悪の場合、これが70 msに加算され、従って、これは、ユーザが入力装置421を操作する時点から、ディスプレイ装置422に映像が現れるまで、141ないし222 msの合計往復待ち時間を生じ、遥かに高いものとなる。そして、利用可能な最大データレートが2.2 Mbpsより下がる場合には、
30
40
50

待ち時間は更に増大する。

【0128】

又、一般的に、ピークデータレート623でISPを「ジャミング」させて、利用可能なデータレート622を遥かに越える厳しい結果が生じることにも注意されたい。異なるISPの装置は、異なる挙動をするが、利用可能なデータレート622より非常に高いデータレートでパケットを受け取るときには、次の挙動がDSL及びケーブルモデムISPの間で極めて一般的となる。(a)パケットをキューイングすることでパケットを遅延させ(待ち時間を導入し)、(b)パケットの幾つか又は全部をドロップさせ、(c)ある期間接続をディスエイブルする(おそらく、ISPに関するものであるから、「サービス拒絶」攻撃のような悪意のある攻撃である)。従って、図6bに示すような特性で全データレートにおいてパケットストリームを送信することは、実現可能な選択肢ではない。ピーク623は、ホスティングサービス210においてキューイングされ、そして利用可能な最大データレートより低いデータレートで送信され、上述した許容できない待ち時間を導入することがある。

【0129】

更に、図6bに示すビデオストリームデータレートシーケンス624は、非常に「従順な」ビデオストリームデータレートシーケンスであり、そして著しく変化せず且つほとんど動きのないビデオシーケンスからビデオを圧縮することにより生じると予想される種類のデータレートシーケンスである(例えば、カメラが固定位置にあってほとんど動かず、そしてシーン内のオブジェクト、例えば、椅子に座って話している人がほとんど動きを示さないビデオ遠隔会議において一般的であるように)。

【0130】

図6cに示すビデオストリームデータレートシーケンス634は、動画又はビデオゲーム或いはあるアプリケーションソフトウェアにおいて発生されるような、遥かに多いアクションを伴うビデオから見ると予想されるものに対して典型的なシーケンスである。Iフレームピーク633に加えて、極めて大きく且つ多くの場合に利用可能な最大データレートを越える635及び636のようなPフレームピークもあることに注意されたい。これらのPフレームピークは、Iフレームピークと同程度に大きいことはあまりないが、全データレートでチャンネルにより搬送されるには遥かに大き過ぎるもので、又、Iフレームピークと同様に、Pフレームピークは、ゆっくり送信されねばならない(従って、待ち時間が増大する)。

【0131】

広帯域巾チャンネル(例えば、100MbpsのLAN、又は広帯域巾100Mbpsのプライベート接続))において、ネットワークは、Iフレームピーク633又はPフレームピーク636のような大きなピークを許容することができ、そして原理的に、短い待ち時間を維持することができる。しかし、このようなネットワークは、多くの場合、大勢のユーザ間に共有され(例えば、オフィス環境において)、そしてこのような「ピーク」データは、特に、ネットワークトラフィックがプライベート共有接続ヘルレーティングされた場合に(例えば、リモートデータセンターからオフィスへ)、LANの性能に影響を及ぼす。先ず、この例は、60fpsにおいて640x480ピクセルの比較的低解像度のビデオストリームであることを銘記されたい。60fpsにおいて1920x1080のHDTVストリームは、近代的なコンピュータ及びディスプレイによって容易に取り扱われ、そして60fpsにおいて2560x1440解像度のディスプレイが益々入手可能になってきている(例えば、アップル社の30"ディスプレイ)。又、60fpsにおいて1920x1080の高アクションビデオシーケンスは、適度なクオリティレベルに対してH.264圧縮を使用して4.5Mbpsを必要とする。公称データレートの10倍のIフレームピークを仮定すれば、それは45Mbps以下のピークを生じるが、まだかなりのPフレームピークがある。大勢のユーザが同じ100Mbpsのネットワーク(例えば、オフィスとデータセンターとの間のプライベートなネットワーク接続)を経てビデオストリームを受信する場合、大勢のユーザのビデオストリームからのピークがどのよう

に整列して、ネットワークの帯域巾を圧倒し、そしてネットワーク上のユーザをサポートするスイッチのバックプレーンの帯域巾を潜在的に圧倒するか容易に見ることができる。ギガビットのイーサネットネットワークの場合でも、十分なユーザが十分なピークを一度に整列させる場合は、ネットワーク又はネットワークスイッチを圧倒することができる。そして、2560x1440解像度のビデオがより平凡なものになると、平均ビデオストリームデータレートが9.5Mbpsとなり、おそらく、95Mbpsのピークデータレートを生じさせる。言うまでもなく、データセンターとオフィスとの間の100Mbpsの接続（これは、今日、非常に高速な接続である）は、単一ユーザからのピークトラフィックによって完全に沈没される。従って、LAN及びプライベートネットワーク接続が、ピークストリーミングビデオを更に許容できるとしても、高いピークを伴うストリーミングビデオは、望ましいものではなく、オフィスのIT部門による特殊なプランニング及び適応を要求する。

10

【0132】

もちろん、標準的なリニアビデオアプリケーションの場合、これらの事柄は、問題ではない。というのは、データレートが送信点において「平滑化」され、各フレームのデータが利用可能な最大データレート622より低く、そしてクライアントのバッファが、I、P及びBフレームのシーケンスを、それらが解凍されるまで記憶するからである。従って、ネットワークにわたるデータレートは、ビデオストリームの平均データレートの近くに保たれる。不都合なことに、これは、Bフレームが使用されない場合でも待ち時間を導入し、即ち高速応答時間を要求するビデオゲーム及びアプリケーションのような短待ち時間アプリケーションについては受け容れられない。

20

【0133】

高いピークを有するビデオストリームを軽減する1つの従来の解決策は、「コンスタントビットレート」（CBR）エンコーディングとしばしば称される技術を使用することである。CBRという語は、全てのフレームが同じビットレート（即ち、サイズ）を有するように圧縮されることを意味すると思われるが、それが通常指すのは、ある数のフレーム（ここでは、1フレーム）にわたり最大ビットレートが許されるという圧縮パラダイムである。例えば、図6cのケースでは、ビットレートを、定格最大データレート621の例えば70%に制限するエンコーディングにCBR制約が適用される場合に、圧縮アルゴリズムは、定格最大データレート621の70%以上を使用して通常圧縮されるフレームが少ないビットで圧縮されるように、各フレームの圧縮を制限する。その結果、所与のクオリティレベルを維持するために通常より多くのビットを要求するフレームがビットの「欠乏」となり、これらフレームの映像クオリティは、定格最大データレート621の70%以上のビットを要求しない他のフレームの場合より悪くなる。この解決策は、（a）予想される動き又はシーンの変化があまりなく、且つ（b）ユーザが周期的なクオリティ低下を受け容れ得るようなある形式の圧縮ビデオに対して、受け容れられる結果を生じさせることができる。CBRに適したアプリケーションの良い例は、ビデオ遠隔会議である。というのは、ピークが僅かしかなく、且つクオリティが短時間低下した場合に（例えば、カメラがパンして、著しいシーンの動き及び大きなピークを生じ、そのパン中に、高クオリティ映像圧縮に十分なビットがなく、映像クオリティの低下を招く場合に）、ほとんどのユーザにそれが受け容れられるからである。不都合なことに、CBRは、非常に複雑な又は多くの動きのあるシーンを有し、及び／又は適度な一定レベルのクオリティが要求される他の多数のアプリケーションにはあまり適していない。

30

40

【0134】

一実施形態に使用される短待ち時間の圧縮ロジック404は、高いクオリティを維持しながら短待ち時間の圧縮ビデオをストリーミングすることに伴うある範囲の問題に対処するために多数の異なる技術を使用する。第1に、短待ち時間の圧縮ロジック404は、Iフレーム及びPフレームしか発生せず、各Bフレームをデコードするために多数のフレーム分の時間を待機する必要性を緩和する。更に、図7aに示すように、一実施形態では、短待ち時間の圧縮ロジック404は、各非圧縮フレーム701-760を一連の「タイル

50

」へと細分化し、そして各タイルをIフレーム又はPフレームのいずれかとして個々にエンコードする。圧縮されたIフレーム及びPフレームのグループは、ここでは、「Rフレーム」711-770と称される。図7aに示す特定例では、各非圧縮フレームが、16タイルの4x4マトリクスへと細分化される。しかしながら、これらの基礎的な原理は、特定の細分化スキムに限定されない。

【0135】

一実施形態では、短待ち時間の圧縮ロジック404は、ビデオフレームを多数のタイルに分割し、そして各フレームからの1つのタイルをIフレームとしてエンコード（即ち、圧縮）し（即ち、タイルは、フル映像サイズの1/16の個別ビデオフレームであるかのように圧縮され、この「ミニ」フレームに使用される圧縮は、Iフレーム圧縮である）、そして残りのタイルをPフレームとしてエンコード（即ち、圧縮）する（即ち、1/16の各「ミニ」フレームに使用される圧縮は、Pフレーム圧縮である）。Iフレーム及びPフレームとして圧縮されるタイルは、各々、「Iタイル」及び「Pタイル」と称される。各連続するビデオフレームでは、Iタイルとしてエンコードされるべきタイルが変化される。従って、所与のフレーム時間では、ビデオフレーム内のタイルのうちの1つのタイルだけがIタイルであり、そして残りのタイルがPタイルである。例えば、図7aでは、非圧縮フレーム701のタイル0がIタイル I_0 としてエンコードされ、そして残りの1-15タイルがPタイル P_1 - P_{15} としてエンコードされ、Rフレーム711を形成する。次の非圧縮ビデオフレーム702では、非圧縮フレーム701のタイル1がIタイル I_1 としてエンコードされ、そして残りのタイル0及び2から15がPタイル P_0 及び P_2 から P_{15} としてエンコードされ、Rフレーム712を形成する。従って、タイルとしてIタイル及びPタイルは、連続するフレームにわたり時間的に次々にインターリーブされる。このプロセスは、マトリクス内の最後のタイルがIタイル（即ち、 I_{15} ）としてエンコードされてRタイル770が発生されるまで続く。次いで、このプロセスが再開されて、フレーム711のような別のRフレームが発生する（即ち、タイル0に対してIタイルをエンコードし、等々）。図7aには示されていないが、一実施形態では、Rフレームのビデオシーケンスのうちの第1のRフレームがIタイルしか含まない（即ち、その後のPフレームが、動きを計算するための基準映像データを有するように）。或いは又、一実施形態では、スタートアップシーケンスが、通常と同じIタイルパターンを使用するが、Iタイルでまだエンコードされていないタイルに対してPタイルを含まない。換言すれば、あるタイルは、第1のIタイルが到着するまでデータでエンコードされず、従って、以下に詳細に述べる図9aのビデオストリームデータレート934におけるスタートアップピークを回避する。更に、以下に述べるように、これらの基礎的な原理に依然適合しつつ、タイルに対して種々の異なるサイズ及び形状を使用することができる。

【0136】

クライアント415で実行されるビデオ解凍ロジック412は、各タイルを、小さなI及びPフレームの個別のビデオシーケンスであるかのように解凍し、次いで、各タイルをフレームバッファドライブレディスプレイ装置422へレンダリングする。例えば、Rフレーム711ないし770からの I_0 及び P_0 を使用して、ビデオ映像のタイル0を解凍し、レンダリングする。同様に、Rフレーム711ないし770からの I_1 及び P_1 を使用して、タイル1を再構成し、等々となる。上述したように、Iフレーム及びPフレームの解凍は、良く知られた技術であり、Iタイル及びPタイルの解凍は、クライアント415で実行されるビデオ解凍の複数のインスタンスを得ることにより遂行することができる。乗算プロセスは、クライアント415の計算負担を増大すると思われるが、実際には、そうではない。というのは、タイルそれ自体は、付加的なプロセスの数に対し比例的に小さく、従って、表示されるピクセルの数は、1つのプロセスがあって従来のフルサイズのI及びPフレームを使用する場合と同じだからである。

【0137】

このRフレーム技術は、図6b及び6cに示したIフレームに関連した帯域中ピークを著しく軽減する。というのは、所与のフレームが、大抵は、Iフレームより典型

的に小さなPフレームで作られるからである。例えば、典型的なIフレームが160Kbであると再び仮定すれば、図7aに示す各フレームのIタイルは、この量のほぼ1/16即ち10Kbとなる。同様に、典型的なPフレームが16Kbであると仮定すれば、図7aに示す各タイルのPフレームは、ほぼ1Kbとなる。最終の結果として、ほぼ10Kb + 15 * 1Kb = 25KbのRフレームとなる。従って、各60フレームシーケンスは、25Kb * 60 = 1.5Mbpsとなる。従って、60フレーム/秒では、これは、1.5Mbpsの帯域巾を維持することのできるチャンネルを要求するが、Iタイルのために相当に低いピークが60フレームインターバル全体にわたって分布される。

【0138】

Iフレーム及びPフレームに対して同じデータレートを仮定した先の例では、平均データレートが1.1Mbpsであったことに注意されたい。これは、先の例では、新たなIフレームが60フレームの時間ごとに一度しか導入されず、一方、この例では、16フレームの時間、ひいては、Iフレームに等しい時間でIフレームサイクルを作り上げる16個のタイルが16フレームの時間ごとに導入され、著しく高い平均データレートとなるからである。実際に、より頻繁にIフレームを導入しても、データレートを直線的に高めることはない。これは、Pフレーム（又はPタイル）が主として手前のフレームから次のフレームへの相違をエンコードするという事実のためである。従って、手前のフレームが次のフレームにかなり類似している場合には、Pフレームが非常に小さくなり、一方、手前のフレームが次のフレームとかなり相違がある場合には、Pフレームが非常に大きくなる。しかし、Pフレームは、主として、実際のフレームではなく、手前のフレームから導出されるので、それにより生じるエンコードされたフレームは、十分なビット数でIフレームより大きなエラー（例えば、視覚欠陥）を含むことができる。そして、1つのPフレームに別のPフレームが続くときには、エラーの蓄積が生じ、長いシーケンスのPフレームがあるときには悪化する。ここで、精巧なビデオコンプレッサは、一連のPフレームの後に映像のクオリティが低下することを検出し、必要に応じて、より多くのビットをその後続Pフレームに割り当てて、クオリティを上げるか、又はそれが最も効率的なアクションのコースである場合には、PフレームをIフレームに置き換える。従って、長いシーケンスのPフレーム（例えば、上述した例のように59個のPフレーム）が使用されるとき、特に、シーンが多大な複雑さ及び/又は動きを有するときには、典型的に、PフレームがIフレームから更に離れるにつれて、より多くのビットがPフレームに必要とされる。

【0139】

或いは、Pフレームを逆の視点から見ると、Iフレームに接近して続くPフレームは、Iフレームから更に離れたPフレームよりビットを必要としない傾向がある。従って、図7aに示す例では、先行するIフレームから15フレーム以上離れたPフレームはなく、一方、先の例では、Pフレームは、Iフレームから59フレーム離れることができる。従って、より頻繁にIフレームがあると、Pフレームは、より小さくなる。もちろん、厳密な相対的サイズは、ビデオストリームの性質に基づいて変化するが、図7aの例では、Iタイルが10Kbである場合、Pタイルは、平均で、0.75Kbのみのサイズとなり、10Kb + 15 * 0.75Kb = 21.25Kbとなるか、又は60フレーム/秒では、データレートは、21.25Kb * 60 = 1.3Mbpsとなり、又はIフレームの後に59個のPフレームが続く1.1Mbpsのストリームより約16%高いデータレートとなる。この場合も、ビデオ圧縮に対するこれら2つの解決策間の相対的な結果は、ビデオシーケンスに基づいて変化するが、典型的に、Rフレームを使用すると、I/Pフレームシーケンスを使用する場合より、所与のクオリティレベルに対して約20%多くのビットを要求することが実験で分かった。しかし、当然、Rフレームは、ピークを劇的に減少させ、I/Pフレームシーケンスより遥かに短い待ち時間でビデオシーケンスを使用可能にする。

【0140】

Rフレームは、ビデオシーケンスの性質、チャンネルの信頼性、及び利用できるデータレートに基づいて、種々の異なる仕方で構成することができる。別の実施形態では、4x

10

20

30

40

50

4 構成の 1 6 とは異なる数のタイルが使用される。例えば、2 枚のタイルを 2 x 1 又は 1 x 2 構成で使うことができ、4 枚のタイルを 2 x 2、4 x 1 又は 1 x 4 構成で使うことができ、6 枚のタイルを 3 x 2、2 x 3、6 x 1 又は 1 x 6 構成で使うことができ、或いは 8 枚のタイルを 4 x 2 (図 7 b に示すような)、2 x 4、8 x 1 又は 1 x 8 構成で使うことができる。タイルは、方形である必要がなく、又、ビデオフレームも、方形又は長方形である必要がないことに注意されたい。タイルは、使用するビデオフレーム及びアプリケーションに最も適した形状に分割することができる。

【0 1 4 1】

別の実施形態では、I 及び P タイルのサイクルがタイルの数に固定されない。例えば、8 タイルの 4 x 2 構成では、図 7 b に示すように 1 6 サイクルシーケンスを依然使用することができる。順次の非圧縮フレーム 7 2 1、7 2 2、7 2 3 は、各々、8 枚のタイル、0-7 に分割され、各タイルが個々に圧縮される。R フレーム 7 3 1 から、タイル 0 だけが I タイルとして圧縮され、残りのタイルは、P タイルとして圧縮される。その後の R フレーム 7 3 2 については、8 枚のタイル全部が P タイルとして圧縮され、次いで、その後の R フレーム 7 3 3 については、タイル 1 が I タイルとして圧縮され、そして他のタイル全部が P タイルとして圧縮される。従って、1 6 個のフレームについてシーケンスが続き、I タイルは、1 つおきのフレームのみに発生され、最後の I タイルは、1 5 番目のフレーム時間中にタイル 7 について発生され (図 7 b には示さず)、そして第 1 6 番目のフレーム時間中に、全 P タイルを使用して R フレーム 7 8 0 が圧縮される。次いで、タイル 0 が I タイルとして圧縮されそして他のタイルが P タイルとして圧縮されるようにしてシーケンスが再開される。先の実施形態と同様に、全ビデオシーケンスの各第 1 フレームは、典型的に、全て I タイルであり、その点から前方への P タイルの基準を与える。I タイル及び P タイルのサイクルは、タイルの数の偶数倍である必要がない。例えば、8 枚のタイルでは、I タイルをもつ各フレームの後に、全て P タイルをもつ 2 つのフレームが続き、その後に、別の I タイルが使用される。更に別の実施形態では、あるタイルは、例えば、スクリーンのあるエリアがより多くの動きを有し頻繁な I タイルを要求すると分かっているが、他のエリアはより静的であって (例えば、ゲームのスコアを示す) 頻繁な I タイルをほとんど要求しない場合に、I タイルが他のタイルより頻繁にある状態でシーケンスされる。更に、各フレームは、図 7 a-b において単一の I タイルと共に示されているが、(送信チャンネルの帯域巾に基づいて) 単一のフレーム内で複数の I タイルをエンコードすることもできる。逆に、あるフレーム又はフレームシーケンスを、I タイルなしで (即ち、P タイルのみで) 送信することもできる。

【0 1 4 2】

先の段落で述べた解決策が良く機能する理由は、各単一フレームにわたって I タイルを分布させないことで、大きなピークを生じると思われるが、システムの挙動は、そのように単純ではないことである。各タイルは、他のタイルとは個別に圧縮されるので、タイルが小さくなるほど、各タイルのエンコーディングは、効率的でなくなる。というのは、所与のタイルのコンプレッサが、他のタイルからの類似映像特徴及び類似の動きを利用できないためである。従って、スクリーンを 1 6 枚のタイルに分割すると、一般的に、スクリーンを 8 枚のタイルに分割する場合よりエンコーディングの効率が低下する。しかし、スクリーンが 8 枚のタイルに分割され、そして全 I フレームのデータを、1 6 フレームごとではなく 8 フレームごとに導入させる場合には、全体的に非常に高いデータレートとなる。従って、8 フレームごとではなく 1 6 フレームごとに全 I フレームを導入することにより、全体的なデータレートが減少される。又、1 6 枚の小さなタイルではなく 8 枚の大きなタイルを使用することにより、全データレートが減少され、大きなタイルにより生じるデータピークをある程度軽減させる。

【0 1 4 3】

別の実施形態では、図 7 a 及び 7 b の短待ち時間のビデオ圧縮ロジック 4 0 4 は、R フレームにおける種々のタイルへのビットの割り当てを、圧縮されるべきビデオシーケンスの既知の特性に基づく設定で予め構成することにより、或いは各タイルにおける映像クオ

リティの進行中の分析に基づいて自動的に、コントロールする。例えば、ある競争ビデオゲームでは、（シーンにおいて比較的動きの少ない）プレーヤの車の前部が、シーンの下半分の大部分を占有し、一方、シーンの上半分が、ほぼ常に動いている接近する道路、ビル及び風景で完全に埋められる。圧縮ロジック 404 が各タイルに等しい数のビットを割り当てると、図 7b の非圧縮フレーム 721 のスクリーンの下半分のタイル（タイル 4-7）が、一般的に、図 7b の非圧縮フレーム 721 のスクリーンの上半分のタイル（タイル 0-3）より高いクオリティで圧縮される。この特定のゲーム又はゲームのこの特定のシーンがこのような特性を有すると分かっている場合には、ホスティングサービス 210 のオペレータは、スクリーンの下部のタイルよりスクリーンの上部のタイルにより多くのビットを割り当てるように圧縮ロジック 404 を構成することができる。或いは、圧縮ロジック 404 は、フレームが圧縮された後に（ピーク信号対雑音比（PSNR）のような多数の圧縮クオリティメトリックの 1 つ以上を使用して）タイルの圧縮のクオリティを評価することができ、そしてある時間ウィンドウにわたり、あるタイルが高いクオリティの結果を一貫して生じると決定する場合に、種々のタイルが同様のクオリティレベルに達するまで、低いクオリティの結果を生じるタイルにより多くのビットを徐々に割り当てる。別の実施形態では、圧縮ロジック 404 は、特定のタイル又はタイルのグループに高いクオリティを得るようにビットを割り当てる。例えば、スクリーンの縁より中心部のクオリティを高くするように全体的に良好な知覚上の見掛けを与えることができる。

【0144】

一実施形態では、ビデオストリームのある領域の解像度を改善するために、ビデオ圧縮ロジック 404 は、シーンの複雑さ及び／又は動きが比較的大きいビデオスクリーンのエリアを、シーンの複雑さ及び／又は動きが比較的小さいビデオスクリーンのエリアより小さいタイルを使用してエンコードする。例えば、図 8 に示すように、1 つの R フレーム 811 の 1 つのエリア内の動くキャラクタ 805 の周りに小さなタイルが使用される（潜在的に、その後、同じタイルサイズをもつ一連の R フレーム（図示せず）が続く）。次いで、キャラクタ 805 が映像の新たなエリアへ移動すると、図示されたように、別の R フレーム 812 内のこの新たなエリアの周りに小さなタイルが使用される。上述したように、これらの基礎的な原理に依然合致しながら、種々の異なるサイズ及び形状を「タイル」として使用することができる。

【0145】

上述した循環する I/P タイルは、ビデオストリームのデータレートのピークを実質的に減少するが、特に、動画、ビデオゲーム及びあるアプリケーションソフトウェアで発生する急速に変化する又は非常に複雑なビデオ映像の場合には、ピークを完全に排除することができない。例えば、突然のシーンの移行中に、複雑なフレームの後に、完全に異なる別の複雑なフレームが続くことがある。多数の I タイルが若干のフレームの時間だけシーンの移行に先行することがあっても、それらは、この状態において助けとならない。というのは、新たなフレーム資料が手前の I タイルと関係がないからである。このような状態（及び何もかもが変化しなくても、多くの映像が変化する他の状態）では、ビデオコンプレッサ 404 は、P タイルの全部でなくてもその多くが I タイルとしてより効率的にコード化され、その結果、そのフレームに対するデータレートに非常に大きなピークが生じると決定する。

【0146】

上述したように、ほとんどの消費者グレードのインターネット接続（及び多くのオフィス接続）では、図 6c に 622 として示す利用可能な最大データレートを越えるデータを定格最大データレート 621 と共に「ジャム(jam)」することが単純に実行できないケースがある。定格最大データレート 621（例えば、“6 Mbps DSL”）は、インターネット接続の購入を考えるユーザにとって本質的にマーケティングナンバーであるが、一般的に、性能のレベルを保証するものではない。本発明の説明上、これは、無関係である。というのは、ビデオが接続を通してストリーミングされるときには、利用可能な最大データレート 622 だけが問題だからである。その結果、図 9a 及び 9c において、ピ

ークの問題に対する解決策を説明するときに、定格最大データレートがグラフから削除され、利用可能な最大データレート 9 2 2 だけが示される。ビデオストリームデータレートは、利用可能な最大データレート 9 2 2 を越えてはならない。

【0147】

これに対処するために、ビデオコンプレッサ 4 0 4 が第 1 に実行すべきことは、チャンネルが着実に取り扱うことのできるデータレートであるピークデータレート 9 4 1 を決定することである。このレートは、多数の技術によって決定することができる。1 つのこのような技術は、図 4 a 及び 4 b において、ホスティングサービス 2 1 0 からクライアント 4 1 5 へ次第に高いデータレートテストストリームを徐々に送信し、そしてクライアントがパケットロス及び待ち時間のレベルに関してホスティングサービスヘフィードバックを与えるようにさせるものである。パケットロス及び／又は待ち時間が鋭い増加を示し始めるときには、それは、利用可能な最大データレート 9 2 2 に到達しつつあるという指示である。その後、ホスティングサービス 2 1 0 は、適度な期間中受け容れられるパケットロスレベルでテストストリームが受信され且つ待ち時間がほぼ最小であることをクライアント 4 1 5 が報告するまで、テストストリームのデータレートを徐々に減少することができる。これは、ピーク最大データレート 9 4 1 を確立し、これは、次いで、ストリーミングビデオのためのピークデータレートとして使用される。時間と共に、ピークデータレート 9 4 1 が変動し（例えば、家の別のユーザがインターネット接続を激しく使用し始める場合に）、そしてクライアント 4 1 5 は、それを常時監視して、パケットロス又は待ち時間が増大するかどうか調べる必要があり、利用可能な最大データレート 9 2 2 が以前に確立されたピークデータレート 9 4 1 より降下することを指示し、そしてもしそうであれば、ピークデータレート 9 4 1 を指示する。同様に、時間と共に、クライアント 4 1 5 は、パケットロス及び待ち時間が最適なレベルに保たれることを見つけた場合には、ビデオコンプレッサがデータレートをゆっくり増加して利用可能な最大データレートが増加したかどうか調べることを要求でき（例えば、家の別のユーザがインターネット接続の激しい使用を止めた場合に）、そして利用可能な最大データレート 9 2 2 を越えたことをパケットロス及び／又は長い待ち時間が指示し、ピークデータレート 9 4 1 に対して低いレベルを再び見つけることができるが、それがおそらく増加データレートのテストの前のレベルより高いものとなるまで、再び待機する。従って、この技術（及びそれに類似した他の技術）を使用することにより、ピークデータレート 9 4 1 を見つけて、必要に応じて、周期的に調整することができる。ピークデータレート 9 4 1 は、ユーザへビデオをストリーミングするためにビデオコンプレッサにより使用できる最大データレートを確立する。ピークデータレートを決定するためのロジックは、ユーザの家屋 2 1 1 及び／又はホスティングサービス 2 1 0 において具現化することができる。ユーザの家屋 2 1 1 において、クライアント装置 4 1 5 は、計算を行ってピークデータレートを決定し、そしてその情報をホスティングサービス 2 1 0 へ返送し、ホスティングサービス 2 1 0 において、ホスティングサービスのサーバー 4 0 2 は、クライアント 4 1 5 から受け取った統計情報（例えば、パケットロス、待ち時間、最大データレート、等）に基づいて、計算を行ってピークデータレートを決定する。

【0148】

図 9 a は、図 7 a、7 b 及び図 8 に示されて上述した循環 I / P タイル圧縮技術を使用して発生された実質的なシーンの複雑さ及び／又は動きを有するビデオストリームデータレート 9 3 4 を例示する。ビデオコンプレッサ 4 0 4 は、ピークデータレート 9 4 1 より低い平均データレートで圧縮ビデオを出力するように構成され、ほとんどの時間、ビデオストリームデータレートは、ピークデータレート 9 4 1 より低く保たれることに注意されたい。I / P / B 又は I / P フレームを使用して生成された図 6 c に示すビデオストリームデータレート 6 3 4 とデータレート 9 3 4 との比較は、循環 I / P タイル圧縮が非常に滑らかなデータレートを発生することを示している。更に、フレーム 2 x ピーク 9 5 2 （これは 2 x ピークデータレート 9 4 2 に接近する）及びフレーム 4 x ピーク 9 5 4 （これは 4 x ピークデータレート 9 4 4 に接近する）において、データレートは、ピークデータ

レート 941 を越え、受け容れられない。実際に、迅速に変化するビデオゲームからの高アクションビデオでも、ピークデータレート 941 を越えるピークがフレームの 2 % 未満で発生し、2 x ピークデータレート 942 を越えるピークがまれに発生し、そして 3 x ピークデータレート 943 を越えるピークはほとんど発生しない。しかし、それらが（例えば、シーンの移行中に）発生するときには、それらによって要求されるデータレートは、良好なクオリティのビデオ映像を発生するのに必要なものである。

【0149】

この問題を解決する 1 つの方法は、最大データレート出力がピークデータレート 941 となるようにビデオコンプレッサ 404 を単に構成することである。不都合なことに、それによりピークフレーム中に生じるビデオ出力のクオリティは、圧縮アルゴリズムがビットについて「欠乏」しているので、不十分である。その結果、突然の移行又は高速な動きがあるときに圧縮欠陥が現れ、やがて、ユーザは、突然の変化又は急速な動きがあるときに常に欠陥が現れ、それが極めて迷惑であることを悟る。

【0150】

人間の視覚系は、突然の変化又は急速な動きの間に現れる視覚上の欠陥には極めて敏感であるが、このような状態におけるフレームレートの減少の検出にはあまり敏感でない。実際に、このような突然の変化が生じたときに、人間の視覚系は、変化を追跡することに心を奪われ、フレームレートが短時間で 60 f p s から 30 f p s へ降下し、次いで、即座に 60 f p s へ復帰することに気付かないと思われる。そして、突然のシーンの切り換えのような非常に急激な移行の場合には、人間の視覚系は、フレームレートが 20 f p s 又は 15 f p s へ降下し、次いで、即座に 60 f p s へ戻ることには気付かない。観察する人間にとってフレームレートの減少が時々生じるだけである限り、ビデオは、60 f p s で連続的に動作したかのように見える。

【0151】

人間の視覚系のこの特性は、図 9 b に示す技術により利用される。サーバー 402（図 4 a 及び 4 b からの）は、一定のフレームレート（一実施形態では 60 f p s）の非圧縮ビデオ出力ストリームを発生する。時間線は、各フレーム 961-970 を各々 1 / 60 秒で示している。各非圧縮ビデオフレームは、フレーム 961 でスタートして、短待ち時間のビデオコンプレッサ 404 へ出力され、これは、フレームを 1 フレーム時間未満で圧縮し、第 1 フレームとして圧縮フレーム 1 981 を発生する。圧縮フレーム 1 981 に対して発生されるデータは、上述したように、多数のファクタに基づいて大きくてもよいし小さくてもよい。データが、フレーム時間（1 / 60 秒）内又はピークデータレート 941 未満でクライアント 415 へ送信できるに十分なほど小さい場合には、それが送信時間（x m i t 時間）991 中に送信される（矢印の長さは、送信時間の巾を示す）。次のフレーム時間において、サーバー 402 は、非圧縮フレーム 2 962 を発生し、これは、圧縮フレーム 2 982 へと圧縮され、そしてピークデータレート 941 においてフレーム時間未満である送信時間 992 中にクライアント 415 へ送信される。

【0152】

次いで、次のフレーム時間に、サーバー 402 は、非圧縮フレーム 3 963 を発生する。これがビデオコンプレッサ 404 により圧縮されると、それにより生じる圧縮フレーム 3 983 は、1 フレーム時間にピークデータレート 941 で送信できる以上のデータとなる。従って、それは、全フレーム時間と次のフレーム時間の一部分とを取り上げた送信時間（2 x ピーク）993 中に送信される。ここで、次のフレーム時間中に、サーバー 402 は、別の非圧縮フレーム 4 964 を発生し、それをビデオコンプレッサ 404 へ出力するが、データは、無視され、974 で示されている。これは、ビデオコンプレッサ 404 が、手前の圧縮フレームをまだ送信している間に到着する更に別の非圧縮ビデオフレームを無視するように構成されているからである。もちろん、クライアント 415 のビデオデコンプレッサは、フレーム 4 を受信せず、2 フレームの時間中、フレーム 3 をディスプレイ装置 422 に単に表示し続ける（即ち、フレームレートを 60 f p s から 30 f p s へ簡単に減少する）。

【0153】

次のフレーム5に対して、サーバー402は、非圧縮フレーム5 965を出力し、これは、圧縮フレーム5 985へと圧縮され、そして送信時間995中の1フレーム以内に送信される。クライアント415のビデオデコンプレッサは、フレーム5を解凍し、ディスプレイ装置422に表示する。次いで、サーバー402は、非圧縮フレーム6 966を出力し、ビデオコンプレッサ404は、それを圧縮フレーム6 986へと圧縮するが、このときには、得られるデータが非常に長い。圧縮フレームは、送信時間(4xピーク)996中にピークデータレート941で送信されるが、フレームを送信するのに、ほぼ4フレームの時間を要する。次の3フレームの時間中に、ビデオコンプレッサ404は、サーバー402からの3フレームを無視し、そしてクライアント415のデコンプレッサは、フレーム6を4フレームの時間中にディスプレイ装置422に確実に保持する(即ち、フレームレートを60fpsから15fpsへ簡単に減少する)。次いで、最終的に、サーバー402は、フレーム10 970を出力し、ビデオコンプレッサ404は、それを圧縮フレーム10 987へ圧縮し、これは、送信時間997中に送信され、そしてクライアント415のデコンプレッサは、フレーム10を解凍してディスプレイ装置422に表示し、そしてもう一度ビデオが60fpsで再開する。

10

【0154】

ビデオコンプレッサ404は、サーバー402により発生されたビデオストリームからビデオフレームをドロップするが、オーディオがどんな形態で到来するかに関わらずオーディオデータはドロップせず、ビデオフレームがドロップされるときにオーディオデータを圧縮し続けて、それらをクライアント415へ送信し、クライアントは、オーディオデータを解凍し続け、そしてオーディオを、ユーザにより使用される装置へ供給し、オーディオを再生することに注意されたい。従って、オーディオは、フレームがドロップされる期間中は非減衰状態を続ける。圧縮オーディオは、圧縮ビデオに比して、帯域巾の比較的僅かな割合を消費するだけであり、その結果、全体的なデータレートに大きな影響は及ぼさない。いずれのデータレート図にも示されていないが、ピークデータレート941内で圧縮オーディオストリームに対して常にデータレート容量が予約されている。

20

【0155】

図9bについて上述した例は、データレートピーク中にフレームレートがいかに降下するか示すために選択されたものであるが、先に述べた循環I/Pタイル技術が使用されるときに、そのようなデータレートピーク及びそれによりドロップされるフレームは、ビデオゲーム、動画及びあるアプリケーションソフトウェアにおいて生じるような高いシーン複雑さ/高いアクションのシーケンス中でもまれであることは、示されていない。従って、フレームレートの減少は、時々で且つ短時間であり、人間の視覚系は、それを検知しない。

30

【0156】

上述したフレームレート減少メカニズムが、図9aに示すビデオストリームデータレートに適用される場合に、それにより生じるビデオストリームデータレートが図9cに示されている。この例では、2xピーク952が平坦な2xピーク953へ減少され、4xピーク955が平坦な4xピーク955へ減少され、そして全ビデオストリームデータレート934がピークデータレート941以下に保たれる。

40

【0157】

従って、上述した技術を使用して、高アクションビデオストリームを、一般的なインターネット及び消費者グレードのインターネット接続を通して短い待ち時間で送信することができる。更に、LAN(例えば、100Mbpsイーサネット又は802.11gワイヤレス)、或いはプライベートネットワーク(例えば、データセンターとオフィスとの間の100Mbps接続)におけるオフィス環境では、高アクションビデオストリームをピークなしに送信することができ、大勢のユーザ(例えば、4.5Mbpsにおいて60fpsで1920x1080を送信する)が、ネットワーク又はネットワークスイッチバックプレーンを圧倒する重畳ピークをもたずに、LAN又は共有プライベートデータ接続を使

50

用できるようにする。

【0158】

データレートの調整

一実施形態では、ホスティングサービス210は、最初、利用可能な最大データレート622及びチャンネルの待ち時間にアクセスして、ビデオストリームのための適当なデータレートを決定し、次いで、それに応答してデータレートを動的に調整する。データレートを調整するために、ホスティングサービス210は、例えば、映像の解像度、及び／又はクライアント415へ送信されるビデオストリームのフレーム数／秒を変更することができる。又、ホスティングサービスは、圧縮ビデオのクオリティレベルを調整することもできる。ビデオストリームの解像度を、例えば、1280x720解像度から640x360へ変更するとき、クライアント415のビデオ解凍ロジック412は、ディスプレイスクリーン上に同じ映像サイズを維持するように映像をスケールアップすることができる。

10

【0159】

一実施形態では、チャンネルが完全にドロップアウトする状況において、ホスティングサービス210は、ゲームを休止する。マルチプレーヤゲームの場合には、ホスティングサービスは、ユーザがゲームからドロップアウトしたことを他のユーザに報告し、及び／又は他のユーザに対してゲームを休止する。

【0160】

ドロップ又は遅延したパケット

一実施形態において、図4a又は4bにおけるビデオコンプレッサ404とクライアント415との間のパケットロスのために、或いはパケットが順序ずれて受け取られて、解凍するには遅過ぎ且つ解凍フレームの待ち時間要件を満足するには遅過ぎる到着であるために、データが失われる場合に、ビデオ解凍ロジック412は、視覚上の欠陥を軽減することができる。ストリーミングI/Pフレーム具現化において、失われ／遅延したパケットがある場合には、スクリーン全体に影響が及び、潜在的に、ある期間中スクリーンを完全に凍結させるか、又は他のスクリーン中の視覚上の欠陥を表示させる。例えば、失われ／遅延したパケットがIフレームの損失を生じさせる場合には、デコンプレッサは、それに続く全てのPフレームに対する基準を、新たなIフレームが受け取られるまで欠くことになる。Pフレームが失われた場合には、それに続く全スクリーンのためのPフレームに影響を及ぼすことになる。Iフレームが現れるまでどれほど長いかに基づき、これは、長い又は短い視覚上の影響となる。図7a及び7bに示すインターリーブされたI/Pタイルを使用すると、失われ／遅延したパケットは、スクリーン全体に影響を及ぼすおそれが非常に僅かになる。というのは、影響を受けるパケットに含まれたタイルにしか影響しないからである。各タイルのデータが個々のパケット内で送信される場合には、パケットが失われた場合に、それが1つのタイルにしか影響しない。もちろん、視覚上の欠陥の中は、Iタイルパケットが失われるかどうか、及びPタイルが失われた場合には、Iタイルが現れるまでにどれほど多くのフレームを要するか、に依存する。しかし、スクリーン上の異なるタイルがIフレームで頻繁に（潜在的にフレームごとに）更新されるとすれば、スクリーン上の1つのタイルが影響を受ける場合でも、他のタイルは、そうではない。更に、ある事象が一度に多数のパケットのロスを生じさせる場合には（例えば、DSLラインに続く電力のスパイクでデータ流を短時間中断するもの）、あるタイルが、他のタイル以上に影響を受けるが、あるタイルは、新たなIタイルで素早く更新されるので、短時間影響を受けるだけである。又、ストリーミングI/Pフレーム具現化では、Iフレームが最も重要なフレームであるだけでなく、非常に大きなフレームでもあり、従って、ドロップされ／遅延されたパケットを生じさせる事象がある場合には、Iフレームが影響を受ける確率が、非常に小さなIタイルより高くなる（即ち、Iフレームが一部分でも失われた場合には、Iフレームを解凍できる見込みが全くなくなる）。これら全ての理由で、I/Pタイルを使用すると、パケットがドロップされ／遅延されるときに、視覚上の欠陥は、I/Pフレームの場合より遥かに少なくなる。

20

30

40

50

【0161】

一実施形態では、圧縮されたタイルをTCP（送信コントロールプロトコル）パケット又はUDP（ユーザデータグラムプロトコル）パケット内にインテリジェントにパッケージすることにより失われたパケットの作用を減少することが試みられる。例えば、一実施形態では、タイルが、可能であれば、パケット境界に整列される。図10aは、この特徴を具現化せずに、タイルを一連のパケット1001-1005内にどのようにパックするか示す。より詳細には、図10aにおいて、単一パケットのロスが複数のフレームのロスを生じるように、タイルがパケット境界に交差し、非効率的にパックされる。例えば、パケット1003または1004が失われる場合に、3つのタイルが失われ、視覚上の欠陥を招く。

10

【0162】

対称的に、図10bは、パケットロスの作用を減少するためにパケット内にタイルをインテリジェントにパッキングするためのタイルパッキングロジック1010を示す。第1に、タイルパッキングロジック1010は、タイルをパケット境界に整列させる。従って、タイルT1、T3、T4、T7及びT2は、各々、パケット1001-1005の境界に整列される。又、タイルパッキングロジックは、パケット境界に交差せずに、考えられる最も効率的な仕方、パケット内にタイルを嵌合させることも試みる。各タイルのサイズに基づいて、タイルT1及びT6は、1つのパケット1001内で合成され、T3及びT5は、1つのパケット1002内で合成され、タイルT4及びT8は、1つのパケット1003内で合成され、タイルT8は、パケット1004に追加され、そしてタイルT2は、パケット1005に追加される。従って、このスキームのもとでは、単一パケットのロスは、（図10aに示す3つのタイルではなく）2つ以下のタイルのロスを招くことになる。

20

【0163】

図10bに示す実施形態に対する1つの付加的な利益は、タイルが、映像内に表示されるのとは異なる順序で送信されることである。このように、隣接パケットが送信を妨げる同じ事象から失われた場合に、スクリーン上の互いに近隣でないエリアに影響を及ぼし、ディスプレイ上の欠陥をあまり目立たなくする。

【0164】

一実施形態では、順方向エラー修正（FEC）技術を使用して、ビデオストリームの幾つかの部分をチャンネルエラーから保護する。この技術で知られているように、リードソロモン及びビタビのようなFEC技術は、エラー修正データ情報を発生して、通信チャンネルを経て送信されるデータに添付させる。基礎的なデータ（例えば、Iフレーム）にエラーが発生した場合には、FECを使用して、エラーを修正することができる。

30

【0165】

FECコードは、送信のデータレートを高め、従って、理想的には、それらが最も必要とされるところに使用されるだけである。非常に目立つ視覚上の欠陥を生じないデータが送信される場合には、データを保護するためにFECコードを使用しないのが好ましい。例えば、失われるIタイルの直前のPタイルは、スクリーン上に1/60秒だけ視覚上の欠陥を生じさせる（即ち、スクリーン上のタイルは更新されない）。このような視覚上の欠陥は、人間の目でかろうじて検知できる。PタイルがIタイルから更に後方にある場合には、Pタイルを失うと、益々目立つことになる。例えば、タイルサイクルパターンが、Iタイルの後に15個のPタイルが続き、その後、Iタイルが再び得られるというものである場合、Iタイルの直後のPタイルが失われると、そのタイルは、15フレームの時間中誤った映像を示すことになる（60fpsでは、それは、250msである）。人間の目は、250ms間のストリームの中断は容易に検知する。従って、Pタイルが新たなIタイルから更に後方にあるほど（即ち、Pタイルがより接近してIタイルに続くほど）、欠陥がより目立ったものになる。上述したように、一般的に、Pタイルがより接近してIタイルに続くほど、そのPタイルに対するデータがより小さくなる。従って、Iタイルに続くPタイルは、失われないよう保護するのがより重要であるだけでなく、サイズが小さ

40

50

いことも重要である。そして、一般的に、保護される必要のあるデータが小さいほど、それを保護するために必要な F E C コードも小さくなる。

【0166】

従って、図 1 1 a に示すように、一実施形態では、ビデオストリームにおける I タイルの重要性のために、I タイルだけに F E C コードが与えられる。従って、F E C 1 1 0 1 は、I タイル 1 1 0 0 に対するエラー修正コードを含み、そして F E C 1 1 0 4 は、I タイル 1 1 0 3 に対するエラー修正コードを含む。この実施形態では、P タイルに対して F E C は発生されない。

【0167】

図 1 1 b に示す一実施形態では、失われた場合に視覚上の欠陥を生じさせるおそれが最もある P タイルについても F E C コードが発生される。この実施形態では、F E C 1 1 0 5 は、最初の 3 つの P タイルに対してエラー修正コードを与えるが、それに続く P タイルについては与えない。別の実施形態では、データサイズが最も小さい P タイルについて F E C コードが発生される（I タイルの後に最も早く発生して、保護することが最も重要な P タイルを自己選択する傾向がある）。

【0168】

別の実施形態では、F E C コードをタイルと共に送信するのではなく、タイルが 2 回、そのたびに異なるパケットで、送信される。1 つのパケットが失われ／遅延された場合には、他のパケットが使用される。

【0169】

図 1 1 c に示す一実施形態では、F E C コード 1 1 1 1 及び 1 1 1 3 は、ビデオと同時にホスティングサービスから送信されるオーディオパケット 1 1 1 0 及び 1 1 1 2 に対して各々発生される。ビデオストリームにおいてオーディオの完全性を維持することが特に重要である。というのは、歪んだオーディオ（例えば、カチカチ音又はスー音）は、特に望ましからぬユーザの経験を招くからである。F E C コードは、オーディオコンテンツがクライアントコンピュータ 4 1 5 において歪なくレンダリングされることを保証する上で助けとなる。

【0170】

別の実施形態では、F E C コードをオーディオデータと共に送信するのではなく、オーディオデータが 2 回、そのたびに異なるパケットで、送信される。1 つのパケットが失われ／遅延された場合には、他のパケットが使用される。

【0171】

更に、図 1 1 d に示す一実施形態では、クライアント 4 1 5 からホスティングサービス 2 1 0 へと上流に送信されるユーザ入力コマンド 1 1 2 0 及び 1 1 2 2（例えば、ボタン押圧）に対して F E C コード 1 1 2 1 及び 1 1 2 3 が各々使用される。これは重要である。というのは、ビデオゲーム又はアプリケーションにおいてボタン押圧又はマウス移動が欠落すると、望ましからぬユーザ経験を招くからである。

【0172】

別の実施形態では、F E C コードをユーザ入力コマンドデータと共に送信するのではなく、ユーザ入力コマンドデータが 2 回、そのたびに異なるパケットで、送信される。1 つのパケットが失われ／遅延された場合には、他のパケットが使用される。

【0173】

一実施形態では、ホスティングサービス 2 1 0 がクライアント 4 1 5 との通信チャンネルのクオリティを評価して、F E C を使用すべきかどうか決定し、もしそうであれば、ビデオ、オーディオ及びユーザコマンドのどの部分に F E C を適用すべきか決定する。チャンネルの「クオリティ」を評価することは、上述したように、パケットロス、待ち時間、等を評価するようなファンクションを含む。チャンネルに特に信頼性がない場合には、ホスティングサービス 2 1 0 は、I タイル、P タイル、オーディオ及びユーザコマンドの全てに F E C を適用することができる。対照的に、チャンネルに信頼性がある場合には、ホスティングサービス 2 1 0 は、オーディオ及びユーザコマンドのみに F E C を適用するか

、又はオーディオ又はビデオにF E Cを適用するか、或いはF E Cを全く使用しない。これら基礎的な原理に依然合致しながら、F E Cの適用の種々の他の順列を使用することができる。一実施形態では、ホスティングサービス210は、チャンネルの状態を常時監視し、それに応じてF E Cポリシーを変更する。

【0174】

別の実施形態では、図4a及び4bを参照すれば、パケットが失われ／遅延されてタイルデータを失うことになるか、又はおそらく特に悪いパケット損失のために、F E Cが、失われたタイルデータを修正できない場合には、クライアント415は、新たなIタイルが受け取られる前にどれほど多くのフレームが残っているか評価し、それをクライアント415からホスティングサービス210への往復待ち時間と比較する。その往復待ち時間が、新たなIタイルが当然到着するまでのフレームの数より少ない場合には、クライアント415は、新たなIタイルを要求するメッセージをホスティングサービス210へ送信する。このメッセージは、ビデオコンプレッサ404へルーティングされ、このコンプレッサは、データが失われたところのタイルに対してPタイルを発生するのではなく、Iタイルを発生する。図4a及び4bに示すシステムが、典型的に80ms未満の往復待ち時間を与えるように設計されているとすれば、それにより、タイルが80ms以内に修正される(60fpsにおいて、フレームは、16.67msの中であり、従って、全フレームの時間において、80msの待ち時間で、5フレームの時間である83.33ms以内にタイルが修正され、即ち、これは、目立った中断であるが、例えば、15フレームに対する250msの中断よりは遥かに目立たないものである)。コンプレッサ404がその通常循環順序からIタイルを発生するときに、Iタイルによりそのフレームの帯域巾が利用可能な帯域巾を越える場合は、コンプレッサ404は、他のタイルのサイクルを遅延させて、他のタイルがそのフレーム時間中にPタイルを受け取るようにし(1つのタイルが通常そのフレーム中に当然Iタイルであるべき場合でも)、次いで、次のフレームでスタートして、通常循環が続き、そして通常はその手前のフレームでIタイルを受信するタイルが、Iタイルを受信するようにする。このアクションは、Rフレーム循環の位相を短時間遅延するが、通常は、視覚上、目立つものではない。

【0175】

ビデオ及びオーディオコンプレッサ／デコンプレッサ具現化

図12は、マルチコア及び／又はマルチプロセッサ1200を使用して8枚のタイルを並列に圧縮する1つの特定の実施形態を示す。一実施形態では、2.66GHz以上で動作するデュアルプロセッサ・クオッドコアXeon CPUコンピュータシステムが使用され、各コアは、オープンソースx264 H.264コンプレッサを独立プロセスとして具現化する。しかしながら、これら基礎的な原理に依然合致しながら、種々の他のハードウェア／ソフトウェアコンフィギュレーションが使用されてもよい。例えば、CPUコアの各々を、FPGAで具現化されるH.264コンプレッサに置き換えることができる。図12に示す例では、コア1201-1208を使用して、Iタイル及びPタイルを8個の独立したスレッドとして同時に処理する。この技術で良く知られているように、現在のマルチコア及びマルチプロセッサコンピュータシステムは、本来、Microsoft Windows XPプロフェッショナルエディション(64ビット又は32ビットエディション)及びLinuxのようなマルチスレッドオペレーティングシステムと一体化されたときにマルチスレッド可能である。

【0176】

図12に示す実施形態では、8個のコアの各々は、1つのタイルについてのみ役立つので、x264の個別インスタンス生成を各々実行する他のコアとは主として独立して動作する。PCI Express x1ベースのDVI捕獲カード、例えば、オランダ、オーステルハウトのマイクロエレクトロニクスからのセンデロ・ビデオ・イメージングIPデベロップメント・ボードを使用して、640x480、800x600、又は1280x720解像度で非圧縮ビデオを捕獲し、そしてこのカードのFPGAは、直接メモリアクセス(DMA)を使用して、DVIバスを経て捕獲されたビデオをシステムRAMへと転送す

る。タイルは、 4×2 配列 1 2 0 5 に配列される（それらは、方形タイルとして示されているが、この実施形態では、 160×240 解像度である）。 $x264$ の各インスタンス化は、8 個の 168×240 タイルの 1 つを圧縮するように構成され、そしてそれらは、最初の I タイル圧縮の後に、各コアがサイクルに入り、各 1 つのフレームが他のフレームと位相ずれし、1 つの I タイル、その後、7 つの P タイルを圧縮するように、図 1 2 に示すように、同期される。

【0177】

フレーム時間ごとに、得られた圧縮されたタイルが、上述した技術を使用して、パケットストリームへと合成され、次いで、その圧縮されたタイルが、行先クライアント 4 1 5 へ送信される。

【0178】

図 1 2 には示されていないが、合成される 8 枚のタイルのデータレートが特定のピークデータレート 9 4 1 を越える場合には、その合成される 8 枚のタイルのデータが送信されてしまうまで必要に応じて多数のフレームの時間中、全部で 8×264 のプロセスが保留される。

【0179】

一実施形態では、クライアント 4 1 5 は、F F m p e g の 8 個のインスタンスを実行する P C においてソフトウェアとして具現化される。受信プロセスは、8 枚のタイルを受信し、そして各タイルは、F F m p e g ヘルパーティングされ、これは、タイルを解凍して、ディスプレイ装置 4 2 2 上の適当なタイル位置にレンダリングする。

【0180】

クライアント 4 1 5 は、P C の入力装置ドライバからキーボード、マウス、又はゲームコントローラ入力を受け取り、そしてそれをサーバー 4 0 2 へ送信する。サーバー 4 0 2 は、次いで、その受け取った入力装置データを、サーバー 4 0 2 で実行されるゲーム又はアプリケーションに適用し、サーバー 4 0 2 は、I n t e l 2 . 1 6 G H z C o r e D u o C P U を使用して W i n d o w s を実行する P C である。次いで、サーバー 4 0 2 は、新たなフレームを発生し、そしてその D V I 出力を通して、マザーボードベースのグラフィックシステムから、或いは N V I D I A 8 8 0 0 G T X P C I エクスプレスカードの D V I 出力を通して、それを出力する。

【0181】

同時に、サーバー 4 0 2 は、ゲーム又はアプリケーションにより形成されたオーディオを、そのデジタルオーディオ出力（例えば、S / P D I F）を通して出力し、これは、ビデオ圧縮を具現化するデュアルクオッドコア X e o n ベース P C のデジタルオーディオ入力に結合される。V o r b i s オープンソースオーディオコンプレッサは、プロセススレッドに対して利用可能なコアを使用して、オーディオをビデオと同時に圧縮するように使用される。一実施形態では、タイルの圧縮を完了するコアが、オーディオ圧縮を最初に実行する。圧縮されたオーディオは、次いで、圧縮されたビデオと共に送信され、そしてクライアント 4 1 5 において V o r b i s オーディオデコンプレッサを使用して解凍される。

【0182】

ホスティングサービスサーバーセンター配布

光ファイバのようなガラスを通る光は、真空中の光の速度の一部分で進行し、従って、光ファイバにおける光の厳密な伝播速度を決定することができる。しかし、実際には、ルーティング遅延の時間、送信の非効率さ、及び他のオーバーヘッドを考慮すると、インターネット上の最適な待ち時間は、光速のほぼ 5 0 % の送信速度を反映することが観察される。従って、最適な 1 0 0 0 マイル往復待ち時間は、約 2 2 m s であり、そして最適な 3 0 0 0 マイル往復待ち時間は、約 6 4 m s である。従って、米国の一方の海岸にある単一のサーバーは、（3 0 0 0 マイル程度離れた）他方の海岸にあるクライアントに望ましい待ち時間でサービスするには離れ過ぎている。しかしながら、図 1 3 a に示すように、ホスティングサービス 2 1 0 のサーバーセンター 1 3 0 0 が、米国の中央部（例えば、カンザス、ネブラスカ、等）に位置されて、米国大陸の任意の点までの距離が約 1 5 0 0 マイ

10

20

30

40

50

ル以下となる場合には、往復インターネット待ち時間を32ms程度に小さくすることができる。図4bを参照すれば、ユーザISP453に許された最悪のケースの待ち時間は、典型的に、25msであるが、DSL及びケーブルモデムシステムでは、10から15msに近い待ち時間が観察されていることに注意されたい。又、図4bは、ユーザの家屋211からホスティングセンター210までの最大距離が1000マイルであると仮定している。従って、15msの典型的なユーザISP往復待ち時間が使用され、32msの往復待ち時間に対して1500マイルの最大インターネット距離では、ユーザが入力装置421を操作しそしてディスプレイ装置422上に応答を見る点からの合計往復待ち時間は、 $1+1+15+32+1+16+6+8=80\text{ms}$ となる。従って、80msの応答時間が、典型的に、1500マイルのインターネット距離にわたって得られる。これは、

10

【0183】

図13bに示した別の実施形態では、ホスティングサービス210のサーバーセンターHS-1ないしHS6が、米国（又は他の地理的領域）に戦略的に配置され、ある大きなホスティングサービスサーバーセンター（例えば、HS2及びHS5）は、人口の多いセンターの付近に配置される。一実施形態では、サーバーセンターHS1ないしHS6は、インターネット又はプライベートネットワーク或いはその両方の組合せであるネットワーク1301を経て情報を交換する。複数のサーバーセンターがある状態では、長いユーザISP待ち時間453をもつユーザに対して短い待ち時間でサービスを提供することができる。

20

【0184】

インターネット上の距離は、確実に、インターネットを通しての往復待ち時間に貢献するファクタであるが、時々、待ち時間にあまり関係のない他のファクタが影響するようになる。時々、パケットストリームがインターネットを通して離れた位置へルーティングされそして再び戻り、長いループから待ち時間を招く。時々、適切に動作しないルーティング装置が経路上にあり、送信遅延を招く。時々、経路に過負荷となるトラフィックがあって、遅延を導入する。そして、時々、ユーザのISPが所与の行先へルーティングされるのを全く阻止する故障が生じる。従って、一般的なインターネットは、通常、距離により主として決定される非常に信頼性のある最適なルート及び待ち時間で（特に、ユーザのローカルエリアの外部へのルートを生じさせる長距離接続で）、ある点から別の点への接続を与えるが、このような信頼性及び待ち時間は、保証されるものではなく、多くの場合、ユーザの家屋から一般的なインターネット上の所与の行先まで得ることができない。

30

【0185】

一実施形態では、ユーザクライアント415が、最初にホスティングサービス210に接続して、ビデオゲームをプレイするか又はアプリケーションを使用するときに、クライアントは、始動時に利用可能なホスティングサービスサーバーセンターHS1ないしHS6の各々と通信する（例えば、上述した技術を使用して）。特定の接続に対して待ち時間が充分短い場合には、その接続が使用される。一実施形態では、クライアントは、最短待ち時間接続をもつものが選択されたところのホスティングサービスサーバーセンターの全部又はサブセットと通信する。クライアントが最短待ち時間接続をもつサービスセンターを選択してもよいし、又はサービスセンターが最短的時間接続をもつものを識別しそしてこの情報を（例えば、インターネットアドレスの形態で）クライアントに与えてもよい。

40

【0186】

特定のホスティングサービスサーバーセンターが過負荷状態になり及び／又はユーザのゲーム又はアプリケーションが、別の負荷の少ないホスティングサービスサーバーセンターへの待ち時間を許容できる場合には、クライアント415は、他のホスティングサービスサーバーセンターに向け直される。このような状況では、ユーザが実行しているゲーム又はアプリケーションは、ユーザの過負荷状態にサーバーセンターにおけるサーバー402上で休止され、そしてゲーム又はアプリケーションの状態データは、別のホスティング

50

サービスサーバーセンターにおけるサーバー 402 へ転送される。次いで、ゲーム又はアプリケーションが再開される。一実施形態では、ホスティングサービス 210 は、ゲーム又はアプリケーションが自然の休止点（例えば、ゲームにおけるレベル間、又はユーザがアプリケーションにおいて「セーブ」オペレーションを開始した後）に到達して転送を行うまで、待機する。更に別の実施形態では、ホスティングサービス 210 は、ユーザのアクティビティが特定の期間中（例えば 1 分間）休止するまで待機し、次いで、その時点で転送を開始する。

【0187】

上述したように、一実施形態では、ホスティングサービス 210 は、そのクライアントに保証された待ち時間を与えるよう試みるために、図 14 のインターネットバイパスサービス 440 に契約する。ここで使用されるインターネットバイパスサービスとは、保証された特性（例えば、待ち時間、データレート、等）をもつインターネットにおいてある点から別の点へのプライベートネットワークルートを与えるサービスである。例えば、ホスティングサービス 210 が、AT&T のサンフランシスコベースの中央オフィスへのルーティングではなく、サンフランシスコで提供される AT&T の DSL サービスを利用してユーザから大量のトラフィックを受信した場合に、ホスティングサービス 210 は、サンフランシスコベースの中央オフィスと、ホスティングサービス 210 のための 1 つ以上のサービスセンターとの間のサービスプロバイダー（おそらく AT&T それ自体又は別のプロバイダー）から大容量のプライベートデータ接続をリースすることができる。次いで、全てのホスティングサービスサーバーセンター H S 1 ないし H S 6 から一般的なインターネットを経て AT&T の DSL を使用するサンフランシスコのユーザへのルートが非常に長い待ち時間を生じる場合には、それに代わって、プライベートデータ接続を使用することができる。プライベートデータ接続は、一般的に、一般的なインターネットを通るルートより費用が高いが、それがユーザへのホスティングサービス 210 接続の僅かな割合に保たれる限り、全コストへの影響は低く、ユーザは、より一貫したサービスを経験することになる。

【0188】

サービスセンターは、多くの場合、停電時に 2 層のバックアップ電源を有する。第 1 の層は、典型的に、バッテリーから（又は発電機に取り付けられてランニング状態に保たれるフライホイールのような別の直ちに利用できるエネルギー源から）のバックアップ電源であって、主電源が停電したときに直ちに電力を供給して、サーバーセンターをランニング状態に保持する。停電が短時間で、主電源が素早く（例えば、1 分以内に）回復する場合には、バッテリーは、全て、サーバーセンターをランニング状態に保持するために必要とされる。しかし、停電が長期間である場合には、典型的に、発電機（例えば、ジーゼル付勢）が始動され、バッテリーを引き継いで、燃料がある限り運転することができる。このような発電機は、サーバーセンターが通常主電源から得るのと同程度の電力を発生できねばならないので、非常に高価である。

【0189】

一実施形態では、ホスティングサービス H S 1 ないし H S 5 の各々は、互いにユーザデータを共有し、1 つのサーバーセンターが停電した場合に、進行中のゲーム及びアプリケーションを休止し、次いで、ゲーム又はアプリケーションの状態データを各サーバー 402 から他のサーバーセンターのサーバー 402 へ転送し、そして各ユーザのクライアント 415 に通知して、新たなサーバー 402 と通信するように指令する。このような状態が頻繁に生じないとすれば、最適な待ち時間を与えることのできないホスティングサービスサーバーセンターへのユーザの転送を受け容れることができ（即ち、ユーザは、単に停電期間中に長い待ち時間を許容するだけでよく）、これは、ユーザを転送するための非常に広範囲のオプションを許す。例えば、米国を横切る時間ゾーン差が与えられると、東海岸のユーザは、PM 11 時 30 分には就寝するが、PM 8 時 30 分の西海岸のユーザは、ビデオゲームの使用がピークになり始めている。その時間に西海岸のホスティングサービスサーバーセンターに停電が生じた場合に、他のホスティングサービスサーバーセンターに

は、全てのユーザを取り扱うに十分な西海岸サーバー４０２が存在しないことがある。このような状況では、利用できるサーバー４０２を有する東海岸のホスティングサービスサーバーセンターへあるユーザを転送することができ、それらユーザに対してのみ長い待ち時間となる。ユーザが、停電したサーバーセンターから転送されると、サーバーセンターは、次いで、そのサーバー及び装置の整然とした遮断を開始し、バッテリー（又は他の即座の電源バックアップ）が尽きるまでに全ての装置を遮断することができる。このように、サーバーセンター用の発電機のコストを回避することができる。

【０１９０】

一実施形態では、ホスティングサービス２１０の激しい負荷の時間中に（ピークユーザ負荷によるか、又は１つ以上のサーバーセンターが停電したために）、ユーザは、彼等が使用しているゲーム又はアプリケーションの待ち時間要件に基づいて他のサーバーセンターへ転送される。従って、短待ち時間を要求するゲーム又はアプリケーションを使用しているユーザは、電源に制限があるときに、利用可能な短待ち時間サーバー接続への優先権が与えられる。

【０１９１】

ホスティングサービスの特徴

図１５は、以下の特徴説明に使用されるホスティングサービス２１０のためのサーバーセンターのコンポーネントの実施形態を示す。図２ａに示されたホスティングサービス２１０と同様に、サーバーセンターのコンポーネントは、特に指示のない限り、ホスティングサービス２１０の制御システム４０１により制御され整合される。

【０１９２】

ユーザクライアント４１５からのインバウンドインターネットトラフィック１５０１は、インバウンドルーティング１５０２に向けられる。典型的に、インバウンドインターネットトラフィック１５０１は、インターネットへの高速光ファイバ接続を経てサーバーセンターに入るが、適当な帯域巾、信頼性及び短待ち時間のネットワーク接続手段で充分である。インバウンドルーティング１５０２は、ネットワーク（ネットワークは、イーサネットネットワーク、ファイバチャネルネットワークとして、或いは他の搬送手段を経て具現化できる）スイッチ、及びスイッチをサポートするルーティングサーバーのシステムであり、これは、到着するパケットを取り上げ、そして各パケットを適当なアプリケーション／ゲーム（app/game）サーバー１５２１－１５２５へルーティングする。一実施形態では、特定のapp/gameサーバーに配送されるパケットは、クライアントから受信されたデータのサブセットを表し、及び／又はデータセンター内の他のコンポーネント（例えば、ゲートウェイ及びルーターのようなネットワークコンポーネント）によって変換／変更されてもよい。あるケースでは、例えば、ゲーム又はアプリケーションが複数のサーバー上で並列に一度実行される場合に、パケットが一度に２つ以上のサーバー１５２１－１５２５へルーティングされる。RAIDアレイ１５１１－１５１２は、インバウンドルーティングネットワーク１５０２へ接続され、app/gameサーバー１５２１－１５２５がそれらRAIDアレイ１５１１－１５１２の読み取り及び書き込みを行えるようにする。更に、RAIDアレイ１５１５（複数のRAIDアレイとして具現化できる）も、インバウンドルーティング１５０２に接続され、RAIDアレイ１５１５からのデータは、app/gameサーバー１５２１－１５２５から読み出すことができる。インバウンドルーティング１５０２は、インバウンドインターネットトラフィック１５０１を根にもつスイッチのツリー構造、全ての種々の装置を相互接続するメッシュ構造、或いは相互通信装置間の集中トラフィックが他の装置間の集中トラフィックから分離されるようにして相互接続された一連のサブネットを含めて、広範囲な従来のネットワークアーキテクチャーで具現化することができる。ネットワークコンフィギュレーションの一形式は、SANであり、これは、典型的に記憶装置として使用されるが、装置間の一般的な高速データ転送に使用することもできる。又、app/gameサーバー１５２１－１５２５は、各々、インバウンドルーティング１５０２への複数のネットワーク接続を有する。例えば、サーバー１５２１－１５２５は、RAIDアレイ１５１１－１５１２に取り付け

られたサブネットへのネットワーク接続と、他の装置に取り付けられたサブネットへの別のネットワーク接続とを有することができる。

【0193】

app/gameサーバー1521-1525は、図4aに示す実施形態におけるサーバー402に関して上述したように、全部が同じに、幾つかが異なるように、又は全部が異なるように構成されてもよい。一実施形態では、各ユーザは、ホスティングサービスを利用するときに、典型的に、少なくとも1つのapp/gameサーバー1521-1525を使用する。説明を簡略化するため、所与のユーザがapp/gameサーバー1521を使用すると仮定するが、1人のユーザが複数のサーバーを使用することができると共に、複数のユーザが単一のapp/gameサーバー1521-1525を共有することができ
10
る。上述したようにクライアント415から送られるユーザ制御入力は、インバウンドインターネットトラフィック1501として受け取られ、そしてインバウンドルーティング1502を通してapp/gameサーバー1521へルーティングされる。app/gameサーバー1521は、ユーザの制御入力を、サーバー上で実行されるゲーム又はアプリケーションへの制御入力として使用し、そしてビデオ及びそれに関連したオーディオの次のフレームを計算する。次いで、app/gameサーバー1521は、非圧縮のビデオ/オーディオ1529を共有ビデオ圧縮1530へ出力する。app/gameサーバーは、非圧縮のビデオを、1ギガビット以上のイーサネット接続を含む手段を経て出力できるが、一実施形態では、ビデオがDVI接続を経て出力され、そしてオーディオ及び他の圧縮並びに通信チャンネル状態情報は、ユニバーサルシリアルバス（USB）
20
接続を経て出力される。

【0194】

共有ビデオ圧縮1530は、app/gameサーバー1521ないし1525からの非圧縮ビデオ及びオーディオを圧縮する。圧縮は、完全にハードウェアで具現化されてもよいし、又はハードウェア実行ソフトウェアで具現化されてもよい。各app/gameサーバー1521ないし1525に対する専用コンプレッサがあってもよいし、又はコンプレッサが充分高速である場合には、所与のコンプレッサを使用して、2つ以上のapp/gameサーバー1521ないし1525からのビデオ/オーディオを圧縮することができ
30
る。例えば、60fpsでは、ビデオフレーム時間は、16.67msである。コンプレッサがフレームを1msで圧縮できる場合には、そのコンプレッサを使用して、16個程度のapp/gameサーバー1521ないし1525からのビデオ/オーディオを圧縮することができ、これは、あるサーバーからの入力を別のサーバーの後に取り上げ、コンプレッサが各ビデオ/オーディオ圧縮プロセスの状態をセーブし、そしてコンテキストを、それがサーバーからのビデオ/オーディオストリーム間で循環するときにスイッチすることにより行われる。その結果、圧縮ハードウェアにおいて実質的なコスト節約が得られる。異なるサーバーが異なる時間にフレームを完了するので、一実施形態では、コンプレッサリソースが、各圧縮プロセスの状態を記憶するための共有記憶手段（例えば、RAM、フラッシュ）と共に共有プール1530にあり、サーバー1521-1525のフレームが完了して圧縮の準備ができると、制御手段は、どの圧縮リソースがそのとき利用できるか決定し、圧縮リソースに、サーバーの圧縮プロセスの状態と、圧縮すべき非圧縮
40
ビデオ/オーディオのフレームとを与える。

【0195】

各サーバーの圧縮プロセスの状態の一部分は、圧縮それ自体に関する情報、例えば、Pタイルの基準として使用できる以前のフレームの解凍フレームバッファデータ、ビデオ出力の解像度；圧縮のクオリティ；タイル構造；タイル当たりのビットの割り当て；圧縮のクオリティ、オーディオフォーマット（例えば、ステレオ、サラウンドサウンド、Dolby（登録商標）、AC-3）を含むことに注意されたい。しかし、圧縮プロセスの状態は、ピークデータレート941に関する通信チャンネル状態情報、以前のフレーム（図9bに示す）が現在出力されるかどうか（その結果、現在フレームを無視すべきかどうか）、そして潜在的に、圧縮に考慮すべきチャンネル特性、例えば、圧縮の判断に影響する過
50

剰なパケットロス（例えば、I タイルの頻度、等に関して）があるかどうか、も含む。クライアント 4 1 5 から送られるデータを監視する各ユーザをサポートする app / game サーバー 1 5 2 1 - 1 5 2 5 により決定されるように、ピークデータレート 9 4 1 又は他のチャンネル特性が時間と共に変化するとき、app / game サーバー 1 5 2 1 - 1 5 2 5 は、共有ハードウェア圧縮 1 5 3 0 へ関連情報を送信する。

【0 1 9 6】

又、共有ハードウェア圧縮 1 5 3 0 は、上述したような手段を使用して、圧縮されたビデオ／オーディオをパケット化し、そしてもし適当であれば、F E C コードを適用し、あるデータを複写し、又は他のステップを行って、ビデオ／オーディオデータストリームをクライアント 4 1 5 により受信しそしてできるだけ高いクオリティ及び信頼性で解凍するという能力を十分に確保するようにする。

10

【0 1 9 7】

以下に述べるような幾つかのアプリケーションでは、所与の app / game サーバー 1 5 2 1 - 1 5 2 5 のビデオ／オーディオ出力を同時に複数の解像度で（又は他の複数のフォーマットで）得られることが要求される。従って、app / game サーバー 1 5 2 1 - 1 5 2 5 が共有ハードウェア圧縮 1 5 3 0 のリソースを通知する場合には、その app / game サーバー 1 5 2 1 - 1 5 2 5 の非圧縮ビデオ／オーディオ 1 5 2 9 が、異なるフォーマット、異なるリソース、及び／又は異なるパケット／エラー修正構造で同時に圧縮される。あるケースでは、幾つかの圧縮リソースを、同じビデオ／オーディオを圧縮する複数の圧縮プロセス間で共有することができる（例えば、多くの圧縮アルゴリズムには、圧縮を適用する前に映像を複数のサイズにスケーリングするステップがある。異なるサイズの映像を出力することが要求される場合に、このステップを使用して、一度に多数の圧縮プロセスに応じることができる）。他のケースでは、フォーマットごとに個別の圧縮リソースが要求される。いずれにせよ、所与の app / game サーバー 1 5 2 1 - 1 5 2 5 に要求される全ての種々の解像度及びフォーマットの圧縮されたビデオ／オーディオ 1 5 3 9 は、（1 つであっても多数であっても）アウトバウンドルーティング 1 5 4 0 へ一度に出力される。一実施形態では、圧縮されたビデオ／オーディオ 1 5 3 9 の出力は U D P フォーマットであり、従って、パケットの位置方向性ストリームである。

20

【0 1 9 8】

アウトバウンドルーティングネットワーク 1 5 4 0 は、圧縮された各ビデオ／オーディオストリームを、アウトバウンドインターネットトラフィック 1 5 9 9 インターフェイス（これは、典型的に、インターネットへのファイバインターフェイスに接続される）を通して意図されたユーザ（1 人又は複数）又は他の行先へ向け、及び／又は遅延バッファ 1 5 1 5 へ戻し、及び／又はインバウンドルーティング 1 5 0 2 へ戻し、及び／又はビデオ配布のためにプライベートネットワーク（図示せず）を通して向ける一連のルーティングサーバー及びスイッチを備えている。（以下に述べるように）アウトバウンドルーティング 1 5 4 0 は、所与のビデオ／オーディオストリームを一度に複数の行先へ出力することに注意されたい。一実施形態では、これは、一度に複数の行先へストリーミングされることが意図された所与の U D P ストリームをブロードキャストし、そしてこのブロードキャストをアウトバウンドルーティング 1 5 4 0 内のルーティングサーバー及びスイッチにより繰り返すようなインターネットプロトコル（I P）マルチキャストを使用して具現化される。ブロードキャストの複数の行先は、インターネットを経て複数のユーザのクライアント 4 1 5、インバウンドルーティング 1 5 0 2 を経て複数の app / game サーバー 1 5 2 1 - 1 5 2 5、及び／又は 1 つ以上の遅延バッファ 1 5 1 5 である。従って、所与のサーバー 1 5 2 1 - 1 5 2 2 の出力は、1 つ又は複数のフォーマットへ圧縮され、そして各圧縮されたストリームは、1 つ又は複数の行先へ向けられる。

30

40

【0 1 9 9】

更に、別の実施形態では、複数の app / game サーバー 1 5 2 1 - 1 5 2 5 が 1 人のユーザにより同時に使用され（例えば、複雑なシーンの 3 D 出力を生成するために並列処理構成で）そして各サーバーが出来上がる映像の一部分を発生する場合には、複数のサ

50

ーバー 1 5 2 1－1 5 2 5 のビデオ出力を共有ハードウェア圧縮 1 5 3 0 により合成フレームへと合成することができ、そしてその点以降は、それが単一の a p p / g a m e サーバー 1 5 2 1－1 5 2 5 から到来したかのように、上述したように取り扱われる。

【0 2 0 0】

一実施形態では、a p p / g a m e サーバー 1 5 2 1－1 5 2 5 により発生される全ビデオのコピー（少なくともユーザが見るビデオの解像度又はそれ以上）が、少なくとも数分（一実施形態では、1 5 分）、遅延バッファ 1 5 1 5 に記録されることに注意されたい。これは、各ユーザが、以前の働き又は成績（ゲームの場合）を再検討するために各セッションからビデオを「巻き戻し」できるようにする。従って、一実施形態では、ユーザクライアント 4 1 5 ヘルディングされる圧縮された各ビデオ／オーディオ出力 1 5 3 9 のストリームは、遅延バッファ 1 5 1 5 へもマルチキャストされる。ビデオ／オーディオが遅延バッファ 1 5 1 5 に記憶されるときには、遅延バッファ 1 5 1 5 のディレクトリが、遅延されたビデオ／オーディオのソースである a p p / g a m e サーバー 1 5 2 1－1 5 2 5 のネットワークアドレスと、遅延されたビデオ／オーディオを見つけることのできる遅延バッファ 1 5 1 5 上の位置との間の相互参照を与える。

【0 2 0 1】

生の、瞬時に見られる、瞬時にプレイ可能なゲーム

a p p / g a m e サーバー 1 5 2 1－1 5 2 5 は、ユーザのための所与のアプリケーション又はビデオゲームを実行するのに使用できるだけでなく、ホスティングサービス 2 1 0 によるナビゲーション及び他の特徴をサポートするホスティングサービス 2 1 0 のためのユーザインターフェイスアプリケーションを生成するのにも使用できる。1つのこのようなユーザインターフェイスアプリケーションのスクリーンショットが図 1 6 に示されており、即ち「ゲームファインダー」スクリーンである。この特定のユーザインターフェイススクリーンは、ユーザが、他のユーザにより生で（又は遅延されて）プレイされている 1 5 のゲームを見ることができるようになる。1 6 0 0 のような「サムネール」ビデオウィンドウの各々は、動きのある生のビデオウィンドウで、1 人のユーザのゲームからのビデオを示している。サムネールに示される視野は、ユーザが見ている同じ視野でもよいし、遅延された視野でもよい（例えば、ユーザがコンバットゲームをプレイする場合、ユーザは、自分がどこに隠れているか他のユーザから見えるのを望まず、又、ゲームプレイの視野を、ある期間、例えば、1 0 分遅らせることを選択してもよい）。又、視野は、ユーザの視野とは異なるゲームのカメラ視野でもよい。メニュー選択肢（この図には示さず）を通して、ユーザは、種々の基準に基づいて、一度に見るゲームの選択を選ぶことができる。例示的選択の小さな例として、ユーザは、（図 1 6 に示すような）ゲームのランダムな選択、（異なるプレーヤにより全てプレイされる）一種類の全てのゲーム、ゲームのトップランクプレーヤのみ、ゲームの所与のレベルのプレーヤ、低いランクのプレーヤ（例えば、プレーヤが基礎を学習している場合）、「相棒」（又はライバル）であるプレーヤ、視聴者の最も多いゲーム、等を選ぶことができる。

【0 2 0 2】

一般的に、各ユーザは、自分のゲーム又はアプリケーションからのビデオが他人に見られるかどうか判断し、もしそうであれば、どんな他人、いつ他人に見られるか、遅れて見られるだけかどうか、判断することに注意されたい。

【0 2 0 3】

図 1 6 に示すユーザインターフェイススクリーンを発生する a p p / g a m e サーバー 1 5 2 1－1 5 2 5 は、ゲームが要求される各ユーザについて a p p / g a m e サーバー 1 5 2 1－1 5 2 5 へメッセージを送信することにより 1 5 個のビデオ／オーディオフィードを取得する。このメッセージは、インバウンドルーティング 1 5 0 2 又は別のネットワークを通して送信される。このメッセージは、要求されたビデオ／オーディオのサイズ及びフォーマットを含み、ユーザインターフェイススクリーンを見ているユーザを識別する。所与のユーザは、「プライバシー」モードを選択し、他のユーザが彼のゲームのビデオ／オーディオを（彼の視点又は別の視点から）見るのを許さないことを選ぶか、或いは

前記段落で述べたように、ユーザは、彼のゲームからのビデオ／オーディオを見るのを許すが、見るビデオ／オーディオを遅らせることを選んでもよい。ビデオ／オーディオを見ることを許すための要求を受け取って受け容れるユーザ app/game サーバー 1521-1525 は、要求側サーバーに確認を送り、共有ハードウェア圧縮 1530 に、要求されたフォーマット又はスクリーンサイズで付加的な圧縮されたビデオストリームを発生する要求も通知し（フォーマット及びスクリーンサイズが、既に発生されたものとは異なると仮定すれば）、そしてその圧縮されたビデオの行先（即ち、要求側サーバー）も指示する。要求されたビデオ／オーディオが遅延されるだけである場合には、要求側 app/game サーバー 1521-1525 は、そのように通知され、そして遅延バッファ 1515 のディレクトリにおけるビデオ／オーディオの位置と、遅延されたビデオ／オーディオのソースである app/game サーバー 1521-1525 のネットワークアドレスとをルックアップすることにより、遅延バッファ 1515 から遅延されたビデオ／オーディオを取得する。これらの要求が全て発生されて処理されると、15までの生のサムネールサイズのビデオストリームが、アウトバウンドルーティング 1540 から、app/game サーバー 1521-1525 へのインバウンドルーティング 1502 ヘルディングされて、ユーザインターフェイススクリーンを発生し、そしてサーバーにより解凍されて表示される。遅延されたビデオ／オーディオストリームは、スクリーンサイズが大き過ぎることがあり、もしそうであれば、app/game サーバー 1521-1525 は、ストリームを解凍して、ビデオストリームをサムネールサイズへスケールダウンする。一実施形態では、オーディオ／ビデオのための要求が、図 4a のホスティングサービス制御システムと同様の中央「管理」サービス（図 15 には示さず）へ送られ（及びそれにより管理され）、これは、次いで、その要求を適当な app/game サーバー 1521-1525 へ向け直す。更に、一実施形態では、サムネールが、それを許すユーザのクライアントへ「プッシュ」されるので、要求が発せられないこともある。

【0204】

全て同時に混合される 15 個のゲームからのオーディオは、不快な音を発することがある。ユーザは、このように全ての音を一緒に混合することを選んでもよいし（おそらく、見ている全てのアクションにより生じる騒音の感覚を単に得るために）、又はユーザは、一度に 1 つのゲームからのオーディオのみを聞くことを選んでもよい。単一ゲームの選択は、黄色の選択ボックス 1601（図 16 の白黒レンダリングでは、黒い長方形の輪郭として現われる）を所与のゲームへ移動することにより達成される（黄色のボックスの移動は、キーボード上の矢印キーを使用するか、マウスを移動するか、ジョイスティックを動かすか、又は移動電話のような別の装置の方向ボタンを押すことにより達成できる）。単一のゲームが選択されると、そのゲームからのオーディオだけが再生される。又、ゲーム情報 1602 も示される。このゲームの場合には、例えば、発行者ロゴ（例えば、「電子アートについては、“EA”）及びゲームロゴ、例えば、“Need for Speed Carbon”、及びオレンジの水平バー（図 16 では、縦縞をもつバーとしてレンダリングされる）は、その特定の瞬間にゲームをプレイするか又は見ている人々の数を相対的な表現で指示する（この場合は、多数、従って、ゲームは、「ホット」）。更に、「スタツ(Stats)」（即ち、統計値）も設けられ、これは、145 人のプレーヤが Need for Speed Game の 80 個の異なるインスタンス化を能動的にプレイしており（即ち、個人プレーヤゲーム又はマルチプレーヤゲームのいずれかによりプレイできる）、そして 680 人の視聴者がいる（このユーザは、そのうちの 1 人である）ことを示している。これらの統計値（及び他の統計値）は、ホスティングサービス 210 のオペレーションのログを保持すると共に、ユーザに適宜請求し、又、コンテンツを提供する発行者に支払するために、ホスティングサービス制御システム 401 により収集され、そして R A I D アレイ 1511-1512 に記憶される。統計値のあるものは、サービス制御システム 401 によりアクションによって記録され、そしてあるものは、個々の app/game サーバー 1521-1525 によりサービス制御システム 401 へ報告される。例えば、この「ゲームファインダー」アプリケーションを実行する app/game サーバー 1521-1525 は、ゲームを見て

10

20

30

40

50

いるとき（及びゲームを見終えたとき）ホスティングサービス制御システム401へメッセージを送信し、どれほど多くのゲームが見られているかの統計値を更新できるようにする。統計値のあるものは、この「ゲームファインダー」アプリケーションのようなユーザインターフェイスアプリケーションに利用される。

【0205】

ユーザは、入力装置のアクチベーションボタンをクリックした場合に、フルスクリーンサイズへの生のビデオのプレイを続けながら黄色のボックス内のサムネイルビデオをズームアップして見ることができる。この効果が図17のプロセスに示されている。ビデオウィンドウ1700は、サイズが増大していることに注意されたい。この効果を具現化するために、app/gameサーバー1521-1525は、選択されたゲームを実行しているapp/gameサーバー1521-1525から、そこにルーティングされたゲームのフルスクリーンサイズのビデオストリームのコピーを取ることを要求する（ユーザのディスプレイ装置422の解像度で）。ゲームを実行しているapp/gameサーバー1521-1525は、共有ハードウェアコンプレッサ1530に、ゲームのサムネイルサイズのコピーがもはや必要ない（別のapp/gameサーバー1521-1525がそのようなサムネイルを要求しない限り）ことを通知し、次いで、ビデオのフルスクリーンサイズコピーを、ビデオをズームしているapp/gameサーバー1521-1525へ送信することを指令する。ゲームをプレイしているユーザは、ゲームをズームアップしているユーザと同じ解像度のディスプレイ装置422を有していても、いなくてもよい。更に、ゲームを見る他の者も、ゲームをズームアップしているユーザと同じ解像度のディスプレイ装置422を有していても、いなくてもよい（且つ異なるオーディオ再生手段、例えば、ステレオ又はサラウンドサウンドを有してもよい）。従って、共有ハードウェアコンプレッサ1530は、ビデオ/オーディオストリームを要求しているユーザの要件を満足する適当に圧縮されたビデオ/オーディオストリームが既に発生されたかどうか決定し、もしそれが存在する場合には、ビデオをズームしているapp/gameサーバー1521-1525へストリームのコピーをルーティングするようにアウトバウンドルーティング1540に通知し、又、もしそうでなければ、そのユーザに適したビデオの別のコピーを圧縮して、ビデオをズームしているapp/gameサーバー1521-1525及びインバウンドルーティング1502へストリームを返送するようにアウトバウンドルーティングに命令する。選択されたビデオのフルスクリーンバージョンを現在受信しているこのサーバーは、それを解凍し、フルサイズまで徐々にスケールアップする。

【0206】

図18は、ゲームがフルスクリーンへ完全にズームアップされた後にスクリーンがどのように見えるか示しており、ゲームは、矢印1800で指された映像により示すように、ユーザのディスプレイ装置422の全解像度で示されている。ゲームファインダーアプリケーションを実行しているapp/gameサーバー1521-1525は、もはや必要でないサムネイルを発生した他のapp/gameサーバー1521-1525へメッセージを送信すると共に、他のゲームがもはや見られていないホスティングサービス制御サーバー401へメッセージを送信する。この点において、発生される表示は、スクリーンの頂部にあって、情報及びメニュー制御をユーザに与えるオーバーレイ1801である。このゲームが進行するにつれて、観衆は、2503人の視聴者まで増加したことに注意されたい。このように大勢の視聴者では、同じ又はほぼ同じ解像度をもつディスプレイ装置422に大勢の視聴者が結び付けられる（各app/gameサーバー1521-1525は、適合性を調整するためにビデオをスケールアップする能力を有する）。

【0207】

図示されたゲームは、マルチプレーヤゲームであるから、ユーザは、ある点でゲームに加わることを決定できる。ホスティングサービス210は、種々の理由で、ユーザがゲームに加わるのを許しても許さなくてもよい。例えば、ユーザは、ゲームをプレイするために支払しなければならず、又、プレイしないことを選んでもよく、ユーザは、その特定のゲームに加わるに十分なランキングを有していないことがあり（例えば、他のプレーヤと

の競争に耐えない)、或いはユーザのインターネット接続は、ユーザがプレイするのに十分な短い待ち時間を有していないことがある(例えば、ゲームを見るための待ち時間制約がなく、従って、遠くで(實際上、別の大陸で)プレイされるゲームを、待ち時間の問題なしに見ることはできるが、プレイするゲームについては、待ち時間は、(a)ゲームを楽しむために、及び(b)短い待ち時間接続を有する他のプレーヤと平等の立場であるために、ユーザにとって充分短いものでなければならない)。ユーザがプレイを許された場合には、ユーザに対して「ゲームファインダー」ユーザインターフェイスを与えたapp/gameサーバー1521-1525は、ホスティングサービス制御サーバー401が、特定のゲームをプレイするために適当に構成されたapp/gameサーバー1521-1525を始動して(即ち、探索し始動して)、RAIDアレイ1511-1512からゲームをロードし、次いで、ホスティングサービス制御サーバー401が、ゲームを現在ホストしているapp/gameゲームサーバーへユーザから制御信号を転送するようにインバウンドルーティング1502に命令すると共に、「ゲームファインダー」アプリケーションをホストしたapp/gameサーバーからのビデオ/オーディオを圧縮することから、ゲームを現在ホストしているapp/gameサーバーからのビデオ/オーディオを圧縮することへスイッチするように共有ハードウェアコンプレッサ1530に命令することを要求する。「ゲームファインダー」app/gameサービス、及びゲームをホストしている新たなapp/gameサーバーの垂直同期は、同期されず、その結果、2つの同期間におそらく時間差が生じる。共有ビデオ圧縮ハードウェア1530は、app/gameサーバー1521-1525がビデオフレームを完成したときにビデオの圧縮を開始するので、新たなサーバーからの第1のフレームは、古いサーバーのフルフレーム時間より早期に完成され、これは、その手前の圧縮されたフレームがその送信を完了する前である(例えば、図9bの送信時間992を考えると、非圧縮フレーム3963がフレーム時間の半分早期に完成された場合に、送信時間992に影響を及ぼす)。このような状況では、共有ビデオ圧縮ハードウェア1530は、新たなサーバーからの第1のフレームを無視し(例えば、フレーム4964が無視される(974)のと同様に)、クライアント415は、古いサーバーからの最後のフレームを特別のフレーム時間中保持し、そして共有ビデオ圧縮ハードウェア1530は、ゲームをホストする新たなapp/gameサーバーからの次のフレーム時間ビデオの圧縮を開始する。視覚上、あるapp/gameサーバーから他のapp/gameサーバーへの移行は、シームレスである。ホスティングサービス制御サーバー401は、次いで、「ゲームファインダー」をホストしていたapp/gameゲームサーバー1521-1525に、それが再び必要になるまでアイドル状態にスイッチするように通知する。

【0208】

次いで、ユーザは、ゲームをプレイすることができる。そして、何が優れているかと言えば、ゲームは、知覚上即座にプレイされ(ギガビット/秒の速度でRAIDアレイ1511-1512からapp/gameゲームサーバー1521-1525にロードされるので)、そしてゲームは、理想的なドライバ、レジストリ構成(Windowsの場合)をもち、且つゲームのオペレーションと競合し得るサーバーで実行される他のアプリケーションをもたずに、ゲームに対して厳密に構成されたオペレーティングシステムと一緒にゲームに厳密に適したサーバーにロードされる。

【0209】

又、ユーザがゲームを通して進行するにつれて、ゲームの各セグメントが、RAIDアレイ1511-1512からギガビット/秒の速度でサーバーへロードされ(即ち、1ギガバイトの負荷が8秒で)、そしてRAIDアレイ1511-1512の膨大な記憶容量のために(これは多数のユーザ間の共有リソースであるから、非常に大きく、又、コスト効率がよい)、幾何学的な設定又は他のゲームセグメント設定を予め計算して、RAIDアレイ1511-1512に記憶し、そして非常に迅速にロードすることができる。更に、各app/gameサーバー1521-1525のハードウェア構成及び計算能力が既知であるから、ピクセル及び頂点シェーダーを予め計算することができる。

【0210】

従って、ゲームは、ほぼ瞬間的に始動し、理想的な環境で実行され、そしてその後のセグメントがほぼ瞬間的にロードされる。

【0211】

しかし、これらの効果を越えて、ユーザは、ゲームをプレイする他人を見ることができ（上述した「ゲームファインダー」及び他の手段を経て）、そしてゲームに関心があるかどうか判断し、もしそうであれば、他人を見ることから秘訣を学習することができる。そして、ユーザは、大きなダウンロード及び／又はインストールを待機する必要なく、ゲームを瞬時にデモすることができ、又、ユーザは、おそらく、僅かな費用の試行ベースで、又は長期間ベースで、ゲームを瞬時にプレイすることができる。そして、ユーザは、ゲームを、ウィンドウズPC、マッキントッシュ、家のTV受像機で、又、旅行中も、移動電話で、充分短い待ち時間のワイヤレス接続によりプレイすることができる（見物するだけでは、待ち時間は問題にならないが）。又、これは、全て、ゲームのコピーを物理的に所持せずに行うことができる。

【0212】

上述したように、ユーザは、自分のゲームプレイを他人に見られることを許さないか、自分のゲームを遅延の後に見られるのを許すか、自分のゲームを選択されたユーザにより見られるのを許すか、又は自分のゲームを全てのユーザにより見られるのを許すかを判断することができる。それにも関わらず、ビデオ／オーディオは、一実施形態では、15分間、遅延バッファ1515に記憶され、そしてユーザは、デジタルビデオレコーダ（DVR）でTVを見ていたときに行うことができたのと同様に、自分の以前のゲームプレイを「巻き戻して」見ることができ、又、休止し、スローで再生し、早送りし、等々を行うことができる。この例では、ユーザは、ゲームをプレイするが、ユーザがアプリケーションを使用する場合には、同じ“DVR”能力を利用することができる。これは、以前の行動を再検討すること及び以下に述べる他のアプリケーションでも有用である。更に、カメラの視野を変更できる、等のように、ゲーム状態情報の利用に基づいて巻き戻すことができるようにゲームが設計されている場合は、この“3D DVR”能力もサポートされるが、それをサポートするようにゲームを設計することが要求される。遅延バッファ1515を使用する“DVR”能力は、もちろん、ゲーム又はアプリケーションが使用されたときに発生するビデオに限定して、ゲーム又はアプリケーションと共に機能するが、3D DVR能力をもつゲームの場合には、ユーザは、以前にプレイしたセグメントの3Dにおける「フライスルー(fly through)」を制御することができ、そしてそれにより得られるビデオを遅延バッファ1515で記録させ、ゲームセグメントのゲーム状態を記録させることができる。従って、特定の「フライスルー」が、圧縮されたビデオとして記録されるが、ゲーム状態も記録されるので、後日、ゲームの同じセグメントについて異なるフライスルーが考えられることになる。

【0213】

以下に述べるように、ホスティングサービス210におけるユーザは、各々、彼等に関する情報及び他のデータを掲示することのできる「ユーザページ」を有する。ユーザが掲示することのできるものの中には、彼等がセーブしたゲームプレイからのビデオセグメントがある。例えば、ユーザがゲームにおいて特に困難な挑戦を克服した場合には、ユーザは、ゲームにおいて偉大な成果をなし得た点の直前まで「巻き戻し」、次いで、他のユーザが見るために、ある時間巾（例えば、30秒）のビデオセグメントをユーザの「ユーザページ」にセーブするようにホスティングサービス210に命令することができる。これを達成するには、ユーザが、遅延バッファ1515に記憶されたビデオをRAIDアレイ1511-1512へ再生してユーザの「ユーザページ」においてそのビデオセグメントをインデックスするように使用するという、単なるapp/gameサーバー1521-1525の問題である。

【0214】

ゲームが、上述した3D DVRの能力を有する場合には、3D DVRに要求される

ゲーム状態情報もユーザによって記録して、ユーザの「ユーザページ」に利用することができる。

【0215】

能動的なプレーヤに加えて「観客」（即ち3Dの世界を進んで、アクションに関与せずにそれを観察できるユーザ）をもつようにゲームが設計されている場合には、「ゲームファインダー」アプリケーションは、ユーザが観客及びプレーヤとしてゲームに加わることができるようにする。具現化の視点から、ユーザが能動的なプレーヤではなく観客である場合も、ホスティングシステム210に対して差はない。ゲームは、app/gameサーバー1521-1525にロードされ、そしてユーザは、ゲームを制御する（例えば、世界を見るバーチャルカメラを制御する）。唯一の相違は、ユーザのゲーム経験である。

10

【0216】

複数ユーザの協力

ホスティングサービス210の別の特徴は、複数のユーザが、大きく異なる視聴装置を使用している、生のビデオを見ながら協力する能力である。これは、ゲームをプレイするとき及びアプリケーションを使用するときの両方に有用である。

【0217】

多くのPC及び移動電話は、ビデオカメラを装備しており、且つ特に、映像が小さいときには、リアルタイム映像圧縮を行う能力を有している。又、テレビに取り付けることのできる小型カメラも入手でき、リアルタイム圧縮を、ソフトウェアで具現化するか、又はビデオを圧縮するための多数のハードウェア圧縮装置の1つを使用して具現化するのは、困難なことではない。又、多数のPC及び全ての移動電話は、マイクロホン有し、又、マイクロホンと共にヘッドセットが利用できる。

20

【0218】

このようなカメラ及び／又はマイクロホンは、ローカルのビデオ／オーディオ圧縮能力（特に、ここに述べる短待ち時間のビデオ圧縮技術を使用する）と合成されると、ユーザが、ユーザの家屋211からのビデオ及び／又はオーディオを入力装置の制御データと共にホスティングサービス210へ送信できるようにする。このような技術が使用されるときには、図19に示す能力を達成でき、即ちユーザは、自分のビデオ及びオーディオ1900を別のユーザのゲーム又はアプリケーション内のスクリーンに出現させることができる。この例は、カーレースにおいてチームメイトが協力するマルチプレーヤゲームである。ユーザのビデオ／オーディオは、チームメイトだけにより選択的に見る／聞くことができる。そして、実質上、待ち時間がないので、上述した技術を使用すると、プレーヤは、気付くほどの遅延を伴うことなく、互いにリアルタイムで話したり動いたりすることができる。

30

【0219】

このビデオ／オーディオの一体化は、ユーザのカメラ／マイクロホンからの圧縮されたビデオ及び／又はオーディオをインバウンドインターネットトラフィック1501として到着させることにより達成される。次いで、インバウンドルーティング1502が、ビデオ及び／又はオーディオを、それを見る／聞くことが許されたapp/gameゲームサーバー1521-1525へルーティングする。次いで、ビデオ及び／又はオーディオを使用することを選択する各app/gameゲームサーバー1521-1525のユーザは、それを解凍し、そして1900で示されたように、必要に応じて、ゲーム又はアプリケーション内に現れるように一体化する。

40

【0220】

図19の例は、このような協力がゲームにおいてどのように利用されるか示しているが、このような協力は、アプリケーションにとって非常にパワフルなツールである。ニューヨーク市ではニューヨークに本拠地を置く不動産開発業者のためにシカゴの建築家により大きなビルが設計されているが、その判断には、旅行中でマイアミの空港にたまたまいる金融投資家が含まれ、そして投資家と不動産開発業者の両方を満足させるために、周囲のビルにどのように調和させるかに関してそのビルの幾つかの設計要素について判断をする

50

必要がある状況を考える。建設会社は、シカゴにおいてカメラがPCに取り付けられた高解像度モニタを有し、不動産開発業者は、ニューヨークにおいてカメラ付きのラップトップを有し、そして投資家は、マイアミにおいてカメラ付きの携帯電話を有している。建設会社は、ホスティングサービス210を使用して、非常に現実的な3Dレンダリングを行うことのできるパワフルな建設設計アプリケーションをホストすることができ、ニューヨーク市内のビルの大きなデータベース及び設計中のビルのデータベースを使用することができる。建設設計アプリケーションは、app/gameサーバー1521-1525の1つで実行され、又は多大な計算パワーを要求する場合には、その幾つかにおいて実行される。異なる位置にいる3人のユーザの各々は、ホスティングサービス210に接続され、そしてその各々は、建設設計アプリケーションのビデオ出力の同時ビューを有するが、これは、各ユーザが有する所与の装置及びネットワーク接続特性に対して共有ハードウェア圧縮1530により適当なサイズにされる（例えば、建設会社は、20Mbpsの商業的インターネット接続を通して2560x1440の60fpsディスプレイを見ることができ、ニューヨークの不動産開発業者は、自分のラップトップの6MbpsのDSL接続を経て1280x720の60fps映像を見ることができ、そして投資家は、自分の携帯電話の250Kbpsのセルラーデータ接続を経て320x180の60fps映像を見ることができる。各当事者は、他の当事者の音声を聞き（会議コールは、app/gameサーバー1521-1525において広く利用できる多数の会議コールソフトウェアパッケージのいずれかにより取り扱われ）、そしてユーザ入力装置上のボタンの操作を通して、ユーザは、ローカルカメラを使用してそれら自身のビデオ出演をすることができる。ミーティングが進むにつれて、建築家は、そのビルを回転し且つそのエリアの他のビルの近くを飛行したときにそのビルがどのように見えるか、非常にホトリアルな3Dレンダリングで示すことができ、そして同じビデオを、各当事者のディスプレイ装置の解像度で全ての当事者が見ることができる。当事者により使用されるローカル装置が、どれも、このようなリアルリズムで3Dアニメーションを取り扱いきれないことは問題ではないが、ニューヨーク市の周囲のビルをレンダリングするために要求される膨大なデータベースをダウンロードするか又は記憶することは言うまでもない。各ユーザの視点から、距離が離れているにも関わらず、又、異なるローカル装置であるにも関わらず、ユーザは、単純に、信じられないほどのリアルリズムで繋ぎ目のない経験をする。そして、1人の当事者が顔が見えるようにして感情を良く伝えたいときには、そのようにすることができる。更に、不動産開発業者又は投資家のいずれかが建設プログラムの制御を手に入れてそれら自身の入力装置（キーボード、マウス、キーパッド又はタッチスクリーン）を使用したい場合には、知覚上待ち時間なしに応答でき、又、応答することになる（それらのネットワーク接続が不合理な待ち時間をもたないと仮定して）。例えば、携帯電話の場合には、その携帯電話が空港のWiFiネットワークに接続された場合には、待ち時間が非常に短くなる。しかし、米国で今日利用できるセルラーデータネットワークを使用する場合には、おそらく、顕著な遅れで悩まされることになる。更に、投資家が見ているミーティングのほとんどの目的では、建築家がビルの接近飛行を制御し、又はビデオ遠隔会議で話をするためには、セルラー待ち時間でも許容できねばならない。

【0221】

最終的に、協力会議コールの終わりに、不動産開発業者及び投資家は、ホスティングサービスから彼等のコメントを発し且つ署名し、建設会社は、遅延バッファ1515に記録された会議のビデオを「巻き戻し」、そしてコメント、顔の表情、及び／又はミーティング中に作られたビルの3Dモデルに加えられたアクションを再検討することができる。セーブしたい特定のセグメントがある場合は、ビデオ／オーディオのそれらセグメントを、記録保管及び後で再生するために遅延バッファ1515からRAIDアレイ1511-1512へ移動することができる。

【0222】

又、コストの観点から、建築家が計算パワー及びニューヨーク市の大きなデータベースを15分の会議コールにしか使用する必要がない場合には、ハイパワーのワークステーシ

ョンを所有し且つ大きなデータベースの高価なコピーを購入するのではなく、リソースを使用した時間について支払するだけでよい。

【0223】

ビデオリッチなコミュニティサービス

ホスティングサービス210は、インターネットにビデオリッチなコミュニティサービスを確立するための空前の機会を可能にする。図20は、ホスティングサービス210におけるゲームプレーヤーのための規範的「ユーザページ」を示す。「ゲームファインダー」アプリケーションと同様に、「ユーザページ」は、app/gameサーバー1521-1525の1つにおいて実行されるアプリケーションである。このページにおけるサムネール及びビデオウィンドウは、全て、絶えず動くビデオ（セグメントが短い場合は、ループになる）を示す。

10

【0224】

ビデオカメラを使用するか、又はビデオをアップロードすることにより、ユーザ（ユーザ名は“KILLHAZARD”である）は、自分自身のビデオ2000を掲示することができ、他のユーザがこれを見ることができる。このビデオは、RAIDアレイ1511-1512に記憶される。又、他のユーザがKILLHAZARDの「ユーザページ」に到来するときに、KILLHAZARDがそのときホスティングサービス210を使用していれば、（ユーザが彼を見るために彼の「ユーザページ」を見ることを許すと仮定すれば）彼が何を行おうとその生のビデオ2001が示される。これは、サーバー制御システム401から要求される「ユーザページ」アプリケーションをホストするapp/gameサーバー1521-1525により達成され、KILLHAZARDがアクティブであるかどうか、そしてもしそうであれば、彼が使用するapp/gameサーバー1521-1525により達成される。次いで、「ゲームファインダー」アプリケーションにより使用される同じ方法を使用して、適当な解像度及びフォーマットの圧縮されたビデオストリームが、「ユーザページ」アプリケーションを実行するapp/gameサーバー1521-1525へ送信され、そして表示される。ユーザがKILLHAZARDの生のゲームプレイでウィンドウを選択し、次いで、その入力装置を適当にクリックする場合は、ウィンドウがズームアップされ（この場合も「ゲームファインダー」アプリケーションと同じ方法を使用して）、そして生のビデオが、見ているユーザのインターネット接続の特性に適した、見ているユーザのディスプレイ装置422の解像度でスクリーンを埋める。

20

30

【0225】

従来に勝るこの解決策の重要な効果は、「ユーザページ」を見ているユーザが、自分が所有していないゲームの生のプレイを見ることができ、且つ非常に良いことに、ゲームをプレイできるローカルコンピュータ又はゲームコンソールをもたなくてよいことである。これは、ゲームをプレイする「アクション中」である「ユーザページ」に示されたユーザをユーザが見る良い機会を与えると共に、見ているユーザが試してみたい又は上手になりたいゲームに関して学習する機会でもある。

【0226】

KILLHAZARDの相棒2002からのカメラ記録又はアップロードされたビデオクリップも「ユーザページ」に示され、そして各ビデオクリップの下には、相棒がオンラインでゲームをプレイしているかどうか示すテキストがある（例えば、six_shotは、ゲーム“Eragon”をプレイしており（ここでは、Game4として示す）、MrSnuggles99は、オフラインであり、等）。メニューアイテム（図示せず）をクリップすることにより、相棒のビデオクリップは、記録又はアップロードされたビデオを示すものから、ホスティングサービス210上で現在ゲームをプレイしている相棒がそのときゲームにおいて何をしているかの生のビデオへとスイッチする。従って、これは、相棒のためのゲームファインダーグループとなる。相棒のゲームが選択されそしてユーザがそれをクリックする場合には、全スクリーンをズームアップし、そしてユーザは、全スクリーンでプレイされるゲームを生で見ることができる。

40

【0227】

50

この場合も、相棒のゲームを見ているユーザは、ゲームのコピーも、ゲームをプレイするためのローカルコンピューティング／ゲームコンソールリソースも、所有していない。ゲームを見るのは、實際上、瞬間的である。

【0228】

上述したように、ユーザがホスティングサービス210においてゲームをプレイするときには、ユーザは、ゲームを「巻き戻し」、セーブしたいビデオセグメントを見出し、次いで、ビデオセグメントを自分の「ユーザページ」へセーブする。これらは、「自慢のクリップ(Brag Clips)(TM)」と称される。ビデオセグメント2003は、全て、プレイした以前のゲームからKILLHARZARDによりセーブされた「自慢のクリップ」2003である。番号2004は、「自慢のクリップ」が何回見られたかを示し、そして「自慢のクリップ」が見られるときに、ユーザは、それを格付けする機会を得、オレンジ（ここでは黒の輪郭として示す）のキー穴形状のアイコン2005の数は、どれほど高い格付けであるかを示す。「自慢のクリップ」2003は、ユーザが「ユーザページ」を見るとき、そのページにおけるビデオの残りと共に絶えずループする。ユーザが「自慢のクリップ」2003の1つを選択してクリックする場合には、それがズームアップし、「自慢のクリップ」2003を、クリップの再生、休止、巻き戻し、早送り、ステップスルー、等を行えるようにするDVRコントロールと共に提示する。

【0229】

「自慢のクリップ」2003の再生は、ユーザが「自慢のクリップ」を記録し、それを解凍して再生するときに、RAIDアレイ1511-1512に記憶された圧縮ビデオセグメントをapp/gameサーバー1521-1525がロードすることにより具現化される。

【0230】

又、「自慢のクリップ」2003は、“3D DVR”ビデオセグメント（即ち、再生でき且つユーザがカメラの視点を変更するのを許すゲームからのゲーム状態シーケンス）でもあり、これは、そのような能力をサポートするゲームからのものである。この場合、ゲームセグメントが記録されたときにユーザが作った特定の「フリスルー」の圧縮ビデオ記録に加えて、ゲーム状態情報が記憶される。「ユーザページ」が見られ、そして全てのサムネール及びビデオウィンドウが絶えずループ化するときには、3D DVR「自慢のクリップ」2003は、ユーザがゲームセグメントの「フリスルー」を記録したときに圧縮ビデオとして記録された「自慢のクリップ」2003を絶えずループ化する。しかし、ユーザが3D DVR「自慢のクリップ」2003を選択してそれをクリップするときには、圧縮ビデオの「自慢のクリップ」を再生できるようにするDVRコントロールに加えて、ユーザは、ゲームセグメントに対する3D DVR能力を与えるボタンをクリックすることができる。それらは、彼等自身でゲームセグメント中にカメラの「フリスルー」を制御することができ、そしてそれらが望む（及びユーザページを所有するユーザがそれを許す）場合に、別の「自慢のクリップ」の「フリスルー」を圧縮ビデオ形態で記録することができ、これは、次いで、ユーザページの他の視聴者に利用可能となる（直ちに、又はユーザページの所有者が「自慢のクリップ」を再検討する機会をもった後に）。

【0231】

この3D DVR「自慢のクリップ」2003の能力は、記録されたゲーム状態情報を別のapp/gameサーバー1521-1525において再生しようとしているゲームを作動することによりイネーブルされる。ゲームは、ほぼ瞬間的に作動できるので（上述したように）、「自慢のクリップ」により記録されたゲーム状態にプレイを限定してゲームを作動し、次いで、ユーザが、圧縮ビデオを遅延バッファ1515に記録しながらカメラで「フリスルー」を実行できるようにするのは、困難ではない。ユーザが「フリスルー」の実行を完了すると、ゲームが不作動にされる。

【0232】

ユーザの視点から、3D DVRの「自慢のクリップ」2003で「フリスルー」を作動することは、リニアな「自慢のクリップ」2003のDVRコントロールを制御する

ことより努力が要らない。それらは、ゲームについて又はゲームをどのようにプレイするかについて何も知らなくてよい。それらは、別の者により記録されるゲームセグメント中に3D世界を凝視するバーチャルカメラオペレータに過ぎない。

【0233】

又、ユーザは、それら自身のオーディオを、マイクロホンから記録されるか又はアップロードされる「自慢のクリップ」にオーバーダビングすることもできる。このようにして、「自慢のクリップ」は、ゲームからのキャラクタ及びアクションを使用して、カスタムアニメーションを生成するのに使用できる。このアニメーション技術は、「マシニマ(machinima)」として一般に知られている。

【0234】

ユーザがゲームを進めるにつれて、異なるスキルレベルを達成する。プレイしたゲームは、達成度をサービスコントロールシステム401へ報告し、それらのスキルレベルが「ユーザページ」に示される。

【0235】

双方向アニメ化広告

オンライン広告は、テキストから、静止映像へ、ビデオへ、そして今や、Adobe Flashのようなアニメーションシンクライアントを使用して典型的に具現化される双方向セグメントへと推移してきた。アニメーションシンクライアントを使用する理由は、ユーザが、典型的に、製品やサービスが彼等に推薦される特典に対する遅れが耐え難いからである。又、シンクライアントは、非常に低性能のPCで実行され、従って、広告主は、双方向アドが適切に機能するという高い信頼度をもつことができる。不都合なことに、Adobe Flashのようなアニメーションシンクライアントは、双方向性の度合い及び経験の中が制限される(ダウンロード時間を軽減し、そしてGPUをもたない低性能PC及びMac、又は高性能CPUを含むほぼ全てのユーザ装置で動作するために)。

【0236】

図21は、ショールーム内で車を回転させ、車がどのように見えるかリアルタイムレイトレーシングで示しながら、ユーザが車の外部及び内部の色を選択する双方向広告を示している。ユーザは、車を運転するアバターを選択し、次いで、ユーザは、レーストラック上、又はモナコのようなエキゾチックな場所でドライブするように車を取り出すことができる。ユーザは、より大きなエンジン又はより優れたタイヤを選択することができ、次いで、その変更した構成が車の加速性能又は路面の食い付きにどのように影響するか見ることができる。

【0237】

もちろん、広告は、實際上、精巧な3Dビデオゲームである。しかし、PC又はビデオゲームコンソール上に再生できる広告については、おそらく、100MBダウンロードを要求し、そしてPCの場合には、特殊なドライバのインストールを要求すると共に、PCが十分なCPU又はGPU計算能力に欠ける場合には全く実行されないことがある。従って、このような広告は、従来の構成では不可能である。

【0238】

ホスティングサービス210では、このような広告は、ほぼ瞬時に起動し、そしてユーザのクライアント415の能力がどんなものであっても、完全に実行される。従って、それらは、シンクライアント双方向アドより素早く起動し、著しく経験に富み、且つ非常に信頼性が高い。

【0239】

リアルタイムアニメーション中のストリーミング幾何学形状

RAIDアレイ1511-1512及びインバウンドルーティング1502は、リアルタイムアニメーション(例えば、複雑なデータベースを伴うフライスルー)中にゲームプレイ又はアプリケーションの中間において幾何学形状をオンザフライで確実に配信するためにRAIDアレイ1511-1512及びインバウンドルーティング1502に依存するビデオゲーム及びアプリケーションを設計できるように高速で且つ短待ち時間のデータ

レートを与えることができる。

【0240】

図1に示すビデオゲームシステムのような従来のシステムでは、特に実際の家庭用装置に使用できる大量記憶装置は、要求される幾何学形状をある程度予想できる状況を除き、ゲームプレイ中に幾何学形状をストリームするには遥かに低速である。例えば、特定の道路が存在するドライブゲームにおいて、視野に入ってくるビルなどの幾何学形状は、合理的に充分なほど予想でき、そして大量記憶装置は、近づきつつある幾何学形状が位置している場所を前もって探すことができる。

【0241】

しかし、予想できない変化を伴う複雑なシーンにおいて（例えば、あたり一面に複雑なキャラクタが存在する戦闘のシーンにおいて）、PC又はビデオゲームシステムのRAMが、現在視野にある物体に対する幾何学形状で完全に埋められており、次いで、ユーザが突然それらのキャラクタを、それらキャラクタの背後の視野へ方向転換する場合に、幾何学形状がRAMに予めロードされていなければ、それを表示できるまでに遅延が生じることになる。

【0242】

ホスティングサービス210において、RAIDアレイ1511-1512は、ギガビットイーサネット速度以上でデータをストリームすることができ、そしてSANネットワークでは、10ギガビットのイーサネット又は他のネットワークに勝る10ギガビット/秒の速度を達成することができる。10ギガビット/秒は、ギガバイトのデータを1秒未満でロードする。60fpsのフレーム時間（16.67ms）では、ほぼ170メガビット（21MB）のデータをロードすることができる。当然、RAID構成にある回転するメディアは、1フレーム時間より大きな待ち時間を依然被るが、フラッシュベースのRAID記憶装置は、結局、回転するメディアRAIDアレイと同程度の大きさであり、そのような長い待ち時間は被らない。一実施形態では、大量RAMライトスルーキャッシングを使用して、非常に短い待ち時間のアクセスを与える。

【0243】

従って、十分に高いネットワーク速度及び十分に短い待ち時間の大量記憶では、CPU及び/又はGPUが3Dデータを処理できるのと同程度の速さでapp/gameゲームサーバー1521-1525へ幾何学形状をストリームすることができる。従って、ユーザがそのキャラクタを突然方向転換しそして後を見る上述した例では、後方の全てのキャラクタに対する幾何学形状を、キャラクタが回転を完了する前にロードすることができ、従って、ユーザにとって、生のアクションと同程度にリアルなホトリアルな世界にいるかのように思える。

【0244】

上述したように、ホトリアルなコンピュータアニメーションにおける最後の未開拓の1つは、人間の顔であり、不完全さに対して人間の目が敏感であるために、ホトリアルな顔からの若干のエラーが視聴者から否定的なリアクションを招くことがある。図22は、ContourTM「リアリティキャプチャーテクノロジー」（本CIP出願の譲受人に各々譲渡された同時係争中の特許出願、即ち2004年9月15日に出願された第10/942,609号“Apparatus and method for capturing the motion of a performer”、2004年9月15日に出願された第10/942,413号“Apparatus and method for capturing the expression of a performer”、2005年2月25日に出願された第11/066,954号“Apparatus and method for improving marker identification within a motion capture system”、2005年3月10日に出願された第11/077,628号“Apparatus and method for performing motion capture using shutter synchronization”、2005年10月20日に出願された第11/255,854号“Apparatus and method for performing motion capture using a random pattern on capture surfaces”、2006年6月7日に出願された第11/449,131号“System and method for performing motion capture using phosphor application techniques”、2

10

20

30

40

50

006年6月7日に出願された第11/449,043号“System and method for performing motion capture by strobing a fluorescent lamp”、2006年6月7日に出願された第11/449,127号“System and method for three dimensional capture of stop-motion animated characters”の要旨)を使用して捕獲される生の演技が、非常に滑らかな捕獲表面を、次いで、高多角形数追跡表面(即ち、多角形の動きが顔の動きに正確に追従する)をどのように生じるか示している。最終的に、生の演技のビデオが追跡表面にマップされて、テクスチャ処理表面を形成するときに、ホトリアルな結果が得られる。

【0245】

現在のGPU技術は、追跡表面における多角形の数及びテクスチャをレンダリングしそして表面をリアルタイムで照明することができるが、多角形及びテクスチャがフレーム時間ごとに変化する(これは、最もホトリアルな結果を生じる)場合には、近代的PC又はビデオゲームコンソールの利用可能な全てのRAMを急速に消費してしまう。

【0246】

上述したストリーミング幾何学形状技術を使用すると、app/gameゲームサーバー1521-1525へ幾何学形状を連続的に供給して、ホトリアルな顔を連続的にアニメ化し、生の動きの顔とほとんど見分けがつかない顔をもつビデオゲームを作成できるようにするが実際的である。

【0247】

リニアなコンテンツと双方向特徴との一体化

動画、テレビ番組及びオーディオ資料(集合的に「リニアコンテンツ」)は、家庭及びオフィスのユーザに多数の形態で広く利用することができる。リニアコンテンツは、CD、DVD及びBlu-rayメディアのような物理的なメディアにおいて取得することができる。又、これは、衛星及びケーブルTV放送からDVRにより記録することもできる。そして、衛星及びケーブルTVを通してペイパービュー(PPV)コンテンツとして、及びケーブルTVのビデオオンデマンド(VOD)として利用することもできる。

【0248】

益々多くのリニアコンテンツが、インターネットを通して、ダウンロードされたコンテンツとして及びストリーミングコンテンツとして利用できるようになった。今日、リニアメディアに関連した全ての特徴を経験するための場所は、実際に1つではない。例えば、DVD及び他のビデオ光学メディアは、典型的に、演出家の解説、「メイキングオブ(making of)」特徴、等のように、他の場所では利用できない双方向特徴を有する。オンライン音楽サイトは、一般的にCDでは入手できないカバー技術及び歌情報を有するが、全てのCDをオンラインで入手できるのではない。又、テレビ番組に関連したウェブサイトは、しばしば、特別な特徴、ブログ、及びときには、俳優又は創作スタッフからのコメントを有する。

【0249】

更に、多くの動画又はスポーツの出来事では、(動画の場合)しばしばリニアメディアと一緒に発売されるか、或いは(スポーツの場合)実世界の出来事(例えば、プレイヤーのトレード)に密接に結びついたビデオゲームがしばしばある。

【0250】

ホスティングサービス210は、異なる形態の関連コンテンツと一緒にリンクしてリニアコンテンツを配布するのに良く適している。確かに、動画の配信は、もはや、高度な双方向性のビデオゲームを配信する挑戦ではなく、ホスティングサービス210は、リニアコンテンツを家庭又はオフィスの広範な装置或いは移動装置へ配信することができる。図23は、リニアコンテンツの選択を示すホスティングサービス210の規範的なユーザーインターフェイスページを示している。

【0251】

しかし、ほとんどのリニアコンテンツ配信システムとは異なり、ホスティングサービス210は、関連する双方向コンポーネント(例えば、DVDにおけるメニュー及び特徴、

10

20

30

40

50

H D - D V D における双方向オーバーレイ、及びウェブサイトにおける A d o b e F l a s h アニメーション（以下に述べる）を配信することもできる。従って、クライアント装置 4 1 5 の制限は、どの特徴が利用できるかについて、もはや制限を導入しない。

【0252】

更に、ホスティングサービス 2 1 0 は、リニアコンテンツをビデオゲームコンテンツと動的に且つリアルタイムでリンクすることができる。例えば、ユーザがハリーポッターの映画においてクィディッチの試合を見て、クィディッチのプレイにトライしたい場合に、ボタンをクリックするだけで、映画を休止し、直ちに、ハリーポッタービデオゲームのクィディッチセグメントへ移ることができる。クィディッチの試合をプレイした後に、ボタンの別のクリックで、即座に映画が再開する。

10

【0253】

写真的に捕獲されたビデオが生の動きのキャラクタと見分けがつかないようなホトリアルなグラフィック及び制作技術では、ユーザが、生の動きの映画におけるクィディッチゲームから、ここに述べるホスティングサービスにおけるビデオゲームのクィディッチゲームへ移行するときに、2つのシーンは、実質上、見分けがつかない。これは、2つの世界の間の線が見分けられないのでリニアコンテンツ及び双方向（例えば、ビデオゲーム）コンテンツの両方の演出家にとって全く新しい創作オプションを与える。

【0254】

図 1 4 に示すホスティングサービスアーキテクチャーを使用して、3 D 映画のバーチャルカメラの制御権を視聴者に与えることができる。例えば、列車の車両内で生じるシーンでは、視聴者がバーチャルカメラを制御して、ストーリーが進行する間に車両の周囲を見ることができる。これは、車両内の全ての 3 D オブジェクト（アセット）が利用できると共に、シーンをリアルタイムでレンダリングできる充分なレベルの計算パワー、及びオリジナルの映画を仮定している。

20

【0255】

そして、非コンピュータ発生娯楽であっても、非常にエキサイティングな双方向特徴を提供することができる。例えば、2 0 0 5 動画“高慢と偏見(Pride and Prejudice)”は、華麗な古い英国風大邸宅における多くのシーンを有している。大邸宅の幾つかのシーンに対し、ユーザは、ビデオを休止し、次いで、カメラを制御して、大邸宅の塔、又はおそらく、周囲のエリアを撮影することができる。これを具現化するために、従来のアップル社の Q u i c k T i m e V R が具現化されたのと同様に、位置を見失わないように魚眼レンズをつけて大邸宅を通してカメラを搬送することができる。次いで、種々のフレームが変換され、従って、映像は歪むことなく、映画と共に R A I D アレイ 1 5 1 1 - 1 5 1 2 に記憶され、そしてユーザがバーチャルの塔に行くことを選択したときに再生することができる。

30

【0256】

スポーツ行事では、バスケットボールゲームのような生のスポーツ行事を、ユーザが普通の T V で見るために、ホスティングサービス 2 1 0 を通してストリーミングすることができる。ユーザが特定のプレイを見た後に、ゲームのビデオゲーム（最終的には実際のプレーヤーと同程度ホトリアルに見えるバスケットプレーヤーを伴う）を、プレーヤーが同じ位置でスタートして始めることができ、そしてユーザ（おそらく各々 1 人のプレーヤーの制御権を得ている）がプレイをやり直して、プレーヤーよりうまくいくかどうか確かめることができる。

40

【0257】

ここに述べるホスティングサービス 2 1 0 は、この未来派的な世界をサポートするのに非常に良く適している。というのは、家庭又はほとんどのオフィス設定でインストールするのが不可能である計算パワー及び大量記憶リソースを保持することができ、又、家庭用設定では常に古い世代の P C 及びビデオゲームをもつものとなるが、その計算リソースは、入手可能な最新の計算ハードウェアで、常に、最新のものだからである。そして、ホスティングサービス 2 1 0 では、この計算の複雑さが全てユーザから隠され、従って、非常

50

に精巧なシステムを使用しても、ユーザの観点から、テレビのチャンネルを切り換えるように簡単である。更に、ユーザは、全ての計算パワーにアクセスし、そして計算パワーがクライアント415から得られるところの経験にアクセスすることができる。

【0258】

マルチプレーヤゲーム

ゲームがマルチプレーヤゲームであるという点で、インバウンドルーティング1502のネットワークを通してapp/gameゲームサーバー1521-1525と通信できると共に、インターネット（図示せず）へのネットワークブリッジで、ホスティングサービス210において実行されないサーバー又はゲームマシンと通信することができる。一般的なインターネットにおけるコンピュータでマルチプレーヤゲームをプレイするときには、app/gameゲームサーバー1521-1525は、インターネットへのアクセスが非常に速い（ゲームが家庭のサーバーで実行された場合に比して）という利点を有するが、低速な接続でゲームをプレイする他のコンピュータの能力により制限を受けると共に、インターネット上のゲームサーバーが、比較的低速の消費者用インターネット接続上の家庭用コンピュータである最小公分母を受け容れるように設計されたことによっても潜在的に制限を受ける。

【0259】

しかし、マルチプレーヤゲームが完全にホスティングサービス210のサーバーセンタ内でプレイされるときには、差の世界を達成することができる。ユーザのためのゲームをホストする各app/gameゲームサーバー1521-1525は、他のapp/gameゲームサーバー1521-1525と、非常に高速で非常に短待ち時間の接続及び膨大な非常に高速の記憶アレイをもつマルチプレーヤゲームのための集中制御をホストするサーバーとに相互接続される。例えば、インバウンドルーティング1502のネットワークにギガビットのイーサネットが使用される場合には、app/gameゲームサーバー1521-1525は、ギガビット/秒の速度において潜在的に1ms以下の待ち時間で、互いに通信すると共に、マルチプレーヤゲームのための集中制御をホストするサーバーとも通信する。更に、RAIDアレイ1511-1512は、非常に迅速に応答し、そしてギガビット/秒の速度でデータを転送することができる。例えば、家庭においてPC又はゲームコンソールで実行されるゲームクライアントに制限された従来のシステムで、キャラクタが大量の幾何学形状及びそのキャラクタ独特の挙動を有するように、容貌及び服装に関してユーザがキャラクタをカスタマイズする場合に、そのキャラクタが別のユーザの視野に入ってきた場合には、ユーザは、長くて低速のダウンロードが完了して全ての幾何学形状及び挙動データがコンピュータにロードされるまで、待機しなければならない。ホスティングサービス210内では、その同じダウンロードを、RAIDアレイ1511-1512から対応されるギガビットのイーサネットを経てギガビット/秒の速度で行うことができる。家庭のユーザが8Mbpsのインターネット接続（これは、今日の規格では非常に高速である）を有する場合でも、ギガビットイーサネットは、100倍も高速である。従って、高速インターネット接続で1分かかるものが、ギガビットイーサネットでは1秒未満である。

【0260】

トッププレーヤグループ及びトーナメント

ホスティングサービス210は、トーナメントに非常に良く適している。ローカルクライアントでゲームが実行されないの、ユーザがチートする機会がない（例えば、従来のトーナメントにおいてローカルPCで実行されるゲームのコピーを変更して不公平な利益を与えることにより得られる）。又、出力ルーティング1540がUDPストリームをマルチキャストできるので、ホスティングサービス210は、メジャートーナメントを観衆の中の数千以上の人々に一度にブロードキャストすることができる。

【0261】

実際に、数千のユーザが同じストリーム（例えば、メジャートーナメントのビューを示す）を受信するように普及した幾つかのビデオストリームがあるときは、多くのクライア

ント装置 4 1 5 へ大量配布するために A k a m a i 又は L i m e l i g h t のような「コンテンツデリバリーネットワーク」(C D N)へビデオストリームを送信するのがより効率的である。

【0 2 6 2】

C D Nを使用してトッププレーヤグループの「ゲームファインダー」ページを示すときには、同様の効率レベルを得ることができる。

【0 2 6 3】

メジャートーナメントの場合、生の有名アナウンサを使用して、ある試合中に実況放送することができる。メジャートーナメントを見ているのは大勢のユーザであるが、トーナメントでプレイするのは比較的少数である。有名アナウンサからの音声は、トーナメントでプレイしているユーザをホストすると共にトーナメントにおけるゲームの観客モードコピーをホストする a p p / g a m e ゲームサーバー 1 5 2 1 - 1 5 2 5 ヘルパーティングすることができ、そしてその音声は、ゲーム音声の上にオーバーダビングすることができる。有名アナウンサの映像も、ゲームにオーバーレイし、おそらくは、観客のビューにオーバーレイすることができる。

【0 2 6 4】

ウェブページロードの加速

ワールドワイドウェブ及びその一次トランスポートプロトコルであるハイパーテキストトランスファープロトコル(H T T P)は、ビジネスだけが高速インターネット接続をもちそしてオンラインにいる消費者がダイヤルアップモデム又は I S D N を使用していた時代に考えられ定義されたものである。そのとき、高速接続用の「黄金律(gold standard)」は、1 . 5 M b p s のデータレートを対称的に(即ち、両方向に等しいデータレートで)与える T 1 ラインであった。

【0 2 6 5】

今日、状況は、完全に異なる。開発された世界の多くにおいて D S L 又はケーブルモデム接続を通る平均的な家庭用接続の速度は、T 1 ラインより遥かに高いダウンストリームデータレートを有している。実際に、世界の幾つかの部分では、ファイバー・ツー・ザ・カーブ(fiber-to-the-curb)が、5 0 ないし 1 0 0 M b p s 程度の高さのデータレートを家庭へ運ぶ。

【0 2 6 6】

不都合なことに、H T T P は、これらの急激な速度改善の利点を効果的に取り入れるように構成されてこなかった(又、具現化もされなかった)。ウェブサイトは、リモートサーバーにおけるファイルの集合である。非常に簡単に言えば、H T T P は、第 1 のファイルを要求し、そのファイルがダウンロードされるのを待機し、次いで、第 2 のファイルを要求し、そのファイルがダウンロードされるのを待機し、等々である。実際に、H T T P は、2 つ以上の「オープン接続」を許し、即ち一度に 2 つ以上のファイルを要求できるようにするが、合意された規格(及びウェブサーバーが過負荷にならないよう防止する要望)があるために、非常に僅かなオープン接続しか許されない。更に、ウェブページが構成される方法のために、ブラウザは、多くの場合、直ちにダウンロードに利用できる複数の同時のページに気付かない(即ち、ページをパースした後にのみ、映像と同様に、新たなファイルをダウンロードする必要があることが明らかとなる)。従って、ウェブサイトのファイルは、本質的に 1 つ 1 つロードされる。そして、H T T P によって使用される要求及び応答プロトコルのために、ロードされる各ファイルに関連して(米国で典型的なウェブサーバーにアクセスすると)おおよそ 1 0 0 m s の待ち時間がある。

【0 2 6 7】

比較的低速の接続では、これにより大きな問題が導入されることはない。というのは、ファイルそれ自体のダウンロード時間がウェブページの待ち時間を支配するからである。しかし、接続速度が上がるにつれて、特に、複雑なウェブページでは、問題が起き始める。

【0 2 6 8】

図24に示す例では、典型的な商業的ウェブサイトが示されている（この特定のウェブサイトは、一流運動選手用の靴のブランドからのものである）。このウェブサイトには、54のファイルがある。これらファイルは、HTML、CSS、JPEG、PHP、Java（登録商標）Script、及びFlashファイルを含むと共に、ビデオコンテンツを含む。ページが生になる（即ち、ユーザがそれをクリックして、それを使用し始める）前に、合計1.5Mバイトをロードしなければならない。多数のファイルには、多くの理由がある。1つのこととして、複雑で且つ精巧なウェブページであり、そして別のこととして、ページにアクセスするユーザに関する情報（例えば、ユーザの母国、言語、以前に購入したかどうか、等）に動的に基づいてアッセンブルされるウェブページであり、そしてこれら全てのファクタに基づいて、異なるファイルがダウンロードされる。依然、これは、非常に典型的な商業用ウェブページである。

10

【0269】

図24は、接続速度が上がるにつれてウェブページが生になるまでに経過する時間量を示している。1.5Mbpsの接続速度2401では、従来のウェブブラウザを伴う従来のウェブサーバーを使用すると、ウェブページが生になるまでに13.5秒を要する。12Mbpsの接続速度2402では、ロード時間は、6.5秒に短縮され、即ち約2倍の速さになる。しかし、96Mbpsの接続速度2403では、ロード時間は、約5.5秒までしか短縮されない。その理由として、このように高いダウンロード速度では、ファイルそれ自体をダウンロードするための時間は最小であるが、ファイル当たりの待ち時間、各々約100msが依然あって、その結果、54ファイル $\times$ 100ms=5.4秒の待ち時間となる。従って、家庭への接続がどれほど速くても、このウェブサイトは、常に、それが生となるまでに少なくとも少なくとも5.4秒を要する。別のファクタは、サーバー側のキューイングであり、即ち各HTTP要求がキューの後部に追加され、従って、ビジーのサーバーでは、これが大きな影響を及ぼす。というのは、ウェブサーバーから小さなアイテムを得るたびに、HTTP要求がその方向転換を待機する必要があるためである。

20

【0270】

これらの問題を解決する1つの仕方は、HTTPを破棄するか又は再定義することである。或いは、おそらく、ウェブサイト所有者がそのファイルを単一のファイル（例えば、Adobe Flashフォーマットの）へうまく合併することである。しかし、実際的な問題として、この会社及び多くの他の者は、そのウェブサイトアーキテクチャー多大な投資をしている。更に、ある家庭は、12-100Mbpsの接続を有するが、大多数の家庭は、依然、ゆっくりした速度であり、HTTPも、低速度で機能する。

30

【0271】

1つの別の仕方は、app/gameサーバー1521-1525のウェブブラウザをホストすると共に、RAIDアレイ1511-1512のウェブサーバーに対するファイルをホストすることである（或いは潜在的にウェブブラウザをホストするapp/gameサーバー1521-1525のRAM又はローカル記憶装置において）。インバウンドルーティング1502を通して（又はローカル記憶装置への）非常に高速の相互接続があるために、HTTPを使用してファイル当たり100msの待ち時間となるのではなく、HTTPを使用してファイル当たり最小の待ち時間となる。従って、家にいるユーザがHTTPを通してウェブページにアクセスするのではなく、ユーザは、クライアント415を通してウェブページにアクセスすることができる。次いで、1.5Mbpsの接続では（このウェブページがそのビデオに対して著しい帯域巾を要求しないので）、ウェブページは、線2400当たり1秒未満で生となる。本質的に、app/gameサーバー1521-1525で実行されるウェブブラウザが生ページを表示するまでは待ち時間がなく、そしてクライアント415がウェブブラウザからのビデオ出力を表示するまでは検出可能な待ち時間がない。ユーザがウェブページ上でマウス操作し及び/又はタイプするときに、ユーザの入力情報がapp/gameサーバー1521-1525で実行されるウェブブラウザへ送信され、そしてウェブブラウザが適宜に応答する。

40

【0272】

50

この解決策の1つの欠点は、コンプレッサがビデオデータを常時送信する場合に、ウェブページが静的なものとなっても、帯域巾が使用されることである。これは、ウェブページが変化するときだけ（及びその場合にのみ）データを送信し、次いで、変化したページの部分にのみデータを送信するようにコンプレッサを構成することにより、改善することができる。常時変化するフラッシングバナー、等をもつ幾つかのウェブページがあるが、このようなウェブページは、うっとうしい傾向となり、通常、ウェブページは、何かが動く（例えば、ビデオクリップ）理由がない限り静的なものとなる。このようなウェブページの場合、おそらく、従来のウェブサーバーよりも少ないデータがホスティングサービス210を使用して送信される。というのは、実際に表示される映像だけが送信され、シンクライアント実行コードも、決して見ることもない大きなオブジェクト、例えば、ロールオーバー映像も、送信されないからである。

10

【0273】

従って、ホスティングサービス210を使用して、レガシーウェブページをホストすると、ウェブページのロード時間は、ウェブページをオープンすることがテレビのチャンネルを切り換えることと同じである点まで短縮することができ、即ちウェブページは、実際上瞬時に生となる。

【0274】

ゲーム及びアプリケーションのデバッグの容易化

上述したように、リアルタイムグラフィックを伴うビデオゲーム及びアプリケーションは、非常に複雑なアプリケーションであり、典型的に、市場に発売されるときに、バグを含むことがある。ソフトウェア開発者は、バグに関してユーザからフィードバックを得、そしてクラッシュ後にマシン状態を還元するための何らかの手段を有するが、ゲーム又はリアルタイムアプリケーションに何が起きてクラッシュさせ又は不適切に機能させたか厳密に識別することは非常に困難である。

20

【0275】

ゲーム又はアプリケーションがホスティングサービス210で実行されるときに、ゲーム又はアプリケーションのビデオ／オーディオ出力は、遅延バッファ1515に常時記録される。更に、ウォッチドッグプロセスが、各app/gameサーバー1521-1525で動作し、app/gameサーバー1521-1525が円滑に動作することをホスティングサービスコントロールシステム401に定期的に報告する。ウォッチドッグプロセスが報告を怠った場合には、サーバーコントロールシステム401は、app/gameサーバー1521-1525と通信するように試み、それが成功すれば、どんなマシン状態が得られてもそれを収集する。どんな情報が得られてもそれが、遅延バッファ1515に記録されたビデオ／オーディオと共に、ソフトウェア開発者へ送信される。

30

【0276】

従って、ゲーム又はアプリケーションソフトウェアの開発者が、ホスティングサービス210からクラッシュの通知を得ると、何がクラッシュを招いたかのフレームごとの記録を得る。この情報は、バグを追跡して修理する上で非常に価値がある。

【0277】

又、app/gameサーバー1521-1525がクラッシュすると、サーバーは、最も最近の再スタート可能な点において再スタートされ、そして技術的な困難さについて謝罪するメッセージがユーザに与えられる。

40

【0278】

リソースの共有及びコストの節約

図4a及び4bに示されたシステムは、エンドユーザとゲーム及びアプリケーションの開発者の両方に種々の利益を与える。例えば、典型的に、家庭及びオフィスのクライアントシステム（例えば、PC又はゲームコンソール）は、一週間に僅かな時間しか使用されない。ニールセンエンターテイメント“Active Gamer Benchmark Study” (<http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/10-05-2006/0004446115&E> DATE=) による2006年10月5日プレスリリースによれば、積極的なゲーマーは、一週

50

間に平均 14 時間を費やしてビデオゲームコンソールをプレイし、そして一週間に約 17 時間をハンドヘルド状態にしている。又、このレポートは、(コンソール、ハンドヘルド及び PC ゲームプレイを含む) 全てのゲームプレイ活動に対し、積極的なゲーマーは、一週間に平均 13 時間を費やすと述べている。コンソールビデオゲームのプレイ時間が高い数値であることを考慮すれば、一週間は、 $24 \times 7 = 168$ 時間であり、これは、積極的なゲーマーの家では、ビデオゲームコンソールが一週間の時間のうち $17 / 168 = 10\%$ しか使用されないことを意味する。即ち、90%の時間は、ビデオゲームコンソールがアイドル状態である。高いコストのビデオゲームコンソールであり、そして製造者がこのような装置に助成しているとすれば、高価なリソースの非常に非効率な使い方である。会社の中の PC も、典型的に、一週間のうち僅かな時間しか使用されず、特に、Auto desk Maya のようなハイエンドアプリケーションにしばしば要求される非ポータブルのデスクトップ PC がそうである。ある会社は、全ての時間及び休日も営業し、そしてある PC (例えば、夜家で仕事をするために持ち運ばれるポータブル) は、全ての時間及び休日も使用されるが、会社のほとんどの活動は、月曜日から金曜日まで所定の営業時間ゾーンのほぼ AM 9 時から PM 5 時までを中心とする傾向があり、休日や休息時間(昼食時のような)にはほとんど行われず、又、ほとんどの PC 使用は、ユーザが PC に積極的に関わる間に生じるので、デスクトップ PC の使用は、これらの営業時間に従う傾向となる。PC が、週 5 日間、AM 9 時から PM 5 時まで恒常的に使用されると仮定すれば、PC が週の時間の $40 / 168 = 24\%$ 使用されることを意味する。高性能デスクトップ PC は、会社にとって非常個高価な投資であり、これは、非常に低い利用レベルを反映している。デスクトップコンピュータで授業する学校は、週のより小さな時間巾でもコンピュータを使用し、授業時間に基づいて変化するが、ほとんどの授業は、月曜日から金曜日までの昼間の時間中に行われる。従って、一般的に、PC 及びビデオゲームコンソールは、週の時間の僅かな部分でしか利用されない。

【0279】

特に、多くの人々は、平日月曜日から金曜日の昼間の時間中は会社や学校にいて、これらの人々は、一般的に、これらの時間中はビデオゲームをプレイせず、従って、ビデオゲームをプレイするのは、一般的に、夜や週末や休日のような他の時間中である。

【0280】

図 4a に示すホスティングサービスの構成が与えられると、上述した 2 つの段落で述べた使用パターンは、リソースの非常に効率的な利用を生じる。明らかなように、特に、ユーザが、精巧な 3D ビデオゲームのような複雑なアプリケーションに対してリアルタイム応答を要求する場合には、所与の時間にホスティングサービス 210 によってサービスを受けることのできるユーザの数に限度がある。しかし、典型的にほとんどの時間アイドル状態にある家庭のビデオゲームコンソールや会社で使用される PC とは異なり、サーバー 402 は、異なる時間に異なるユーザにより再利用することができる。例えば、高性能のデュアル CPU 及びデュアル GPU と大容量の RAM とを伴う高性能サーバー 402 は、平日の AM 9 時から PM 5 時まで会社及び学校により利用することができるが、夜、週末及び休日には精巧なビデオゲームをプレイするゲーマーにより利用することができる。同様に、低性能のアプリケーションは、Celeron CPU をもち、GPU をもたず(又は非常にローエンドの GPU をもち) 且つ制限された RAM をもつ低性能サーバー 402 において会社の時間中に会社及び学校によって利用することができ、そして低性能ゲームは、会社の時間中でないときに低性能サーバー 402 を利用することができる。

【0281】

更に、ここに述べるホスティングサービス構成では、数百万でなくても数千のユーザ間でリソースが効率的に共用される。一般的に、オンラインサービスは、所与の時間にサービスを利用する全ユーザベースの僅かな割合しかもたない。上述したニールセンのビデオゲーム使用統計値を考えると、なぜそうなるか調べることは容易である。積極的なゲーマーがコンソールゲームを一週間に 17 時間しかプレイしない場合に、ゲームのピーク使用時間が、夜(5-AM 12、 $7 \times 5 \text{ 日} = 35 \text{ 時間/週}$)及び週末(AM 8-AM 12、1

6 * 2 = 3 2 時間 / 週) の典型的な非仕事非会社時間中であると仮定すると、1 7 時間のゲームプレイに対して一週間に 3 5 + 3 2 = 6 5 ピーク時間ある。システムに対する厳密なピークユーザ負荷は、多くの理由で推定が困難であり、即ちあるユーザは、オフピーク時間中にプレイし、ある日中の時間にユーザの集団ピークがあり、ピーク時間は、プレイするゲームの形式により影響され (例えば、子供のゲームは、おそらく、夜の早い時間にプレイされ)、等々である。しかし、ゲーマーがプレイする平均時間数が、ゲーマーがおそらくゲームをプレイする日の時間数より遥かに少ないとすれば、所与の時間にホスティングサービス 2 1 0 のユーザ数の一部分がそれを使用するだけである。この分析のためには、ピーク負荷が 1 2 . 5 % であると仮定しなければならない。従って、計算、圧縮及び帯域巾リソースの 1 2 . 5 % が所与の時間に使用されるだけであり、その結果、リソースの再使用により所与の性能レベルのゲームをプレイするように所与のユーザをサポートするのにハードウェアコストの 1 2 . 5 % だけとなる。

10

【0 2 8 2】

更に、あるゲーム及びアプリケーションが他のものより多くの計算パワーを要求するとすれば、ユーザによってプレイされるゲーム又は実行されるアプリケーションに基づいてリソースを動的に割り当てることができる。従って、低性能ゲーム又はアプリケーションを選択するユーザには、低性能 (低廉な) サーバー 4 0 2 が割り当てられ、そして高性能ゲーム又はアプリケーションを選択するユーザには、高性能 (より高価な) サーバー 4 0 2 が割り当てられる。実際に、所与のゲーム又はアプリケーションは、ゲーム又はアプリケーションの低性能区分及び高性能区分を有し、そしてユーザは、ゲーム又はアプリケーションのニーズを満足する最低コストのサーバー 4 0 2 でのユーザの動作を保持するように、ゲーム又はアプリケーションの区分間で、あるサーバー 4 0 2 から別のサーバー 4 0 2 へスイッチすることができる。単一のディスクより遥かに高速の R A I D アレイ 4 0 5 が、低性能のサーバー 4 0 2 でも使用でき、高速ディスク転送レートの利益を得る。従って、プレイされるゲーム又は使用されるアプリケーションの全部にわたるサーバー 4 0 2 当たりの平均コストは、最高性能のゲーム又はアプリケーションをプレイする最も高価なサーバー 4 0 2 のコストよりかなり低く、更に、低性能サーバー 4 0 2 は、R A I D アレイ 4 0 5 からディスク性能の利益を導出する。

20

【0 2 8 3】

更に、ホスティングサービス 2 1 0 のサーバー 4 0 2 は、ネットワークインターフェイス以外のディスク又は周辺インターフェイスをもたない P C マザーボード以上のものではなく、やがて、S A N 4 0 3 への高速ネットワークインターフェイスのみをもつ単一チップへと一体化される。又、R A I D アレイ 4 0 5 は、おそらく、ディスクより遥かに多数のユーザ間で共用され、従って、活動的なユーザ当たりのディスクコストは、1 つのディスクドライブより遥かに低い。この装置は、全て、おそらく、環境的に制御されたサーバールーム環境においてラック内に存在する。サーバー 4 0 2 がフェイルした場合に、ホスティングサービス 2 1 0 において容易に修理又は交換することができる。対照的に、家庭又はオフィスにおける P C 又はゲームコンソールは、叩かれたり落とされたりすることによる適度な摩耗及び引き裂きに生き残ることができねばならず、ハウジングを要求し、少なくとも 1 つのディスクドライブを有し、悪い環境条件 (例えば、他のギアでオーバーヘッド A V キャビネットへ詰め込まれる) に生き残らねばならず、サービス保証を要求し、パッケージされて出荷されねばならず、且つおそらく小売利ざやを徴収する小売店により販売される頑強なスタンドアローン機器でなければならない。更に、P C 又はゲームコンソールは、ほとんどの時間に低性能ゲーム又はアプリケーション (或いはゲーム又はアプリケーションの区分) がプレイされても、将来のある時点で使用される最も計算上強いと予想されるゲーム又はアプリケーションのピーク性能を満足するように構成されねばならない。そして、P C 又はコンソールがフェイルした場合に、それを修理するのは、高価で且つ時間のかかるプロセス (製造者、ユーザ及びソフトウェア開発者に悪影響が及ぶ) である。

30

40

【0 2 8 4】

50

従って、図 4 a に示すシステムがローカル計算リソースに匹敵する経験をユーザに与えるとすれば、家庭、オフィス又は学校のユーザが所与のレベルの計算能力を経験するために、図 4 a に示すアーキテクチャーを通してその計算能力を与えるのにかなり低廉なものとなる。

【0285】

アップグレードの必要性の排除

更に、ユーザは、新たなゲームをプレイし又はより高い性能の新たなアプリケーションを取り扱うために PC 及び／又はコンソールをアップグレードすることに関して心配する必要はもはやない。ホスティングサービス 210 におけるゲーム又はアプリケーションは、そのゲーム又はアプリケーションのためにどんな形式のサーバー 402 が要求されるかに関わらず、ユーザに利用することができ、全てのゲーム及びアプリケーションは、ほぼ即座に（例えば、サーバー 402 上の RAID アレイ 405 又はローカル記憶装置から素早くロードされて）且つ最新の更新及びバグ修理で適切に実行される（即ち、ソフトウェア開発者は、所与のゲーム又はアプリケーションを実行するサーバー 402 にとって理想的なサーバー構成を選択し、次いで、最適なドライバを伴うサーバー 402 を構成することができ、そして時間の経過と共に、開発者は、ホスティングサービス 210 におけるゲーム又はアプリケーションの全てのコピーに更新、バグ修理、等を一度に与えることができる）。実際に、ユーザがホスティングサービス 210 の使用を開始した後に、ユーザは、おそらく、（例えば、更新及び／又はバグ修理を通じて）優れた経験を与え続けるゲーム及びアプリケーションを見出し、そしてユーザは、1 年前に存在しなかったコンピューティング技術（例えば、高性能 GPU）を使用するサービス 210 に新たなゲーム又はアプリケーションを利用できることを 1 年後に発見し、ひいては、1 年後にゲームをプレイし又はアプリケーションを実行する技術を 1 年前に買うことがユーザにとって不可能であったことを発見することになる。ゲームをプレイし又はアプリケーションを実行する計算リソースは、ユーザには見えないから（即ち、ユーザの観点から、ユーザは、テレビのチャンネルを切り換えるのと同様に、ほぼ瞬時に実行を開始するゲーム又はアプリケーションを単に選択するだけであるから）、ユーザのハードウェアは、ユーザが絶えずアップグレードに気付かなくても「アップグレード」されている。

【0286】

バックアップの必要性の排除

会社、学校及び家庭においてユーザに対する別の主たる問題は、バックアップである。ローカル PC 又はビデオゲームコンソールに記憶された情報（例えば、コンソールの場合には、ユーザのゲーム成績及びランキング）は、ディスクがフェイルするか、又は偶発的な消去があった場合に、失われる。PC のための手動又は自動バックアップを与えるアプリケーションは、多数入手でき、ゲームコンソールの状態をバックアップのためにオンラインサーバーへアップロードできるが、ローカルバックアップは、典型的に、別のローカルディスク（又は他の不揮発性記憶装置）へコピーして、これをどこか安全な場所に保管し組織化しなければならない。又、オンラインサービスに対するバックアップは、典型的な低コストインターネット接続を通して利用できるのがゆっくりのアップストリーム速度であるから、しばしば制限がある。図 4 a のホスティングサービス 210 では、RAID アレイ 405 に記憶されるデータは、当業者に良く知られた従来の RAID 構成技術を使用して、ディスクがフェイルした場合にも、データが失われず、そのフェイルしたディスクを収容しているサーバーセンターの専門家が通知を受けて、ディスクを交換し、RAID アレイがもう一度フェイル許容となるよう自動的に更新されるように、構成することができる。更に、全てのディスクドライブが互いに近くにあり、そしてそれらの間には SAN 402 を通して高速ローカルネットワークがあるので、サーバーセンターにおいて全てのディスクシステムを定期的に二次記憶装置でバックアップし、この二次記憶装置をサーバーセンターに保管するか又は離れた場所へ移すことは困難ではない。ホスティングサービス 210 のユーザの視点から、データは、単純に常時セキュアなものであり、バックアップについて考える必要はない。

【0287】

デモへのアクセス

ユーザは、多くの場合に、ゲーム又はアプリケーションを購入する前にそれを試したがる。上述したように、ゲーム及びアプリケーションをデモする従来の手段があるが（「デモ」の動詞形は、デモンストレーションバージョンを試すことを意味し、これは、名詞として「デモ」とも称される）、それらの各々は、制限及び／又は不便さで悩まされている。ホスティングサービス210を使用すると、ユーザがデモを試すのが容易で且つ便利になる。実際に、全てのユーザがユーザインターフェイス（以下に述べるような）を通してモードを選択して、デモを試す。デモは、デモに適したサーバー402へほぼ瞬間的にロードを行い、そして他のゲーム又はアプリケーションと同様に実行するだけである。デモが非常に高性能のサーバー402を要求するか低性能のサーバー402を要求するかに関わらず、又、どんな形式の家庭又はオフィスクライアント415をユーザが使用しても、ユーザの観点から、デモは、きちんと機能する。ゲーム又はアプリケーションデモのソフトウェア発行者は、どんなデモをどれほど長くユーザが試すことが許されるか厳密に制御することができ、そしてもちろん、デモは、デモンストレーションされるゲーム又はアプリケーションのフルバージョンにアクセスする機会をユーザに与えるユーザインターフェイスエレメントを含むことができる。

【0288】

デモは、おそらく、価格未満又は無料で提供されるので、あるユーザは、デモ（特に、繰り返しプレイするのが楽しいゲームデモ）を繰り返し使用することを試みる。ホスティングサービス210は、所与のユーザに対してデモの使用を制限する種々の技術を使用することができる。最も単純な解決策は、ユーザごとにユーザIDを確立して、所与のユーザIDがデモをプレイすることが許される回数を制限することである。しかしながら、ユーザは、ユーザIDが無料の場合には、複数のユーザIDを設定することができる。この問題に対処する1つの技術は、所与のクライアント415がデモをプレイすることが許される回数を制限することである。クライアントがスタンドアローン装置である場合には、装置がシリアルナンバーを有し、ホスティングサービス210は、そのシリアルナンバーをもつクライアントによりデモをアクセスできる回数を制限することができる。クライアント415がPC又は他の装置のソフトウェアとして実行される場合には、ホスティングサービス210によりシリアルナンバーを指定してPCに記憶し、これを使用してデモの使用を制限できるが、PCをユーザが再プログラムでき且つシリアルナンバーを消去又は変更できるとすれば、PCネットワークアダプタ「メディアアクセスコントロール（MAC）」アドレス（及び／又は他のマシン特有の識別子、例えば、ハードドライブのシリアルナンバー、等）のレコードを保持して、それに対してデモの使用を制限するという別のオプションがホスティングサービス210に対して設けられる。しかしながら、ネットワークアダプタのMACアドレスを変更できるとすれば、これは、フルブーフな方法ではない。別の解決策は、所与のIPアドレスに対してデモをプレイできる回数を制限することである。IPアドレスは、ケーブルモデム及びDSLプロバイダーによって周期的に再指定できるが、実際にはあまり頻繁に行われず、IPが、住居DSL又はケーブルモデムアクセスに対するIPアドレスのブロック内にあると決定できる場合には（例えば、ISPにコンタクトすることによって）、典型的に、所与の家庭に対して少数のデモ使用を確立することができる。又、家庭において同じIPアドレスを共有するNATルーターの後方に複数の装置が存在することもあるが、典型的に、住居設定では、このような装置は限定された数だけである。IPアドレスが、会社にサービスするブロック内にある場合には、1つの会社に対して多数のデモを確立することができる。しかし、結局、PCにおけるデモの数を制限する最良の方法は、上述した全ての解決策の組合せである。断固とした技術的に熟練したユーザが、繰り返しプレイされるデモの数を制限できるフルブーフの方法はないが、多数のバリアを生成することで、ほとんどのPCユーザにとってデモシステムを悪用するトラブルは価値がなく、むしろ、デモをそれらが意図されたように使用して新たなゲーム及びアプリケーションを試せるように、十分な抑止力を生み出すことが

できる。

【0289】

学校、会社及び他の施設の利益

特に、図4aに示すシステムを利用する会社、学校及び他の施設に対して著しい利益が得られる。会社及び学校は、PCをインストールし、維持し及びアップグレードすることに関連して著しいコストがかかり、それが、Mayaのような高性能アプリケーションを実行するためのPCになったときには、特にそうである。上述したように、PCは、一般的に、週の時間の僅かな部分しか利用されず、そして家庭の場合と同様に、所与のレベルの性能能力を伴うPCのコストは、オフィス又は学校環境の方がサーバーセンター環境よりも遥かに高い。

10

【0290】

大きな会社又は学校（例えば、大規模な大学）の場合には、そのようなエンティティのIT部門がサーバーセンターを設定し、そしてLANグレード接続を経てリモートアクセスされるコンピュータを維持するのが実際的である。LANを経て又はオフィス間のプライベートな広帯域巾接続を通してコンピュータにリモートアクセスするための解決策は、多数存在する。例えば、Microsoft Windowsターミナルサーバーでは、又はRealVNC社からのVNCのようなバーチャルネットワーク計算アプリケーションを通して、或いはSun Microsystemsからのシンクライアント手段を通して、ユーザは、グラフィック応答時間及びユーザ経験においてある範囲のクオリティでPC又はサーバーへのリモートアクセスを得ることができる。更に、このような自己管理型のサーバーセンターは、典型的に、単一の会社又は学校専用であり、従って、異種のアプリケーション（例えば、娯楽及び会社アプリケーション）が週の異なる時間に同じ計算リソースを利用するときに考えられる使用の重複の利点を取り入れることができない。従って、多くの会社及び学校は、各ユーザへのLAN速度ネットワーク接続を有するサーバーセンターをそれ自身で設定するスケール、リソース又は専門知識に欠ける。実際に、学校及び会社の大部分は、家庭と同じインターネット接続（例えば、DSL、ケーブルモデム）を有する。

20

【0291】

このような組織は、定常的ベース又は周期的ベースのいずれかで非常に高性能の計算の必要性を依然有している。例えば、小さな建築会社は、少数の建築家しかおらず、設計の仕事をするときに計算の必要性が比較的僅かであるが、非常に高性能の3D計算を周期的に必要とする（例えば、クライアントに対する新たな建築設計の3Dフライスルーを創作するときに）。図4aに示すシステムは、このような組織に非常に良く適している。組織は、典型的に非常に安価な、家庭に提供されるものと同じ種類のネットワーク接続（例えば、DSL、ケーブルモデム）以上のものを必要としない。それらは、安価なPCをクライアント415として利用するか、或いはPCを全く省き、制御信号ロジック413及び短待ち時間ビデオ解凍412を簡単に具現化する安価な専用装置を利用するかのものである。これらの特徴は、PCの窃盗や、PC内の専用コンポーネントへのダメージに伴う問題がある学校にとって特に魅力的である。

30

【0292】

このような構成は、このような組織の多数の問題を解決する（そしてこれら効果の多くは、一般的な計算を行う家庭のユーザによっても共用される）。その1つとして、運営コスト（最終的に、実現可能な取引を得るためにある形態でユーザへ還元されねばならない）を非常に低くすることができる。というのは、（a）計算リソースが、週内に異なるピーク使用時間を有する他のアプリケーションと共有され、（b）組織が、必要な時だけ高性能計算リソースにアクセスする（及びそのコストを被る）ことができ、（c）組織が、高性能計算リソースをバックアップ又は維持するためのリソースを設ける必要がないからである。

40

【0293】

海賊行為の排除

50

更に、ゲーム、アプリケーション、双方向映画、等は、今日のように海賊行為を受けることがもはやない。各ゲームは、記憶されて、ホスティングサービスセンター 210 で実行されるので、ユーザには、基礎的なプログラムコードへのアクセス権が与えられず、従って、海賊行為を受けるものは何もない。ユーザがソースコードをコピーする場合でも、ユーザは、標準的なゲームコンソール又は家庭用コンピュータにおいてコードを実行することはできない。これは、標準的なビデオゲームが入手できない中国のような世界の場所に市場を開く。又、ユーザに配布されたゲームのコピーがないので、中古ゲームの再販売はできない。

【0294】

ゲームの開発者にとって、今日、新たな世代のゲームコンソール又は PC が市場に導入されるときに、市場の中断は僅かである。ホスティングサービス 210 は、コンソール又は PC 技術の完全に新たな世代がユーザや開発者にアップグレードを強制し且つゲーム開発者がユーザへのハードウェアプラットフォームの適時の配信に依存する現在の状況（例えば、Play Station 3 の場合には、その導入が 1 年以上遅らされ、開発者は、それが入手できるようになり且つ著しい数のユニットが購入されるまで待たねばならなかった）とは対照的に、ゲームの要件が変化するにつれて時間と共に更に進歩したコンピューティング技術で徐々に更新することができる。

【0295】

ストリーミング双方向ビデオ

以上の説明は、一般的インターネットをベースとする短待ち時間のストリーミング双方向ビデオ（これは、ここで使用されるオーディオもビデオと一緒に暗示的に含む）の新規な基礎的概念により可能となる広範囲なアプリケーションを述べるものである。インターネットを通してストリーミングビデオを与える従来のシステムは、長待ち時間の双方向性で具現化できるアプリケーションを可能にするだけである。例えば、リニアビデオのための再生コントロール（例えば、休止、巻き戻し、早送り）は、長い待ち時間で十分に機能し、リニアビデオフィードの中から選択することができる。そして、上述したように、あるビデオゲームの性質は、それらを長待ち時間でプレイすることを許す。しかし、ストリーミングビデオのための従来の解決策の長待ち時間（又は低圧縮比）は、ストリーミングビデオの潜在的なアプリケーションを甚だしく制限するか、又はそれらの開発を特殊なネットワーク環境へと狭め、そしてそのような環境においても、従来の技術は、ネットワークに実質的な負担を導入する。ここに述べる技術は、インターネットを通る短待ち時間のストリーミング双方向ビデオ、特に、消費者グレードのインターネット接続を通して可能になるもの、で考えられる広範囲なアプリケーションに対してドアを開く。

【0296】

実際上任意の量の計算パワー、任意の量の高速記憶、並びにパワフルなサーバーの中の非常に高速なネットワークでユーザ経験の向上を果たすに十分な、図 4c のクライアント 465 と同程度に小さいクライアント装置で、実際に、新しい計算の時代を可能にする。更に、帯域巾要件は、システムの計算パワーが増大するにつれて拡大しないので（即ち、帯域巾要件は、表示解像度、クオリティ及びフレームレートに結び付いているだけであるので）、ブロードバンドインターネット接続が偏在し（例えば、広く普及した短待ち時間のワイヤレスカバレッジを通して）、信頼性があり、且つ全ユーザのディスプレイ装置 422 のニーズを満足するに十分な広い帯域巾であるときには、典型的な消費者及び会社向けアプリケーションに対して、（Windows、Linux、OS X、等を実行する PC 又は携帯電話のような）シッククライアントが必要か、又は（Adobe Flash 又は Java のような）シンクライアントでもよいか問題となる。

【0297】

ストリーミング双方向ビデオの出現は、計算アーキテクチャーの構造についての仮定を考え直す結果となった。その一例が、図 15 に示すホスティングサービス 210 のサーバーセンター実施形態である。遅延バッファ及び／又はグループビデオ 1550 のためのビデオ経路は、app/game サーバー 1521-1525 のマルチキャストされたスト

リーミング双方向ビデオ出力が、経路1552を経てリアルタイムで、又は経路1551を経て選択可能な遅延の後に、app/gameサーバー1521-1525へフィードバックされる。これは、従来のサーバー又はローカル計算アーキテクチャーでは不可能であるか又は実現不能な広範囲の実際的なアプリケーション（例えば、図16、17及び20に示すもののような）を可能にする。しかし、より一般的なアーキテクチャー上の特徴として、フィードバックループ1550が与えるものは、ストリーミング双方向ビデオレベルでの反復である。というのは、アプリケーションが要求したときに、ビデオを不定にループバックできるからである。これは、従来得られなかった広範囲のアプリケーションの可能性を実現できるようにする。

【0298】

別の重要なアーキテクチャー上の特徴は、ビデオストリームが単一方向のUDPストリームであることである。これは、實際上、ストリーミング双方向ビデオのマルチキャストリングを任意の度合いで行えるようにする（対照的に、TCP/IPストリームの両方向ストリームは、ユーザの数が増えるにつれて前後通信からネットワークに益々大きな渋滞を生じることになる）。マルチキャストリングは、サーバーセンター内で重要な能力である。というのは、システムが、インターネットユーザ（及び実際には世界の人口）の増大するニーズに応答して、1対多又は多対多で、通信を行えるようにするからである。この場合も、ストリーミング双方向ビデオ反復及びマルチキャストリングの両方の使用を示す図16について述べた例は、可能性についての非常に大きな氷山の一角に過ぎない。

【0299】

非トランジットピアリング

一実施形態において、ホスティングサービス210は、ユーザヘインターネットサービスも提供する1つ以上のインターネットサービスプロバイダー（ISP）への1つ以上のピア接続を有し、このようにして、ホスティングサービス210は、ISPネットワーク内に留まる非トランジットルートを通してユーザと通信することができる。例えば、ホスティングサービス210のWANインターフェイス441がコムキャストケーブルコミュニケーションズ社(Comcast Cable Communications Inc.)のネットワークに直結され、そしてユーザの家屋211にコムキャストケーブルモデムでブロードバンドサービスが供給される場合には、ホスティングサービス210とクライアント415との間のルートがコムキャストのネットワーク内に完全に確立される。その潜在的な効果は、通信コストが低く（2つ以上のISPネットワーク間のIPトランジットコストが回避されるので）、接続の信頼性が潜在的に高く（ISPネットワーク間に混雑又は他のトランジット混乱がある場合に）、及び待ち時間が短い（ISPネットワーク間に混雑、非効率的なルート又は他の遅延がある場合に）ことを含む。

【0300】

この実施形態では、クライアント415が、最初に、セッションの始めに、ホスティングサービス210に連絡するときに、ホスティングサービス210は、ユーザの家屋211のIPアドレスを受け取る。例えば、ARIN（American Registry for Internet Numbers）からの利用可能なIPアドレステーブルを使用して、そのIPアドレスが、別のISPを通してIPトランジットせずにユーザの家屋211へルーティングできるホスティングサービス210に接続された特定のISPに割り当てられたものであることを調べる。例えば、IPアドレスが76.21.0.0と76.21.127.255との間である場合には、そのIPアドレスは、コムキャストケーブルコミュニケーションズ社に指定される。この例において、ホスティングサービス210が、コムキャスト、AT&T、及びCox ISPへの接続を維持する場合には、特定のユーザへの最適なルートを与える見込みが最も高いISPとしてコムキャストを選択する。

【0301】

フィードバックを使用するビデオ圧縮

1つの実施形態において、首尾良い（又は不首尾な）タイル及び／又はフレームの配信を指示するためにクライアント装置からホスティングサービスへフィードバックが与えら

れる。次いで、クライアントから与えられるフィードバック情報を使用して、ホスティングサービスにおいてビデオ圧縮動作を調整する。

【0302】

例えば、図25a-bは、クライアント装置205とホスティングサービス210との間にフィードバックチャンネル2501が確立される本発明の一実施形態を示す。フィードバックチャンネル2501は、首尾良く受信されたタイル／フレームのパケット化された確認、及び／又は受信が不首尾なタイル／フレームの指示を送信するためにクライアント装置205によって使用される。

【0303】

1つの実施形態において、各タイル／フレームを首尾良く受信した後に、クライアントは、ホスティングサービス210へ確認メッセージを送信する。この実施形態では、ホスティングサービス210は、指定の期間の後に確認を受信しない場合、及び／又は送信されたものではないその後のタイル／フレームをクライアント装置205が受信したという確認を受信した場合には、パケットロスを検出する。それとは別に又はそれに加えて、クライアント装置205は、パケットロスを検出し、そしてパケットロスの指示を、そのパケットロスにより影響を受けるタイル／フレームの指示と共に、ホスティングサービス210へ送信する。この実施形態では、首尾良く配信されたタイル／フレームの連続的な確認は、要求されない。

【0304】

パケットロスがどのように検出されるかに関わらず、図25a-bに示す実施形態では、映像のためのIタイルの最初のセット（図25aには示さず）を発生した後に、エンコーダは、それに続いて、パケットロスが検出されるまでPタイルしか発生しない。図25aにおいて、2510のような各フレームは、4つの垂直タイルとして示されていることに注意されたい。フレームは、2x2、2x4、4x4、等の異なる構成でタイル化されてもよいし、或いはフレームは、タイルなしに（即ち、1つの大きなタイルとして）全体的にエンコードされてもよい。フレームタイル化構成の以上の例は、本発明のこの実施形態を例示するために取り上げたものである。本発明の基礎となる原理は、特定のフレームタイル構成に限定されない。

【0305】

Pタイルのみを送信することで、上述した全ての理由で（即ち、Pタイルは、一般に、Iタイルより小さい）、チャンネルの帯域巾要求が緩和される。フィードバックチャンネル2501を経てパケットロスが検出されたときには、クライアント装置205におけるデコーダ2502の状態を再初期化するために、図25bに示すように、新たなIタイルがエンコーダ2500により発生される。図示されたように、1つの実施形態では、各個々のエンコードされたフレームにより消費される帯域巾を限定するために、複数のエンコードされたフレームにわたってIタイルが分散される。例えば、各フレームが4個のタイルを含む図25では、4つの連続するエンコードされたフレーム内の異なる位置で単一のIタイルが送信される。

【0306】

エンコーダ2500は、この実施形態について述べる技術を、ここに述べる他のエンコード技術と合成することができる。例えば、検出されたパケットロスに応答してIタイルを発生するのに加えて、エンコーダ2500は、映像シーケンスを適切にレンダリングするのにIタイルが有益である他の環境においてIタイルを発生することができる（例えば、突然のシーン移行に応答して）。

【0307】

図26aは、クライアント装置205とホスティングサービス210との間のフィードバックチャンネル2601に依存する本発明の別の実施形態を示す。検出されたパケットロスに応答して新たなIタイル／フレームを発生するのではなく、この実施形態のエンコーダ2600は、Pタイル／フレームの依存性を調整する。最初の事柄として、この例について述べる特定の細部は、本発明の基礎的な原理に合致する必要がないことに注意され

10

20

30

40

50

たい。例えば、この例は、Pタイル／フレームを使用して説明するが、本発明の基礎的な原理は、特定のエンコードフォーマットに限定されない。

【0308】

図26aにおいて、エンコーダ2600は、複数の非圧縮のタイル／フレーム2605を複数のPタイル／フレーム2606へとエンコードし、そしてそれらPタイル／フレームを、通信チャンネル（例えば、インターネット）を経てクライアント装置205へ送信する。クライアント装置205のデコーダ2602は、Pタイル／フレーム2606をデコードして、複数の解凍されたタイル／フレーム2607を発生する。エンコーダ2600の過去の状態2611は、ホスティングサービス210のメモリ装置2610内に記憶され、そしてデコーダ2602の過去の状態2621は、クライアント装置205のメモリ装置2620内に記憶される。デコーダの「状態」は、MPEG-2及びMPEG-4のようにビデオコード化システムにおいて良く知られた用語である。1つの実施形態では、メモリ内に記憶された過去の「状態」は、以前のPタイル／フレームからの合成データを含む。メモリ2611及び2621は、図26aに示すように、エンコーダ2600及びデコーダ2602から取り外されるのではなく、エンコーダ2600及びデコーダ2602内に一体化されてもよい。更に、例えば、これに限定されないが、ランダムアクセスメモリを含む種々の形式のメモリが使用されてもよい。

【0309】

1つの実施形態において、パケットロスが生じないときには、エンコーダ2600は、各Pタイル／フレームを、手前のPタイル／フレームに依存すべくエンコードする。従って、図26aに使用された表記法で示されたように、Pタイル／フレーム4は、Pタイル／フレーム3（表記 4_3 を使用して識別される）に依存し、Pタイル／フレーム5は、Pタイル／フレーム4（表記 5_4 を使用して識別される）に依存し、そしてPタイル／フレーム6は、Pタイル／フレーム5（表記 6_5 を使用して識別される）に依存する。この例では、エンコーダ2600とデコーダ2602との間の送信中にPタイル／フレーム 4_3 が失われる。このロスには、上述したもの（これに限定されないが）を含む種々の仕方でエンコーダ2600へ送信することができる。例えば、デコーダ2602がタイル／フレームを首尾良く受信し及び／又はデコードするたびに、この情報をデコーダ2602からエンコーダ2600へ通信することができる。エンコーダ2600が、特定のタイル／フレームが受信され及び／又はデコードされたという指示をある期間の後に受信しない場合には、エンコーダ2600は、タイル／フレームが首尾良く受信されなかったと仮定する。それとは別に又はそれに加えて、デコーダ2602は、特定のタイル／フレームが首尾良く受信されなかったときにエンコーダ2600に通知することができる。

【0310】

1つの実施形態において、失われたタイル／フレームがどのように検出されるかに関わらず、それが生じると、エンコーダ2600は、デコーダ2602により首尾良く受信されたことが分かっている最後のタイル／フレームを使用して次のタイル／フレームをエンコードする。図26aに示す例では、タイル／フレーム5及び6は、「首尾良く受信された」と考えられない。というのは、それらは、タイル／フレーム4のロスのためにデコーダ2602により適切にデコードできないからである（即ち、タイル／フレーム5のデコーディングは、タイル／フレーム4に依存し、そしてタイル／フレーム6のデコーディングは、タイル／フレーム5に依存する）。従って、図26aに示す例では、エンコーダ2600は、タイル／フレーム7を、デコーダ2602が適切にデコードできないタイル／フレーム6ではなくて、タイル／フレーム3（最後に首尾良く受信されたタイル／フレーム）に依存すべくエンコードする。図26aには示されていないが、付加的なパケットロスが検出されないと仮定すれば、タイル／フレーム8は、その後、タイル／フレーム7に依存すべくエンコードされ、そしてタイル／フレーム9は、タイル／フレーム8に依存すべくエンコードされる。

【0311】

上述したように、エンコーダ2600及びデコーダ2602は、両方とも、過去のエン

コーダ及びデコーダ状態 2 6 1 1 及び 2 6 2 1 を各々メモリ 2 6 1 0 及び 2 6 2 0 に維持する。従って、タイル／フレーム 7 をエンコードするときに、エンコーダ 2 6 0 0 は、タイル／フレーム 3 に関連した以前のエンコーダ状態をメモリ 2 6 1 0 から検索する。同様に、デコーダ 2 6 0 2 に関連したメモリ 2 6 2 0 は、少なくとも最後に知られた良好なデコーダ状態（この例では、P タイル／フレーム 3 に関連した状態）を記憶する。その結果、デコーダ 2 6 0 2 は、タイル／フレーム 7 をデコードできるようにタイル／フレーム 3 に関連した過去の状態情報を検索する。

【0 3 1 2】

上述した技術の結果として、I タイル／フレームが要求されないので（ストリームのスタートにデコーダ及びエンコーダを初期化する以外）、比較的小さな帯域巾を使用して、リアルタイム、短待ち時間の双方向ビデオをエンコードし、ストリーミングすることができる。更に、デコーダにより発生されるビデオ映像は、失われたタイル／フレーム 4 及び（この失われたタイル／フレーム 4 のために適切にデコードできない）タイル／フレーム 5 及び 6 から生じる望ましからぬ歪を一時的に含むが、この歪は、非常に短い時間中、見ることができる。更に、（フルビデオフレームではなく）タイルが使用される場合には、歪は、レンダリングされるビデオ映像の特定の領域に限定される。

【0 3 1 3】

本発明の一実施形態による方法が図 2 6 b に示されている。2 6 5 0 において、以前に発生されたタイル／フレームに基づいて、タイル／フレームが発生される。2 6 5 1 において、失われたタイル／フレームが検出される。1 つの実施形態では、失われたタイル／フレームは、上述したように、エンコーダからデコーダへ通信される情報に基づいて検出される。2 6 5 2 において、首尾よく受信され及び／又はデコーダでデコードされたと分かっているタイル／フレームに基づいて次のタイル／フレームが発生される。1 つの実施形態では、エンコーダは、首尾良く受信され及び／又はデコードされたタイル／フレームに関連した状態をメモリからロードすることにより次のタイル／フレームを発生する。同様に、デコーダは、新たなタイル／フレームを受信すると、首尾良く受信され及び／又はデコードされたタイル／フレームに関連した状態をメモリからロードすることによりタイル／フレームをデコードする。

【0 3 1 4】

1 つの実施形態において、首尾良く受信され及び／又はエンコーダにおいてデコードされた最後のタイル／フレームに基づいて、次のタイル／フレームが発生される。別の実施形態では、発生される次のタイル／フレームが I タイル／フレームである。更に別の実施形態では、以前に首尾良く受信されたタイル／フレームに基づいて次のタイル／フレームを発生するか又は I フレームとして発生するかの選択は、どれほど多くのタイル／フレームが失われたか、及び／又はチャンネルの待ち時間に基づく。比較的少数（例えば、1 又は 2）のタイル／フレームが失われ、そして往復待ち時間が比較的短い（例えば、1 又は 2 フレームの時間）状況では、P タイル／フレームを発生するのが最適である。というのは、首尾良く受信された最後のタイル／フレームと、新たに発生されるタイル／フレームとの間の差が比較的小さいからである。多数のタイル／フレームが失われるか又は往復待ち時間が長い場合には、I タイル／フレームを発生するのが最適である。というのは、首尾良く受信された最後のタイル／フレームと、新たに発生されるタイル／フレームとの間の差が大きいからである。1 つの実施形態では、I タイル／フレームを送信するか、P タイル／フレームを送信するか決定するために、タイル／フレームロススレッシュホールド及び／又は待ち時間スレッシュホールド値がセットされる。失われたタイル／フレームの数がタイル／フレームロススレッシュホールドより低く、及び／又は往復待ち時間が待ち時間スレッシュホールド値より低い場合には、新たな I タイル／フレームが発生され、さもなければ、新たな P タイル／フレームが発生される。

【0 3 1 5】

1 つの実施形態において、エンコーダは、常に、首尾良く受信された最後のタイル／フレームに対して P タイル／フレームを発生するように試み、そしてエンコーディングプロ

セスにおいて、Pタイル／フレームがおそらくIタイル／フレームより大きいことをエンコーダが決定した場合には（例えば、それがタイル／フレームの圧縮された1／8を有し、そして圧縮されたサイズが、以前に圧縮された平均Iタイル／フレームのサイズの1／8より大きい場合には）、エンコーダは、Pタイル／フレームの圧縮を断念し、それに代わって、Iタイル／フレームを圧縮する。

【0316】

失われたパケットの発生が頻繁でない場合には、ドロップしたタイル／フレームを報告するためにフィードバックを使用する前記システムは、典型的に、ユーザへのビデオストリームに生じる混乱を非常に僅かなものにする。というのは、エンコーダ2600がタイル／フレームを短時間で圧縮すると仮定すれば、失われたパケットにより混乱を受けるタイル／フレームが、ほぼ1往復の時間に、クライアント装置205とホスティングサービス210との間で置き換わるからである。そして、圧縮される新たなタイル／フレームは、非圧縮のビデオストリームの後期フレームに基づくものであるから、ビデオストリームが非圧縮ビデオフレームの後方に入ることにはない。しかし、新たなタイル／フレームを含むパケットも失われた場合には、少なくとも2往復の遅延を招き、別の新たなタイル／フレームを再び要求及び送信することになり、これは、多くの実際的な状況において、ビデオストリームに著しい混乱を招くことがある。その結果、ドロップしたタイル／フレームの後に送信される新たにエンコードされたタイル／フレームがホスティングサービス210からクライアント装置205へ首尾良く送信されることが非常に重要である。

【0317】

1つの実施形態において、図11a、11b、11c及び11dに示されて上述されたもののような順方向エラー修正（FEC）コード化技術を使用して、新たにエンコードされたタイル／フレームを失う確率を軽減する。タイル／フレームを送信するときにFECコーディングが既に使用されている場合には、新たにエンコードされるタイル／フレームに対してより強いFECコードが使用される。

【0318】

ドロップしたパケットの1つの潜在的な原因は、例えば、ユーザの家屋211におけるブロードバンド接続の他のユーザが多量の帯域巾を使用し始める場合にチャンネル帯域巾が突然失われることである。（FECが使用される場合でも）新たに発生されたタイル／フレームもパケットのドロップのために失われる場合には、1つの実施形態において、新たにエンコードされた第2のタイル／フレームがドロップしたことがクライアント415によりホスティングサービス210に通知されるとき、ビデオコンプレッサ404は、その後新たにエンコードされるタイル／フレームをエンコードするときにデータレートを下げる。異なる実施形態では、異なる技術を使用してデータレートが減少される。例えば、1つの実施形態において、このデータレートの低下は、圧縮比を高めることにより、エンコードされたタイル／フレームのクオリティを下げることで、達成される。別の実施形態では、ビデオのフレームレートを下げ（例えば、60fpsから30fpsへ）、それに応じて、データ送信のレートを低速にすることで、データレートが下げられる。一実施形態では、データレートを下げるために両方の技術が使用される（例えば、両方とも、フレームレートを下げ、圧縮比を上げる）。データ送信のこの低いレートが、パケットのドロップを首尾良く軽減する場合には、上述したチャンネルデータレート検出及び調整方法により、ホスティングサービス210は、低いデータレートでエンコーディングを続け、次いで、チャンネルが許すとき、データレートを徐々にアップ方向又はダウン方向に調整する。ドロップしたパケット及び／又は待ち時間に関連したフィードバックデータを受信し続けることで、ホスティングサービス210は、現在チャンネル状態に基づいてデータレートを動的に調整することが許される。

【0319】

オンラインゲームシステムにおける状態管理

本発明の一実施形態は、能動的ゲームの現在状態を効率的に記憶してサーバー間で搬送する技術を使用する。ここに述べる実施形態は、オンラインゲームに係るものであるが、

本発明の基礎的な原理は、種々の他の形式のアプリケーション（例えば、設計アプリケーション、ワードプロセッサ、eメール又はインスタントメッセージングのような通信ソフトウェア、等）にも使用することができる。図27aは、この実施形態を具現化するための規範的なシステムアーキテクチャーを示し、図27bは、規範的な方法を示す。この方法及びシステムアーキテクチャーは、同時に説明するが、図27bに示す方法は、特定のシステムアーキテクチャーに限定されない。

【0320】

図27bの2751において、ユーザは、クライアント装置205からホスティングサービス210の新たなオンラインゲームを開始する。それに応答して、2752において、ゲーム2702aの「クリーン」映像が記憶装置（例えば、ゲームを実行するサーバーに直結されるか、又はネットワークを通してサーバーに接続されるかに関わらず、ハードドライブ）から、ホスティングサービス210aのメモリ（例えば、RAM）へロードされる。「クリーン」映像は、（例えば、ゲームを初めて実行するときのように）ゲームのプレイを開始する前のゲームのためのランタイムプログラムコード及びデータを含む。ユーザは、次いで、2753において、ゲームをプレイし、「クリーン」映像を非クリーン映像（例えば、図27aにおいて「状態A」で表された実行ゲーム）へ変化させる。2754において、ユーザ又はホスティングサービス210aによってゲームが休止又は終了される。2755において、ホスティングサービス210aの状態管理ロジック2700aは、ゲームの「クリーン」映像と現在ゲーム状態（状態A）との間の相違を決定する。例えば、Unixオペレーティングシステムで利用できる良く知られた“diff”ユーティリティに使用されるものを含む2つのバイナリ映像間の差を計算するために、種々の既知の技術を使用することができる。もちろん、本発明の基礎的な原理は、差を計算するための特定の技術に限定されない。

【0321】

どのように差を計算するかに関わらず、それらが計算されると、差のデータが記憶装置2705a内にローカル記憶され、及び／又は異なるホスティングサービス210bへ送信される。異なるホスティングサービス210bへ送信される場合には、差のデータが、新たなホスティングサービス210bにおける記憶装置（図示せず）に記憶される。いずれの場合にも、差のデータは、ホスティングサービスにおけるユーザのアカウントに関連付けられ、ユーザがホスティングサービスにログインしてゲームを開始する次のときに、それが識別される。1つの実施形態では、直ちに送信されるのではなく、ユーザがゲームのプレイを試みる（且つ異なるホスティングサービスが、ゲームをホストする最良の選択肢として識別される）次のときまで、差のデータは、新たなホスティングサービスへ送信されない。

【0322】

図27bに示す方法に戻ると、2757において、ユーザは、ユーザが最初にゲームをプレイした同じクライアント装置205であるか、又は異なるクライアント装置（図示せず）であるクライアント装置からゲームを再開する。それに応答して、2758において、ホスティングサービス210bの状態管理ロジック2700bは、ゲームの「クリーン」映像を記憶装置及び差のデータから検索する。2759において、状態管理ロジック2700bは、クリーン映像及び差のデータを合成して、ゲームがオリジナルのホスティングサービス210aにあったときの状態（状態A）を再構成する。例えば、Unixオペレーティングシステムで利用できる良く知られた“パッチ”ユーティリティに使用されるものを含む差のデータを使用してバイナリ映像の状態を再構成するために、種々の既知の技術を使用することができる。PCバックアップのような良く知られたバックアッププログラムに使用される差の計算技術を使用してもよい。本発明の基礎的な原理は、差のデータを使用してバイナリ映像を再生成する特定の技術に限定されない。

【0323】

加えて、2760において、プラットフォーム依存データ2710が最終的なゲーム映像2701bに合体される。プラットフォーム依存データ2710は、行先サーバープラットフォーム

ホームに対して独特のデータを含む。例えば、これに限定されないが、プラットホーム依存データ 2710 は、新たなプラットホームのメディアアクセスコントロール (MAC) アドレス、TCP/IP アドレス、1 日の時刻、ハードウェアシリアル番号 (例えば、ハードドライブ及び CPU のための)、ネットワークサーバーアドレス (例えば、DHCP/Wins サーバー)、及びソフトウェアシリアル番号/アクチベーションコード (オペレーティングシステムシリアル番号/アクチベーションコードを含む) を含む。

【0324】

クライアント/ユーザに関連した他のプラットホーム依存データは、次のものを含む (これに限定されない)。

【0325】

1. ユーザスクリーンの解像度。ユーザがゲームを再開するときに、ユーザは、異なる解像度の異なる装置を使用してもよい。

【0326】

2. ユーザコントローラの構成。ゲームが再開するとき、ユーザは、ゲームコントローラからキーボード/マウスへスイッチすることができる。

【0327】

3. 割引率が終了になったかどうか (例えば、ユーザが促進期間中にゲームをプレイしていたが、今や、通常期間の高いコストでプレイする場合)、又はユーザ又は装置に何らかの年齢制約があるかどうか (例えば、ユーザの親が子供に対する設定を切り換え、子供が成人向け資料を見られないようにする)、又はゲームをプレイする装置 (例えば、パブリックライブラリのコンピュータ) が、成人向け資料を表示できるかどうかについて何らかの制約を有するかどうか、のようなユーザの資格。

【0328】

4. ユーザのランク。ユーザは、あるリーグにおいてマルチプレーヤゲームをプレイすることが許されているが、他の若干のユーザがユーザランクを越えるために、ユーザは、より低いリーグへ格下げされる。

【0329】

プラットホーム依存データ 2710 の以上の例は、本発明の実施形態を例示するために取り上げられたものである。本発明の基礎的な原理は、プラットホーム依存データの特定のセットに限定されない。

【0330】

図 28 は、第 1 のホスティングサービスにおける状態管理ロジック 2700 a が実行ゲーム 2701 a から差のデータ 2800 をどのように抽出するかグラフで示す。第 2 のホスティングサービスにおける状態管理ロジック 2700 b は、次いで、クリーン映像 2702 b を差のデータ 2800 及びプラットホーム依存データ 2710 と合成し、実行ゲーム 2701 b の状態を再生する。図 28 に一般的に示すように、差のデータのサイズは、全ゲーム映像 2701 a のサイズより著しく小さく、その結果、差のデータだけを記憶/送信することで、著しい量の記憶スペース及び帯域巾が保存される。図 28 には示されていないが、プラットホーム依存データ 2700 は、若干の差のデータを、それが最終的なゲーム映像 2701 b に合体されるときに、オーバーライトすることができる。

【0331】

オンラインビデオゲームの具現化について上述したが、本発明の基礎的な原理は、ビデオゲームに限定されない。例えば、以上の状態管理技術は、任意の形式のオンラインホストアプリケーションの環境内で具現化することができる。

【0332】

クライアントデコーダを維持する技術

本発明の一実施形態において、ホスティングサービス 210 は、ユーザがホスティングサービス 210 への接続を要求するたびに、クライアント装置 205 へ新たなデコーダを送信する。従って、この実施形態では、クライアント装置により使用されるデコーダは、常に、最新のものであり、クライアント装置において具現化されるハードウェア/ソフト

10

20

30

40

50

ウェアに対して独特に仕立てられる。

【0333】

図29に例示されたように、この実施形態では、クライアント装置205に永久的にインストールされるアプリケーションは、デコーダを含まない。むしろ、クライアント装置205がホスティングサービス210に接続されるたびに一時的デコーダ2900のダウンロード及びインストールを管理するのはクライアントダウンローダーアプリケーション2903である。このダウンローダーアプリケーション2903は、ハードウェア、ソフトウェア、ファームウェア、又はその組み合わせで具現化することができる。新たなオンラインセッションに対するユーザ要求に応答して、ダウンローダーアプリケーション2903は、クライアント装置205に関連した情報を、ネットワーク（例えば、インターネット）を経て送信する。この情報は、クライアント装置及び／又はクライアント装置のハードウェア／ソフトウェア構成（例えば、プロセッサ、オペレーティングシステム、等）を識別する識別データを含む。

10

【0334】

この情報に基づき、ホスティングサービス210のダウンローダーアプリケーション2901は、クライアント装置205に使用されるべき適当な一時的デコーダ2900を選択する。ホスティングサービスのダウンローダーアプリケーション2901は、次いで、一時的デコーダ2900を送信し、そしてクライアント装置のダウンローダーアプリケーション2903は、クライアント装置205のデコーダを検証し及び／又はインストールする。エンコーダ2902は、次いで、ここに述べるいずれかの技術を使用してオーディオ／ビデオコンテンツをエンコードし、そしてコンテンツ2910をデコーダ2900へ送信する。新たなデコーダ2900がインストールされると、現在オンラインセッションに対するコンテンツをデコードする（即ち、ここに述べるオーディオ／ビデオ解凍技術の1つ以上を使用して）。一実施形態において、セッションが終了したときに、デコーダ2900がクライアント装置205から除去される（例えば、アンインストールされる）。

20

【0335】

一実施形態では、ダウンローダーアプリケーション2903は、一時的デコーダ2900がダウンロードされるときに、チャンネルにおいて達成できるデータレート（例えば、データをダウンロードするのにどれほどの時間を要するか決定することにより）、チャンネルにおけるパケットロスレート、及びチャンネルの待ち時間のようなチャンネル評価を行うことにより、チャンネルを特徴付ける。ダウンローダーアプリケーション2903は、チャンネル評価を表すチャンネル特徴付けデータを発生する。このチャンネル特徴付けデータは、次いで、クライアント装置205からホスティングサービスのダウンローダー2901へ送信され、このダウンローダーは、チャンネル特徴付けデータを使用して、クライアント装置205へメディアを送信するためにチャンネルをどのように最良に使用するか決定する。

30

【0336】

クライアント装置205は、典型的に、一時的デコーダ2900のダウンロード中にホスティングサービス210へメッセージを返送する。これらのメッセージは、パケットがエラーなく受信されたかエラーがあったかを指示する確認メッセージを含むことができる。加えて、これらのメッセージは、データレート（パケットが受信されるレートに基づいて計算された）、パケットエラーレート（エラーを伴って受信されたと報告されたパケットのパーセンテージに基づく）、及びチャンネルの往復待ち時間（送信された所与のパケットに関するフィードバックをダウンローダー2901が受信するまでに要した時間長さに基づく）に関してダウンローダー2901へフィードバックを与える。

40

【0337】

例えば、データレートが2Mbpsと決定された場合には、ダウンローダーは、データレートが5Mbpsと決定された場合（例えば、60fpsにおいて1280x720）より小さなビデオウィンドウ解像度をエンコーダ2902に対して選択する（例えば、60fpsにおいて640x480）。パケットロスの割合に基づいて、異なる順方向エラ

50

一修正 (F E C) 又はパケット構造を選択することができる。

【0338】

パケットロスが非常に低い場合には、圧縮されたオーディオ及びビデオは、エラー修正なしに送信される。パケットロスが中程度の場合には、圧縮されたオーディオ及びビデオは、エラー修正コード技術 (例えば、図 11 a、11 b、11 c 及び 11 d に示されて上述されたような) で送信される。パケットロスが非常に高い場合には、十分なクオリティのオーディオビジュアルストリームを送信できないと決定され、クライアント装置 205 は、通信チャンネル (即ち、「リンク」) を通してホスティングサービスを利用できないことをユーザに通知するか、又はパケットロスの低いホスティングサービスへの異なるルートを確立するよう試みることができる (以下に述べるように)。

10

【0339】

待ち時間が短い場合には、圧縮されたオーディオ及びビデオを短い待ち時間で送信してセッションを確立することができる。待ち時間が非常に長い (例えば、80 ms より長い) 場合には、短い待ち時間を要求するゲームにとって、クライアント装置 205 は、リンクを通してホスティングサービスを利用できないか、又はリンクは利用できるが、ユーザ入力に対する応答時間が鈍いか又は「遅れる」か、或いは待ち時間の短いホスティングサービスへの異なるルートの確立をユーザが試みることができることを、ユーザに通知することができる (以下に述べるように)。

【0340】

クライアント装置 205 は、ネットワーク (例えば、インターネット) を通る別のルートを経てホスティングサービス 210 への接続を試み、障害が減少されたかどうか (例えば、パケットロスが低くなったか、待ち時間が短くなったか、又はデータレートが高くなったか) 調べることができる。例えば、ホスティングサービス 210 は、地理的に多数の位置 (例えば、ロスアンジェルス及びデンバーのホスティングセンター) からインターネットに接続することができ、おそらく、ロスアンジェルスでは混雑のためにパケットロスが高いが、デンバーでは混雑がない。又、ホスティングサービス 210 は、複数のインターネットサービスプロバイダー (例えば、A T & T 及び C o m c a s t) を通してインターネットに接続することができる。

20

【0341】

クライアント装置 205 と、サービスプロバイダーの 1 つ (例えば、A T & T) との間の混雑又は他の問題のために、パケットロス及び／又は長い待ち時間及び／又は制約のあるデータレートを招くことがある。しかしながら、クライアント装置 205 が別のサービスプロバイダー (例えば、C o m c a s t) を通してホスティングサービス 210 に接続される場合には、混雑の問題なく、及び／又は低いパケットロス及び／又は短い待ち時間及び／又は高いデータレートで接続することができる。従って、クライアント装置 205 は、指定のスレッシュホールド (例えば、指定の期間にわたる指定のドロップパケット数を越えるパケットロス、指定のスレッシュホールドを越える待ち時間、及び／又は一時的デコーダ 2900 をダウンロードする間の指定のスレッシュホールドより低いデータレートを経験した場合に、一実施形態では、(典型的に異なる I P アドレス又は異なるドメイン名へ接続することにより) 別のルートを経てホスティングサービス 210 への再接続を試み、より良い接続が得られるかどうか決定する。

30

40

【0342】

別の接続オプションが尽きた後に接続がまだ受け容れられない障害を経験している場合には、インターネットへのクライアント装置 205 のローカル接続が障害を被っているか、又は適度な待ち時間を達成するのにホスティングサービス 210 から離れ過ぎている。このような場合には、クライアント装置 205 は、リンクを通してホスティングサービスが利用できないか、又は障害を伴う状態でしか利用できないか、及び／又はある形式の短待ち時間ゲーム／アプリケーションしか利用できないことをユーザに通知する。

【0343】

一時的デコーダがダウンロードされている間にホスティングサービス 210 とクライア

50

ント装置 205 との間のリンク特性のこの評価及び潜在的な改善が行われるので、クライアント装置 205 が一時的デコーダ 2900 のダウンロードとリンク特性の評価とに別々に費やす必要のある時間量が短縮される。それでも、別の実施形態では、リンク特性の評価及び潜在的な改善は、クライアント装置 205 により、一時的デコーダ 2900 をダウンロードすることから別々に遂行される（例えば、デコーダプログラムコードではなく、ダミーテストデータを使用することにより）。これが好ましい具現化であることの理由は多数ある。例えば、ある実施形態では、クライアント装置 205 は、その一部分又は全体がハードウェアで具現化される。従って、これらの実施形態では、ダウンロードに必要なソフトウェアデコーダは存在しない。

【0344】

規格ベースのタイルサイズを使用する圧縮

上述したように、タイルベースの圧縮が使用されるときに、本発明の基礎的な原理は、特定のタイルサイズ、形状又は方向に限定されない。例えば、MPEG-2 及び MPEG-4 のような DCT ベースの圧縮システムでは、タイルは、マクロブロック（16 x 16 ピクセルのブロックを典型的に表すビデオ圧縮に使用されるコンポーネント）のサイズである。この実施形態は、タイルと共に作用するための非常に微細な粒度レベルを与える。

【0345】

更に、タイルサイズに関わりなく、種々の形式のタイルパターンを使用することができる。例えば、図 30 は、各 R フレーム 3001-3004 に複数の I タイルが使用される一実施形態を示す。I タイルが各 R フレーム全体に分散されて、4 個の R フレームごとに 1 つの完全な I フレームが発生されるような回転パターンが使用される。このように、I タイルを分散することで、パケットロスの影響が減少される（ロスをディスプレイの小さな領域に制限する）。

【0346】

又、タイルは、基礎的な圧縮アルゴリズムの一体的なネーティブ構造に対するサイズであってもよい。例えば、H.264 圧縮アルゴリズムが使用される場合には、1 つの実施形態において、タイルが H.264 「スライス」のサイズにセットされる。これは、ここに述べる技術を、H.264 及び MPEG-4 のような種々の異なる標準圧縮アルゴリズムの環境に容易に一体化できるようにする。タイルサイズがネーティブな圧縮構造にセットされると、上述したものと同一技術を具現化することができる。

【0347】

ストリーム巻き戻し及び再生動作の技術

図 15 を参照して上述したように、app/game サーバー 1521-1525 により発生される非圧縮のビデオ／オーディオストリーム 1529 は、複数の解像度の共有ハードウェア圧縮 1530 により同時に圧縮されて、複数の圧縮されたビデオ／オーディオストリーム 1539 を生じる。例えば、app/game サーバー 1521 により発生されたビデオ／オーディオストリームは、共有ハードウェア圧縮 1530 により 1280 x 720 x 60 fps で圧縮され、アウトバウンドインターネットトラフィック 1599 としてアウトバウンドルート 1540 を経てユーザへ送信される。その同じビデオ／オーディオストリームは、共有ハードウェア圧縮 1530 により app/game サーバー 1522 への経路 1522 を経て（又は遅延バッファ 1515 を通して）サムネールサイズ（例えば、200 x 113）へ同時にスケールダウンされ、図 16 のサムネールの集合体の 1 つのサムネール 1600 として表示される。サムネール 1600 が図 17 の中間サイズ 1700 を通して図 18 のサイズ 1800（1280 x 720 x 60 fps）へとズームされるときに、サムネールストリームを解凍するのではなく、app/game サーバー 1522 は、app/game サーバー 1521 のユーザへ送られる 1280 x 720 x 60 fps ストリームのコピーを解凍し、そして高解像度ビデオを、それがサムネールサイズからズームされるときに、1280 x 720 サイズへとスケールアップする。この解決策は、1280 x 720 の圧縮されたストリームを、2 回、再使用するという効果を有する。しかし、幾つかの欠点もある。（a）ユーザへ送信される圧縮されたビデオストリー

10

20

30

40

50

ムは、ユーザのインターネット接続のデータスループットが変化する場合には画質が変化し、たとえそのユーザのインターネット接続が変化しなくても、a p p / g a m e サーバー 1 5 2 2 の「見物」ユーザが見る画質が変化することになり、(b) a p p / g a m e サーバー 1 5 2 2 は、処理リソースを使用して、1 2 8 0 x 7 2 0 映像全体を解凍し、次いで、ズーム中に相当に小さいサイズ（例えば、6 4 0 x 3 6 0）を表示するためにその映像をスケーリング（及びおそらく再サンプリングフィルタを適用）しなければならない、(c) インターネット接続帯域巾の制限及び／又はパケットの消失／崩壊のためにフレームがドロップし、そして見物ユーザが遅延バッファ 1 5 1 5 に記録されたビデオを「巻き戻し」及び「休止」する場合には、見物ユーザは、ドロップしたフレームが遅延バッファに見つからないことが分かり（これは、ユーザがフレームごとに「ステップ」する場合には特に明らかであり）、及び (d) 見物ユーザが遅延バッファに記録されたビデオにおける特定のフレームを見出すために巻き戻す場合には、a p p / g a m e サーバー 1 5 2 2 は、遅延バッファに記録されたビデオストリームにおける探し求めるフレームの前に I フレーム又は I タイルを見出し、次いで、希望のフレームに到達するまで全ての P フレーム／タイルを解凍しなければならない。この同じ限定が、ビデオ／オーディオストリームを生で「見物している」ユーザに適用されるだけでなく、ビデオ／オーディオストリームのアーカイブ（例えば、“B r a g C l i p”）コピーを見ているユーザ（ビデオ／オーディオストリームを発生したユーザを含む）にも適用される。

10

【0 3 4 8】

本発明の別の実施形態は、ビデオストリームを 2 つ以上のサイズ及び／又は構造で圧縮することによりこれらの問題に対処する。1 つのストリーム（「生」のストリーム）は、ここに述べるように、ネットワーク接続の特性（例えば、データ帯域巾、パケット信頼性）、及びユーザのローカルクライアント能力（例えば、解凍能力、表示解像度）に基づいて、エンドユーザへのストリームへと最適に圧縮される。他のストリーム（“H Q”ストリームと称される）は、高いクオリティ、1 つ以上の解像度、及びビデオ再生可能な構造で圧縮され、このような H Q ストリームは、ルーティングされて、サーバーセンター 2 1 0 内に記憶される。例えば、1 つの実施形態において、H Q 圧縮ストリームは、R A I D ディスクアレイ 1 5 1 5 に記憶され、そして休止、巻き戻し及び他の再生機能（例えば、他のユーザへ配布されて見られる“B r a g C l i p”）を与えるのに使用される。

20

【0 3 4 9】

図 3 1 a に示すように、本発明の一実施形態は、ストリームの混乱のために、或いは I タイル又は I フレームが（上述した）I タイル又は I フレームよりおそらく小さいと決定されるために、必要でない限り、1 つは I タイル又は I フレーム 3 1 1 0 を周期的に含みそしてもう 1 つは I タイル又は I フレーム 3 1 1 1 を含まない少なくとも 2 つのフォーマットでビデオストリームを圧縮できるエンコーダ 3 1 0 0 を備えている。例えば、ビデオゲームをプレイしている間にユーザへ送信される「生の」ストリーム 3 1 1 1 は、P フレームのみを使用して圧縮される（上述したように、I タイル又は I フレームが必要でも小さいものでもない限り）。加えて、この実施形態のエンコーダ 3 1 0 0 は、生のビデオストリーム 3 1 1 1 を、一実施形態において I タイル又は I フレームを周期的に含む第 2 のフォーマット（又は同様の形式の映像フォーマット）で同時に圧縮する。

30

40

【0 3 5 0】

上述した実施形態は、I タイル、I フレーム、P タイル及び P フレームを使用するが、本発明の基本的な原理は、特定の圧縮アルゴリズムに限定されない。例えば、フレームが以前の又はその後のフレームに依存する任意の形式の映像フォーマットを、P タイル又は P フレームに代わって使用することができる。同様に、以前の又はその後のフレームに依存しない任意の形式の映像フォーマットを、前記 I タイル又は I フレームに置き換えることもできる。

【0 3 5 1】

上述したように、H Q ストリーム 3 1 1 0 は、周期的な I フレーム（例えば、一実施形態では、1 2 フレームごと、等々）を含む。これは、ユーザが記憶されたビデオストリー

50

ムを特定のポイントへ素早く巻き戻すことを希望する場合に、I タイル又はI フレームが要求されるので、意義深いことである。P フレームだけの圧縮されたストリームでは（即ち、シーケンスの第1 フレームがI フレームでない）、デコーダは、（数時間の長さのこともある）シーケンスの第1 フレームへ戻り、ユーザが巻き戻しを希望するポイントまでP フレームを解凍する必要がある。H Q ストリーム 3 1 1 0 に記憶された1 2 フレームごとのI フレームでは、ユーザは、特定のスポットへ巻き戻すことを決心し、その直前のH Q ストリームのI フレームは、希望のフレームより1 2 以下のフレームだけ前である。デコーダの最大デコードレートがリアルタイム（例えば、6 0 フレーム／秒のストリームに対して1／6 0 秒）である場合でも、1 2（フレーム）／6 0（フレーム／秒）＝1／5 秒だけI フレームから離れている。そして、多くの場合に、デコーダは、リアルタイムより非常に速く動作することができ、例えば、2 x リアルタイムで、デコーダは、6 フレーム内に1 2 フレームをデコードすることができ、これは、「巻き戻し」に対する第2の遅延の1／1 0 に過ぎない。言うまでもなく、高速デコーダ（例えば、1 0 x リアルタイム）でも、直前のI フレームが巻き戻しポイントより多数のフレーム分だけ前にある場合には受け容れられない遅延をもつことになる（例えば、「巻き戻し」を行うのに1 時間／1 0＝6 分を要する）。別の実施形態では、周期的なI タイルが使用され、この場合に、ユーザが巻き戻しを求めるとき、デコーダは、直前のI タイルを巻き戻しポイントの前に見出し、次いで、巻き戻しポイントまで全てのタイルがデコードされるまで、そのポイントからそのタイルのデコーディングを開始する。周期的なI タイル又はI フレームは、I フレーム全体を排除するより効率の悪い圧縮となるが、ホスティングサービス 2 1 0 は、典型的に、H Q ストリームを管理するのに充分過ぎるローカル利用帯域巾及び記憶容量を有する。

10

20

【0 3 5 2】

別の実施形態において、エンコーダ 3 1 0 0 は、H Q ストリームを、上述したように、周期的なI タイル又はI フレームで、それに続いて、P タイル又はP フレームでエンコードするが、それに先行して、B タイル又はB フレームでエンコードする。B フレームは、上述したように、I フレームに先行するフレームであり、時間的に後方に作用するI フレームからのフレーム差に基づくものである。B タイルは、I タイルに先行するタイル対応部であり、I タイルから後方に作用するフレーム差に基づくものである。この実施形態において、望ましい巻き戻しポイントがB フレームである（又はB タイルを含む）場合には、デコーダは、直前のI フレーム又はI タイルを見出し、そして希望の巻き戻しポイントがデコードされるまで時間的に後方にデコードし、次いで、ビデオの再生がそのポイントから前方に進むにつれて、デコーダは、前方に向かう次々のフレームにおいて、B フレーム、I フレーム及びP フレーム（又はそれらのタイル対応部）をデコードする。I 及びP 形式に加えてB フレーム又はB タイルを使用する効果は、所与の圧縮比においてしばしば高いクオリティを達成できることである。

30

【0 3 5 3】

更に別の実施形態では、エンコーダ 3 1 0 0 は、H Q ストリームを全てのI フレームとしてエンコードする。この解決策の効果は、各巻き戻しポイントがI フレームであり、その結果、巻き戻しポイントに到達するために他のフレームをデコードする必要がないことである。欠点は、圧縮データレートが、I、P 又はI、P、B ストリームのエンコーディングに比して非常に高いことである。

40

【0 3 5 4】

他のビデオストリーム再生アクション（例えば、高速又は低速巻き戻し、高速又は低速送り、等）は、典型的に、周期的なI フレーム又はI タイルで（単独であるか又はP 及び／又はB 対応部と組み合わせられて）非常に实际的に達成される。というのは、各々の場合に、ストリームは、フレームごとに時間的に前方にではなく、異なるフレーム順序で再生され、その結果、デコーダは、シーケンスにおける特定のしばしば任意のフレームを見出してデコードする必要があるからである。例えば、非常に高速の送り（例えば、1 0 0 x 速度）の場合には、再生される各次々のフレームが、手前のフレームより1 0 0 フレーム

50

後である。10xリアルタイムで動作し、1フレームの時間内に10フレームをデコードするデコーダでも、100x高速送りを達成するには、依然、10x遅過ぎである。一方、上述した周期的なIフレーム又はIタイルでは、デコーダは、次に表示する必要のあるフレームに対して最も近い適用可能なIフレーム又はIタイルを探し、そしてターゲットフレームポイントまでの介在するフレーム又はタイルのみをデコードすることができる。

【0355】

別の実施形態では、Iフレームは、HQストリームにおいて一貫した周期（例えば、常に、8フレームごと）でエンコードされ、そしてIフレームレートより速い高速送り及び巻き戻しに対してユーザに使用できる速度乗数は、Iフレーム周期の厳密な倍数である。例えば、Iフレーム周期が8フレームである場合には、ユーザに使用できる高速送り又は巻き戻しの速度は、1X、2X、3X、4X、8X、16X、64X、128X及び256Xである。Iフレーム周期より速い速度の場合には、デコーダは、最初、その速度でのフレーム数だけ先の最も近いIフレームへ前進ジャンプし（例えば、現在表示されているフレームが、Iフレームより3フレーム前の場合には、128Xにおいて、デコーダは、フレーム128+3フレーム先へジャンプし）、次いで、各次々のフレームに対し、デコーダは、選択された速度として厳密なフレーム数だけジャンプし（例えば、128Xの選択された速度では、デコーダは、128フレームジャンプし）、そのたびにIフレームに厳密に着地する。従って、Iフレーム周期より速い全ての速度がIフレーム周期の厳密な倍数であるとすれば、デコーダは、希望のフレームを探すためにその先行又は後続フレームをデコードする必要がなく、表示フレーム当たり1つのIフレームをデコードするだけでよい。Iフレーム周期よりゆっくりとした速度（例えば、1X、2X、3X、4X）、又はIフレーム周期の倍数でない速い速度の場合には、デコーダは、希望のフレームを表示するために付加的な新たにデコードされるフレームを最低限しか必要としないのはどのフレームか求め、未デコードのIフレーム又は既にデコードされたフレームは、デコードされた形態で依然利用できるようにし（RAM又は他の高速記憶装置において）、次いで、希望のフレームがデコードされ表示されるまで、介在フレームを必要に応じてデコードする。例えば、4Xの高速送りでは、8XのIフレーム周期性をもつI、Pエンコードシーケンスにおいて、現在フレームが、Iフレームの1フレーム後のPフレームである場合には、表示されるべき希望のフレームは、4フレーム後となり、これは、先行するIフレームに続く5番目のPフレームである。現在表示されたフレーム（丁度デコードされた）がスタートポイントとして使用される場合には、デコーダは、希望のフレームを表示するのに更に4個のPフレームをデコードする必要がある。先行するIフレームが使用される場合には、デコーダは、希望のフレームを表示するのに6個のフレーム（Iフレーム及びその後の5個のPフレーム）をデコードする必要がある。（明らかに、この場合には、デコードすべき付加的なフレームを最小にするために、現在表示されているフレームを使用するのが効果的である。）次いで、4フレーム先にある次にデコードされるべきフレームは、Iフレームに続く1番目のPフレームである。このケースでは、現在デコードされたフレームがスタートポイントとして使用される場合に、デコーダは、更に4個のフレーム（2個のPフレーム、Iフレーム及びPフレーム）をデコードする必要がある。しかし、それに代わって、次のIフレームが使用される場合には、デコーダは、Iフレーム及びそれに続くPフレームをデコードするだけでよい。（明らかに、この場合には、デコードすべき付加的なフレームを最小にするために、次のIフレームをスタートポイントとして使用するのが効果的である。）従って、この例では、デコーダは、現在デコードされたフレームをスタートポイントとして使用することと、その後のIフレームをスタートポイントとして使用することを交互に行う。一般に重要なこととして、HQビデオストリームの再生モード（高速送り、巻き戻し又は停止）及び速度に関わらず、デコーダは、Iフレームであるか又は以前にデコードされたフレームであるかのどちらかのフレームでスタートし、その再生モード及び速度に対して表示される各次々のフレームについて希望のフレームを表示するのに、新たにデコードされるフレームを最小数しか必要としない。

【0356】

10

20

30

40

50

図 3 1 b に示すように、ホスティングサービス 2 1 0 の 1 つの実施形態は、H Q ストリーム 3 1 1 0 をリプレイするユーザ要求を管理するためのストリームリプレイロジック 3 1 1 2 を含む。このストリームリプレイロジック 3 1 1 2 は、ビデオ再生コマンド（例えば、休止、巻き戻し、特定のポイントからの再生、等）を含むクライアント要求を受け取り、それらコマンドを解釈し、そして特定のポイントからの H Q ストリーム 3 1 1 0 をデコードする（適宜、I フレーム又は以前にデコードされたフレームのいずれかでスタートし、次いで、前方又は後方に特定の位置まで進む）。1 つの実施形態では、デコードされた H Q ストリームは、エンコーダ 3 1 0 0（一度に 2 つ以上のストリームをエンコードできる場合は潜在的に同じエンコーダ 3 1 0 0、又は個別のエンコーダ 3 1 0 0）へ送られて、再圧縮され（ここに述べる技術を使用して）、そしてクライアント装置 2 0 5 へ送信される。クライアント装置のデコーダ 3 1 0 2 は、次いで、上述したように、ストリームをデコードし、レンダリングする。

10

【0 3 5 7】

1 つの実施形態において、ストリームリプレイロジック 3 1 1 2 は、H Q ストリームをデコードせず、エンコーダ 3 1 0 0 がストリームを再エンコードするようにさせる。むしろ、H Q ストリーム 3 1 1 0 を特定のポイントからクライアント装置 2 0 5 へ単に直接的にストリーミングさせる。次いで、クライアント装置 2 0 5 のデコーダ 3 1 0 2 が H Q ストリームをデコードする。ここに述べる再生機能は、典型的に、リアルタイムビデオゲームをプレイするのと同じ短待ち時間要求を有するものではないので（例えば、プレーヤが能動的にプレイせずに、ゲームプレイの前に単にレビューするだけである場合に）、通常の高品質 H Q ストリームに典型的に本来ある追加の待ち時間は、受け容れられるエンドユーザ経験（例えば、待ち時間は長い、高品質ビデオである）を生じさせる。

20

【0 3 5 8】

例えば、これに限定されないが、ユーザがビデオゲームをプレイする場合に、エンコーダ 3 1 0 0 は、ユーザの接続及びローカルクライアントに対して最適化された本質的に全ての P フレームの生のストリームを与える（例えば、6 4 0 x 3 6 0 解像度においてほぼ 1. 4 M b p s）。又、同時に、エンコーダ 3 1 0 0 は、ビデオストリームをホスティングサービス 3 1 0 内で H Q ストリーム 3 1 1 0 として圧縮し、そしてその H Q ストリームを、例えば、1 2 フレームごとに I フレームを伴い 1 0 M b p s において 1 2 8 0 x 7 2 0 でローカルデジタルビデオデコーダ R A I D アレイに記憶する。ユーザが「休止(Pause)」ボタンに触ると、クライアントの最後にデコードされたフレームでゲームが休止され、スクリーンがフリーズする。次いで、ユーザが「巻き戻し(Rewind)」ボタンに触れると、ストリームリプレイロジック 3 1 1 2 は、上述したように、最も近い I フレーム又は既にデコードされた使用可能なフレームから始めて、D V R R A I D から H Q ストリーム 3 1 1 0 を読み取る。ストリームリプレイロジック 3 1 1 2 は、介在する P 又は B フレームを必要に応じて解凍し、それらフレームを必要に応じて再シーケンシングして、再生シーケンスが希望の巻き戻し速度で後方に進むようにし、次いで、表示されることが意図された希望のデコードされたものを（良く知られた従来の映像スケーリング技術を使用して）1 2 8 0 x 7 2 0 から 6 4 0 x 3 6 0 へサイズ変更し、そして生のストリームエンコーダ 3 1 0 0 は、6 4 0 x 3 6 0 解像度の再シーケンシングされたストリームを再圧縮してユーザへ送信する。ユーザが再び休止し、次いで、ビデオを通して単一ステップして、シーケンスを厳密に見る場合には、D V R R A I D の H Q ストリーム 3 1 1 0 は、単一ステップに各フレームを使用できるようにする（元の生のストリームが、ここに述べる多数の理由のいずれかで、フレームをドロップしても）。更に、ビデオ再生の画質は、H Q ストリームの各ポイントにおいてかなり高いが、生のストリームには、例えば、帯域巾が損なわれて、圧縮映像の画質に一時的な低下を招くポイントが生じ得る。短期間中の、又は動画における損なわれた画質は、ユーザにとっては受け入れられるが、ユーザが特定のフレームで停止し（又はゆっくり単一ステップし）、フレームを厳密に調査する場合には、損なわれた画質を受け容れることができない。又、ユーザには、H Q ストリーム内のポイント（例えば、2 分前）を指定することにより、高速送り又は特定スポットへジャンプす

30

40

50

る能力も与えられる。これら全ての動作は、Pフレームのみであるか、又はIフレームを稀に（又は予期せず）有する生のビデオストリームでは、完全な普遍性及び高い画質において不可能である。

【0359】

一実施形態では、H Qストリームがビデオを記憶する限り、ユーザがビデオストリームを通して前後にスイープできるようにする「スクラバー」（即ち左右のスライダーコントロール）を伴うアップルクイックタイム又はアドベフラッシュビデオウインドウのようなビデオウインドウ（図示せず）がユーザに与えられる。ユーザには、生のストリームを通して「スクラビング」しているように見えるが、実際には、ユーザは、記憶されたH Qストリーム 3 1 1 0を通してスクラビングし、これは、次いで、生のストリームとしてサイズ変更され再圧縮される。更に、上述したように、誰か他の者がH Qストリームを同時に見るか、又はユーザが別の時間に見る場合には、生のストリームの解像度より高い（又は低い）解像度で見ることができ、H Qストリームは、同時にエンコードされ、そしてその画像は、潜在的にH Qストリームの画像までは、見る者の生のストリームの画質と同程度となる。

10

【0360】

従って、生のストリーム（ここに述べるように、短い待ち時間、帯域巾及びパケットエラー許容要求に適するように）、及び高画質ストリーム再生アクション要求を伴うH Qストリームの両方を同時にエンコードすることにより、ユーザには、両シナリオの望ましい構成が与えられる。そして、実際に、ユーザにとって、2つの異なるストリームが異なる仕方でエンコードされることが効果的に良く分かる。ユーザの観点から、非常に変化し易く且つ比較的低い帯域巾のインターネット接続で実行されるにも関わらず、その経験は、短い待ち時間で非常に敏感であり、しかも、デジタルビデオレコーディング（DVR）の機能は、非常に高画質で、柔軟なアクション及び柔軟な速度を伴う。

20

【0361】

上述した技術の結果として、ユーザは、オンラインゲームプレイ又は他のオンライン相互作用の間に、生のストリーム又はH Qストリームのいずれの制限の影響もなく、生のストリーム及びH Qビデオストリームの両方の利益を受ける。

【0362】

図31cは、前記動作を遂行するためのシステムアーキテクチャーの一実施形態を示す。図示されたように、この実施形態では、エンコーダ3100は、一連の「生の」ストリーム 3 1 2 1 L、3 1 2 2 L及び3 1 2 5 Lと、それに対応する一連の「H Q」ストリーム 3 1 2 1 H 1-H 3、3 1 2 2 H 1-H 3及び3 1 2 5 H 1-H 3とを各々エンコードする。各H QストリームH 1は、全解像度でエンコードされ、一方、各エンコーダH 2及びH 3は、ビデオストリームをエンコーディング前の小さなサイズにスケールする。例えば、ビデオストリームが1280 x 720の解像度である場合には、H 1は、1280 x 720の解像度でエンコードし、一方、H 2は、640 x 360へスケールしてその解像度でエンコードし、そしてH 3は、320 x 180へスケールしてその解像度でエンコードする。nを1より大きな整数とすれば、多数の同時H nスケイラー／エンコーダを使用して、多数の同時H Qエンコーディングを種々の解像度で与えることができる。

30

40

【0363】

生のストリームの各々は、上述したように、インバウンドインターネット接続3101を経て受け取ったチャンネルフィールドバック信号3161、3162及び3165に応答して動作する（例えば、図25-26のフィールドバック信号2501及び2601の説明を参照されたい）。生のストリームは、インターネット（又は他のネットワーク）を通りアウトバウンドルーティングロジック3140を経て送信される。生のコンプレッサ3121 L-3125 Lは、チャンネルフィールドバックに基づいて、圧縮されたビデオストリームを適応させるロジック（フレームをスケールし、ドロップさせ、等を含む）を含む。

50

【0364】

HQストリームは、インバウンドルーティングロジック3141及び1502により、信号経路3151を経て内部遅延バッファ（例えば、RAIDアレイ3115）又は他のデータ記憶装置へルーティングされ、及び／又は信号経路3152を経てapp/gameサーバー及びエンコーダ3100へ更なる処理のためにフィードバックされる。上述したように、HQストリーム3121Hn-3125Hnは、その後、要求があった際にエンドユーザへストリーミングされる（例えば、図31b及びそれに関連した説明を参照されたい）。

【0365】

1つの実施形態において、エンコーダ3100は、図15に示す共有ハードウェア圧縮ロジック1530で具現化される。別の実施形態では、エンコーダ及びスケーラーの幾つか又は全部が個々のサブシステムである。本発明の基礎的な原理は、スケーリング又は圧縮リソース或いはハードウェア／ソフトウェア構成の特定の共有に限定されない。

【0366】

図31cの構成の効果は、フルサイズより小さいビデオウィンドウを要求するApp/Gameサーバー3121-3125は、フルサイズウィンドウを処理し解凍する必要がないことである。又、中間のウィンドウサイズを要求するApp/Gameサーバー3121-3125は、希望のウィンドウサイズ付近の圧縮されたストリームを受け取り、次いで、希望のウィンドウサイズへスケールアップ又はダウンすることができる。又、App/Gameサーバー3121-3125が、別のApp/Gameサーバー3121-3125から同じサイズのビデオストリームを要求する場合には、インバウンドルーティング3141は、この技術で良く知られたもののようなIPマルチキャスト技術を具現化し、そして要求を行う各App/Gameサーバーへの独立したストリームを必要とせず、要求されたストリームを複数のApp/Gameサーバー3121-3125へ一度に放送することができる。放送を受信するApp/Gameサーバーがビデオウィンドウのサイズを変更する場合には、異なるビデオサイズの放送へ切り換えることができる。従って、任意の大勢のユーザは、各々、それらのビデオウィンドウをスケーリングしそして希望のウィンドウサイズに厳密にスケーリングされたビデオストリームの利益を常に得るという柔軟性で、App/Gameサーバービデオストリームを同時に見ることができる。

【0367】

図31cに示す解決策に伴う1つの欠点は、ホスティングサービス210の多数の実際的な具現化において、全ての圧縮されたHQストリームの全てのサイズをそのままにしておいて、全ての圧縮されたHQストリームが一度に見られるときがないことである。エンコーダ3100が共有リソースとして具現化されるときには（例えば、ソフトウェア又はハードウェアのいずれかで具現化されるスケーラー／コンプレッサ）、この不経済さが軽減される。しかし、多数の未圧縮のストリームを共通の共有リソースに接続する際には、当該帯域巾のために、実際上の問題が生じる。例えば、各1080p60ストリームは、ほぼ3Gbpsであり、ギガビットイーサネットを遥かに上回る。以下に述べる別の実施形態は、この問題に向けられる。

【0368】

図31dは、各App/Gameサーバー3121-3125に次の2つのコンプレッサが割り当てられたホスティングサービス210の別の実施形態を示す。（1）チャンネルフィードバック3161-3165に基づいて圧縮されたビデオストリームを適応させる生のストリームのコンプレッサ3121-3125L、及び（2）上述したように全解像度のHQストリームを出力するHQストリームコンプレッサ。注意すべきことに、生のコンプレッサは、動的で且つ適応性があり、クライアント205との両方向通信を使用するが、HQストリームは、非適応性で、一方向である。ストリーム間の他の相違は、生のストリームの画質がチャンネル条件及びビデオ資料の性質に基づいて劇的に変化することである。あるフレームは、画質が悪く、ドロップしたフレームが生じ得る。又、生のフレ

ームは、ほぼ完全にPフレーム又はPタイルであり、Iフレーム又はIタイルは稀に表れるだけである。H Qストリームは、典型的に、データレートが生のストリームより相当に高く、いずれのフレームもドロップせずに、一貫した高画質を与える。H Qストリームは、全てIフレームでもよいし、又は頻繁で及び／又は規則的なIフレーム又はIタイルを有してもよい。又、H Qストリームは、Bフレーム又はBタイルを含んでもよい。

【0369】

1つの実施形態において、共有ビデオスケーリング及び再圧縮3142（以下で詳細に述べる）は、上述したルーティングのためにインバウンドルーティング3141へ送信される前に、1つ以上の異なる解像度でスケーリング及び再圧縮されるべきあるH Qビデオストリーム3121H1-3125H1のみを選択する。他のH Qビデオストリームは、上述したルーティングのためにフルサイズでインバウンドルーティング3141へ通されるか、或いは全く通されない。1つの実施形態では、どのH Qストリームがスケーリング及び再圧縮され及び／又はどのH Qストリームが全体に通されるかの判断は、その特定のH Qストリームを特定の解像度（或いはスケーリングされた解像度又は全解像度に近い解像度）で要求しているApp/Gameサーバー3121-3125があるかどうかに基づいて決定される。この手段により、スケーリング及び再圧縮される（又は潜在的に全体に通される）H Qストリームのみが、実際に必要とされるH Qストリームである。ホスティングサービス210の多数のアプリケーションでは、これは、スケーリング及び圧縮リソースの劇的な減少を生じさせる。又、各H Qストリームが、少なくとも、コンプレッサ3121H1-3125H1によりその全解像度で圧縮されるとすると、共有ビデオスケーリング及び再圧縮3142内へルーティングされる必要のある帯域巾は、未圧縮のビデオが受け容れられる場合より劇的に減少される。例えば、3 Gbpsの未圧縮の1080p60ストリームは、10Mbpsへ圧縮されても、依然、非常に高い画質を保持することができる。従って、ギガビットイーサネット接続では、1つの未圧縮の3 Gbpsビデオストリームすら搬送できないのではなく、画質の明らかな低下がほとんどない状態で、多数の10Mbpsのビデオストリームを搬送することができる。

【0370】

図31fは、共有ビデオスケーリング及び再圧縮3142の細部を、多数のH QビデオコンプレッサH Q3121H1-3131H1と共に示している。内部ルーティング3192は、App/Gameサーバー3121-3125からの特定のサイズにスケーリングされた特定のビデオストリームの要求ごとに、典型的に、H QビデオコンプレッサH Q3121H1-3131H1からの圧縮されたH Qストリームのサブセットを選択する。この選択されたストリームサブセット内のストリームは、要求されたストリームがスケーリングされるべき場合には、デコンプレッサ3161-3164を通してルーティングされるか、又は要求されたストリームが全解像度である場合には、非スケーリングビデオ経路3196を経てルーティングされる。スケーリングされるべきストリームは、デコンプレッサ3161-3164により未圧縮ビデオへと解凍され、次いで、その各々がスケーラー3171-3174によって要求されたサイズへとスケーリングされ、次いで、その各々がコンプレッサ3181-3184により圧縮される。特定のH Qストリームが2つ以上の解像度で要求される場合には、内部ルーティング3192は、そのストリームを（当業者に良く知られたIPマルチキャスト技術を使用して）1つ以上のデコンプレッサ3161-3164へ及び（1つが要求されたサイズで、全解像度である場合には）アウトバウンドルーティング3193へマルチキャストすることに注意されたい。要求された全てのストリームは、（コンプレッサ3181-3184から）スケーリングされるか（内部ルーティング3192から）されないかに関わらず、アウトバウンドルーティング3193へ送られる。ルーティング3193は、次いで、各々の要求されたストリームを、それを要求したApp/Gameサーバー3121-3125へ送る。1つの実施形態において、2つ以上のApp/Gameサーバーが同じ解像度の同じストリームを要求する場合には、アウトバウンドルーティング3193が、要求をなす全てのApp/Gameサーバー3121-3125へストリームをマルチキャストする。

【0371】

共有ビデオスケーリング及び再圧縮3142の現在好ましい実施形態では、ルーティングがギガビットイーサネットスイッチを使用して具現化され、そして解凍、スケーリング及び圧縮は、各機能を具現化する個別の特殊な半導体装置により具現化される。同じ機能を、ハードウェアにおける高い一体化レベルで、又は非常に高速なプロセッサで具現化することができる。

【0372】

図31eは、上述した遅延バッファ3115の機能が、共有ビデオ遅延バッファ、スケーリング及び解凍サブシステム3143において具現化されるホスティングサービス210の別の実施形態を示す。このサブシステム3143の細部が図31gに示されている。サブシステム3143の動作は、図31fに示すサブシステム3142と同様であるが、3191は、App/Gameサーバー3121-3125からの要求ごとに、どのHQビデオストリームをルーティングすべきか最初に選択し、次いで、遅延されるべく要求されたHQストリームは、ここに示す好ましい実施形態においてRAIDアレイとして具現化された（しかし、十分な帯域巾及び容量の記憶媒体で具現化することもできる）遅延バッファ3194を通してルーティングされ、そして遅延されるべく要求されなかったストリームは、非遅延ビデオ経路3195を通してルーティングされる。遅延バッファ3194及び非遅延ビデオ3195の両出力は、要求されたストリームをスケーリングすべきか否かに基づいて内部ルーティング3192によりルーティングされる。スケーリングされたストリームは、デコンプレッサ3161-3164、スケーラー3171-3174、及びコンプレッサ3181-3184を通してアウトバウンドルーティング3193へルーティングされ、非スケーリングビデオ3196もアウトバウンドルーティング3193へ送られ、次いで、アウトバウンドルーティング3193は、ユニキャスト又はマルチキャストモードのビデオを、図31fのサブシステム3142について上述したのと同様にApp/Gameサーバーへ送信する。

【0373】

ビデオ遅延バッファ、スケーリング及び解凍サブシステム3143の別の実施形態が図31hに示されている。この実施形態では、HQストリームごとに個々の遅延バッファHQ3121D-HQ3131Dが設けられる。個々の圧縮されたビデオストリームを遅延するのに使用できるRAM及びフラッシュROMのコストが急速に低下しているとすれば、共有遅延バッファ3194を有する場合より費用が安く及び／又は柔軟性が高いことになる。或いは、更に別の実施形態では、単一の遅延バッファ3197（点線で示す）が、高性能の集合リソース（例えば、非常に高速なRAM、フラッシュ又はディスク）において全てのHQストリームに対して個々に遅延を与えることができる。いずれのシナリオにおいても、各遅延バッファHQ3121D-HQ3131Dは、HQビデオソースからのストリームに可変の遅延を与えることができ、又は遅延なしにストリームを通すことができる。別の実施形態では、各遅延バッファは、複数のストリームに異なる遅延量を与えることができる。App/Gameサーバー3121-3125によって全ての遅延又は非遅延が要求される。これらの全てのケースにおいて、遅延及び非遅延ビデオストリーム3198が内部ルーティング3192へ送られ、そして図31gについて上述したように、サブシステム3143の残り部分を通して進む。

【0374】

種々の図31nに対する前記実施形態では、生のストリームは、両方向接続を使用し、そして特定のユーザに対して最小の待ち時間で仕立てられることに注意されたい。HQストリームは、一方向接続を使用し、ユニキャスト及びマルチキャストの両方である。マルチキャスト機能は、これらの図では、大規模なシステムにおけるギガビットイーサネットスイッチで具現化できるような単一ユニットとして示されているが、マルチキャスト機能は、おそらく、複数のスイッチのツリーを通して具現化される。実際に、トップランクのビデオゲームプレーヤからのビデオストリームの場合に、プレーヤのHQストリームは、数百万のユーザから同時に見られることになる。このようなケースでは、おそらく、マル

チキャストH Qストリームを放送する次々の段に多数の個々のスイッチが存在する。

【0375】

診断の目的、及び（ユーザに彼のゲームプレイの成績がどれほど評判の良いものであるか知らせるために）ユーザにフィードバックを与えるため、の両方について、1つの実施形態では、ホスティングサービス210は、App/Gameサーバー3121-3125の各ビデオストリームの同時視聴者がどれほどいるか追跡する。これは、特定のビデオストリームに対しApp/Gameサーバーにより能動的な要求の数のランニングカウントを保持することにより達成できる。従って、100000人の同時視聴者がいるゲームプレーヤは、彼のゲームプレイが非常に評判の良いものであることを知り、そしてゲームプレーヤが良い成績を出して視聴者を引き付けるための刺激を生み出す。（例えば、ビデオゲーム選手権大会の）ビデオストリームが非常に大規模で非常に視聴者数が多いときには、ビデオゲームの試合中に解説者が話をして、そのマルチキャストを見ている何人かの又は全てのユーザがそのコメントを聞くことができるようにするのが望ましい。

10

【0376】

App/Gameサーバーで実行されるアプリケーション及びゲームには、特定の特性（例えば、解像度及び遅延量）をもつ特定のビデオストリームに対する要求をアプリケーション及び／又はゲームが提示できるようにするアプリケーションインターフェイス（API）が設けられる。又、App/Gameサーバーで実行されるオペレーティング環境、又は図4aのホスティングサービスコントロールシステム401へ提示されるこれらのAPIは、種々の理由で、このような要求を拒絶してもよい。例えば、要求されるビデオストリームは、幾つかのライセンス交付権制約を有し（例えば、1人の視聴者しか見ることができず、他の者へは放送されないように）、契約上の制約があり（例えば、視聴者は、ストリームを見る権利に対して支払しなければならない）、年齢制約があり（例えば、視聴者は、ストリームを見るには18歳でなければならない）、プライバシーの制約があり（例えば、アプリケーションを使用するか又はゲームをプレイする個人は、選択された数又はクラスの視聴者（例えば、彼又は彼女の「友人」）のみに視聴を制限するか、或いは全く視聴を許さない）、そして資料の要求が遅延されるという制約がある（例えば、ユーザの位置が暴露されるステルスゲームをユーザがプレイする場合）。ストリームの視聴を制限する多数の他の制約がある。これらケースのいずれかにおいて、App/Gameサーバーによる要求は、拒絶の理由のために拒絶され、そして一実施形態では、（契約に対してどんな料金を支払わねばならないか示す）要求を受け容れるところの代替物で拒絶される。

20

30

【0377】

いずれかの前記実施形態において遅延バッファに記憶されたH Qビデオストリームは、ホスティングサービス210以外の他の行先へエクスポートされる。例えば、特に関心のあるビデオストリームは、YouTubeへエクスポートされるべく、App/Gameサーバーにより（典型的に、ユーザの要求により）要求することができる。このようなケースでは、ビデオストリームは、YouTubeと合意されたフォーマットでインターネットを通して、適当な記述情報（例えば、プレイするユーザの名前、ゲーム、時間、スコア、等）と一緒に送信される。これは、コメントを要求する全てのApp/Gameサーバー3121-3125へ個別のストリームにおいてコメント音声をマルチキャストすることにより具現化できる。App/Gameサーバーは、そのコメントの音声を、当業者に良く知られた音声混合技術を使用して、ユーザの家屋211へ送られる音声ストリームへ合流させる。複数の解説者（例えば、異なる視点又は異なる言語の）がいてもよく、又、それらの中からユーザが選択することができる。

40

【0378】

同様に、個別の音声ストリームを、ホスティングサービス210において、リアルタイムのビデオストリームから又は遅延バッファからの音声と混合し又は置き換えることで、混合することもできるし、又は特定のビデオストリーム（又は個々のストリーム）の音声トラックに代わるものとして働くこともできる。このような音声は、コメント又はナレー

50

ションであるか、或いはビデオストリームにおけるキャラクタの音声を与えることもできる。これは、ユーザによってマシナ（ビデオゲームビデオストリームからのユーザ発生アニメーション）を容易に制作できるようにする。

【0379】

本書の全体にわたって述べたビデオストリームは、App/Gameサーバーのビデオ出力から捕獲され、次いで、ストリーミング及び／又は遅延され、そして種々の仕方でも使用又は配布されるものとして示された。同じ遅延バッファを使用して、非App/Gameサーバーソースから到来するビデオ資料を保持し、そして再生及び配布に対する同程度の柔軟性を、適当な制約と共に与えることができる。このようなソースは、テレビ局からの生のフィードを含む（オーバー・ジ・エア、又はCNNのような非オーバー・ジ・エア、及びHBOのような有料、又は無料のいずれか）。又、このようなソースは、事前に記録された映画又はテレビショー、ホームムービー、広告、及び生のビデオ遠隔会議フィードも含む。生のフィードは、App/Gameサーバーの生の出力のように取り扱われる。事前に記録された資料は、遅延バッファの出力のように取り扱われる。

【0380】

ビデオウインドウのズーミング及び並進移動

図16、17及び18は、サムネールビデオウインドウ1600を中間サイズ1700へそして最終的にはフルサイズのビデオウインドウ1800へズーミングする段階を通してユーザインターフェイスを示している。ズーミングされたビデオウインドウ1600、1700及び1800は、上述したように、図15のApp/Gameサーバー1521-1525により発生される。ユーザが典型的なデスクトップモニタ又はTVスクリーンにおいてビデオウインドウ1600、1700及び1800を見ている場合に、ウインドウがフルサイズへズームされると、その画像は、多くの場合、ユーザがユーザのアプリケーションに対してウインドウのコンテンツを充分詳細に見るに足る大きさとなる。例えば、ビデオウインドウ1800に示されるコンテンツは、ビデオゲームであり、そしてディスプレイ装置が32"TVセット又は23"コンピュータモニタであって、ユーザがディスプレイ装置のための典型的な視聴距離内にいる場合には、ユーザは、ゲームをプレイできるに充分なビデオの細部を見分けることができる。例えば、ユーザは、車1802が前方にある（そしておそらくゲームに基づいて、それを回避し、それに追突し、それに追従し、等）のを見ることができ、そしてユーザは、ゲームをプレイできるに充分な明瞭さで地図1803及びスピードメータ1804の細部を認識することができる。

【0381】

しかしながら、ユーザのディスプレイ装置が非常に小型である場合には、ビデオウインドウ1800も小型であって、ユーザは、ユーザのアプリケーションに対してコンテンツの充分な細部を認識することができない（又は視野の限界があるために）。例えば、ディスプレイ装置が、Apple iPhone（登録商標）のような小型のセルラーホンであるか、又はApple iPod Touchのような小型のメディアプレーヤである場合には、非常に小さなスクリーンサイズ及び／又は低いスクリーン解像度のために、車1802を見分け或いは地図1803又はスピードメータ1804の細部を読み取ることが困難又は不可能となる。そのような場合には、ゲームをプレイすることも困難又は不可能である。

【0382】

1つの実施形態において、サーバー1521-1525は、ユーザ入力に応答して、ビデオウインドウ1800をスクリーンのフルサイズより大きくズームし、及び／又はサーバー1521-1525は、ユーザ入力に応答して、ズームアップスクリーンを水平方向及び／又は垂直方向及び／又は対角方向に並進移動する。これは、ユーザがスクリーンの特定エリアにズームインしてそれをより詳細に見ることができるようにする。例えば、ユーザがスクリーンの中央にズームインすると、車1802は、認識できるに充分な大きさになる。ユーザがスクリーンにズームインし、そして画像を並進移動して左下が見えるようにすると、地図1803が認識可能になる。ユーザがスクリーンの右下にズームインす

ると、スピードメータ1804が認識可能になる。ゲーム又はアプリケーションの性質に基づき、ユーザは、ユーザが休止を希望することをサーバー1521-1525に指示するユーザインターフェイスアクションを遂行することで、ズーム／並進移動の前、その間又はその後にビデオの休止を希望してもよい。例えば、ドライブゲームにおいて、ユーザは、スクリーンの中心へズームインして道路に焦点を合わせる間は休止しないが、地図1803又はスピードメータ1804を見る前に休止して、前方の道路がスクリーン上にもはや見えなくなる間に車に衝突しないようにすることができる。生産アプリケーション又はウェブブラウザアプリケーションの場合には、ズーミング又は並進移動のためにビデオを休止する必要はない。

【0383】

例えば、Apple iPhone、iPod Touch又はMacintoshにおいてタッチスクリーン又はトラックパッドを指で「ピンチ」、「スプレッド」及び／又は「スワイプ」する等の、従来のウインドウをスケーリング及び／又は並進移動するのに使用される種々の従来のユーザインターフェイス技術がある。別の例として、マウスのスクロールホイールを使用して、どれほどズームするか指定できる一方、マウスをクリックして移動することを使用して、並進移動を指定することができる。これら及び他の技術を本発明のビデオウインドウ1800の実施形態に適用して、ユーザがズーム及び／又は並進移動を指定できるようにする。指定されたスケーリング及び／又は並進移動ユーザ入力は、図4aのコントロール信号406としてサーバー1521-1525へ送信され、そして要求されたスケーリング及び／又は並進移動アクションが遂行される。1つの実施形態において、スケーリングされ及び／又は並進移動されたビデオウインドウがディスプレイ装置の縁を越えて延びる程度に対し、サーバー1521-1525により、ディスプレイ装置の縁（又は画像のズームされた部分を含むウインドウの縁）まで切り落とされ、共有ハードウェア圧縮1530によって圧縮された可視ピクセルのみが、図4aにおけるホーム又はオフィスクライアント装置415へ送出され、送信帯域巾を減少させる。別の実施形態では、ディスプレイ装置の縁（又は画像のズームされた部分を含むウインドウの縁）を越えて延びるズームされ及び／又は並進移動されたビデオウインドウの若干又は全部がサーバー1521-1525により発生され、そして共有ハードウェア圧縮1530により圧縮されて、ホーム又はオフィスクライアント装置415へ送出され、次いで、ディスプレイ装置の縁又は画像のズームされた部分を含むウインドウの縁まで切り落とされる。前記実施形態は、必要最小限よりも高い送信帯域巾を生じるが、スケーリング及び／又は並進移動動作は、ホーム又はオフィスクライアント装置415によってローカルで実行され、これは、特に、ホーム又はオフィスクライアント装置415とホストサービス210との間の接続が著しい待ち時間を被る場合に、ズーミング及び／又は並進移動動作の応答時間の待ち時間を減少することができる。

【0384】

1つの実施形態において、スケーリングされ及び／又は並進移動されるビデオウインドウは、ウェブブラウザウインドウである。小さなスクリーンを伴う従来の装置、例えば、Apple iPhone又はiPod Touchのようなセルラーホン又はメディアプレーヤーは、（例えば、「ピンチ」、「スプレッド」及び／又は「スワイプ」により）表示されたウェブサイトのウインドウをズーム及び／又は並進移動して、コンテンツをより厳密に見て、より詳細に認識するか、又は低い詳細度で全体を見るためのファシリティを有するウェブブラウザを備えている。

【0385】

実際の環境において、このような従来のウェブブラウザを使用すると、著しい欠点が生じ、多くの場合、ウェブページのローディングが低速になるか又はネットワークリソースの利用効率が悪くなる。ユーザがウェブページにズームインする場合に、典型的に、ブラウザは、ウェブページコンテンツの全部ではないまでもそのほとんどをダウンロードする必要が依然ある。というのは、ウェブブラウザは、ウェブページのズームイン部分にどんな画像が得られるか決定できる前にウェブページのエレメントの全部又はほとんどを処理

して表示することが典型的に必要なからである。1つの単純な例として、ウェブブラウザが大きな j p e g 画像の一部分にズームインされる場合には、j p e g 圧縮アルゴリズムは、典型的に、画像の一部分を表示できるまでに画像全体を解凍する必要がある。別の例として、H T M L ページは、ウェブページのあるエレメントを他のエレメントの位置に対してしばしば暗示的に又は明確に配置させ、ズームイン部分のエレメントの位置が分かる前にダウンロード及びパースされるべきズームインページの縁から外れるエレメントを必要とする。

【0386】

更に、図24に示して上述したように、ウェブサイトのロード時間は、ファイルのローディングを重畳できない程度まで、ウェブサイトを作り上げる各ファイルをロードする際に被るロード時間の和に漸近線的に近づく。又、ロールオーバーのための j p e g 画像のような多くのエレメントは、ロードされねばならないが、表示されることはない。

【0387】

図24は、54ファイルウェブサイトの種々の接続速度におけるロード時間を示し、各ファイルは、ワイヤラインインターネット接続を使用する H T T P ファイルの共通の待ち時間である 100 m s の待ち時間オーバーヘッドを有する。セルラーネットワーク、例えば、A T & T 3 G セルラーネットワークが使用される場合には、H T T P ファイル待ち時間オーバーヘッドは、典型的に非常に高く、おそらく、400 m s 以上の高さである。従って、図24の規範的ウェブサイトは、5.4秒のロード時間に漸近線的に近づくが、400 m s の H T T P ファイル待ち時間では、同じウェブサイトが 21.6秒のロード時間に漸近線的に近付き、そして 3 G セルラーネットワークがワイヤライン接続より低い接続速度で良く動作するとすれば、合計ロード時間は、30秒以上となる。従って、ズームイン部分しか見えなくても、ウェブブラウザは、ウェブページを表示することができる。

【0388】

これは、ユーザにとって不便だけではなく、ワイヤレスデータスループットの非常に非効率的な使用でもあり、これは、しばしば多数のユーザ間で分担され、そして要求よりしばしば低い（たびたび遥かに低い）ワイヤレスデータスループットが得られる。

【0389】

上述したように、1つの実施形態では、ウェブブラウザは、サーバー 1521-1525 にホストされるか、ホスティングサービス 210 にローカル記憶されたソースウェブページファイルにホストされるか、或いはホスティングサービス 210 に利用できるインターネットへの接続を使用することによりソースファイルにホストされる。次いで、ウェブブラウザウィンドウが表示のためにホーム又はオフィスクライアント装置 415 へ送信される。ウェブブラウザは、コントロール信号 406 を経てユーザのアクションにより制御される。ホーム又はオフィスクライアント装置 415 が、セルラーネットワークを通して動作するセルラーホンである場合には、待ち時間は、典型的に、ホスティングサービス 210 とインターネットとの間よりも、セルラーホンとインターネットとの間の方が相当地に長くなる。従って、ウェブページファイルがホスティングサービス 210 にローカル記憶されるか又はリモートウェブサーバーに位置されるかに関わらず、おそらく、ユーザは、サーバー 1521-1525 にホストされたウェブブラウザによって発生されてセルラーホンへビデオとして送信されるウェブページを見るときには、セルラーネットワークを通してウェブページファイルが慣習的に送信されるようにして、セルラーホンでローカルに実行されるウェブブラウザにより同じウェブページが発生された場合にユーザが経験する遅延より、相当地に短い遅延を経験する。更に、上述したように、ウェブブラウザをサーバー 1521-1525 にホストすることで、おそらく、所与のウェブページに対してホーム又はオフィスクライアント装置 415 へ送信されるデータが著しく少なくなる。又、セルラーネットワークの接続速度が低下する場合には、上述した種々の技術を使用して、送信データスループットを動的に減少させ、セルラーチャンネルの容量を越えないようにすることができる。更に、この解決策は、（例えば、A d o b e F l a s h の最新バージョンで）ローカルウェブブラウザを最新の状態に保つ必要性を排除する。というのは、サ

10

20

30

40

50

ーバー 1 5 2 1－1 5 2 5 で実行されるウェブブラウザが最新の状態を保持できるからである。実際に、サーバー 1 5 2 1－1 5 2 5 で実行されるウェブブラウザは、ホーム又はオフィスクライアント装置 4 1 5 の計算能力を越えるオペレーションを遂行することができる。

【0390】

一実施形態において、ホーム又はオフィスクライアント装置 4 1 5 で発生されたビデオウィンドウは、上述したようにズームされ及び／又は並進移動される。このように、ユーザは、ウェブページの細部を見るためにズーム及び／又は並進移動することができ、或いはズームアウトした場合には、ウェブページの全体図を得ることができる。実際に、ウィンドウ 1 6 0 0 及び 1 7 0 0 のようなスクリーンの一部分だけを取り上げるためにウェブページ画像をズームダウンすることや、又、実際に、他のウィンドウが他のウェブページであるかどうか、又はビデオゲーム、ビデオ、アプリケーション、等のビデオウィンドウであるかどうかに関わらず、ウェブページ画像をそれら他のウィンドウと同時に見ることは、良くあることである。

【0391】

Apple Safari のような従来のウェブブラウザは、例えば、ユーザが最も頻繁に訪問するのはどのウェブサイトであるか示すために、一度に複数の縮小サイズウェブサイトのウィンドウを表示することができる。複数のウェブサイトが同時に表示されるので、それら全部を動的に表示する（例えば、現在ウェブサイトにて表示されているものを示す）ことは、典型的に不可能である。というのは、他の問題の中でも、全てのウェブサイトからの帯域巾需要の和が過剰なものになるからである。例えば、8 個のウェブサイトがビデオを示す場合には、全てのウェブサイトからの全ビデオ帯域巾を受信し、解凍しそして縮小サイズウェブサイトのサイズヘスケールダウンしなければならない。これは、ユーザにとって非リアルタイムの経験を生じさせるだけでなく、帯域巾の浪費にもなる。というのは、全ウェブサイトダウンロードする必要がある、それらがスケールダウンされ、細部が相当失われるからである。

【0392】

1 つの実施形態においては、1 つ以上のウェブブラウザがサーバー 1 5 2 1－1 5 2 5 にホストされ、そしてビデオは、オフィス又はクライアント装置 4 1 5 へ送出される前にスケーリングされ、図 1 6 に示すように、複数のスケールダウンしたビデオウィンドウとして見えるようにする。一度に複数のウェブサイトを縮小サイズウィンドウとして表示できる従来のウェブブラウザとは対照的に、この実施形態では、ウェブサイトは、それらの幾つか又は全部がビデオのような広帯域巾エレメントを組み込んでいても、一度に全部を生で表示することができる。そして、これら複数の生のウェブサイトウィンドウは、ビデオゲーム、ビデオ、アプリケーション、等の他の生のビデオウィンドウと同時に表示することができる。

【0393】

1 つの実施形態において、ユーザにより見るのに利用できるウィンドウの数は、ディスプレイ装置の境界内に表示されるビデオウィンドウの数より多い。例えば、図 1 6 では、1 8 個のビデオウィンドウの 6 x 3 アレイが見えるが、ユーザにより見るのに利用できる遥かに多くのビデオウィンドウがあってもよく（例えば、2 0 x 2 0 アレイ）、所与の時間にディスプレイ装置の境界内に見えるのは 6 x 3 サブセットに過ぎない。1 つの実施形態において、ユーザは、タッチスクリーン又はトラックパッドを横切って指を「スワイプ」させることでビデオウィンドウのアレイの並進移動（例えば、水平方向、垂直方向及び／又は対角方向）を行い、これにより、指タッチでビデオウィンドウのアレイがその位置を並進移動させて他のビデオウィンドウを露呈させるという幻想を生み出すことができる。これは、指スワイプコントロール情報（例えば、スワイプの位置、スワイプの速度、等）をコントロール信号 4 0 6 として送信し、そして app/game サーバー 1 5 2 1－1 5 2 5 がビデオウィンドウのアレイの動きを示すアニメーション効果を具現化することにより、達成することができる。1 つの実施形態において、アニメーション効果は、指スワ

イプ（又は他のユーザ入力、例えば、コントローラアクション又はキーボード／マウスの押圧、身体運動、等）に応答してビデオウオールが水平方向、垂直方向又はある角度で動くことを示す並進移動であるが、ビデオウインドウは、それらが移動するとき全てが同じサイズのままとされる。別の実施形態において、アニメーション効果は、非直線運動であり、ここで、ビデオウオールは、指スワイプ（又は他のユーザ入力アクション）に応答して複雑な動きで移動し、そしてビデオウインドウの幾つか又は全部がアニメーション効果中にサイズを変化させる。1つのこのような複雑な動きは、ビデオウオールが3D空間に置かれているという幻想を生み出すパースペクティブ3D運動である。

【0394】

一実施形態では、ここに示す種々の機能的モジュール及び関連するステップは、そのステップを遂行するための固定布線ロジックを含む特定のハードウェアコンポーネント、例えば、特定用途向け集積回路（ASIC）により遂行されるか、又はプログラムされたコンピュータコンポーネント及びカスタムハードウェアコンポーネントの組合せにより遂行される。

【0395】

一実施形態では、これらモジュールは、テキサスインスツルメント社のTMS320xアーキテクチャー（例えば、TMS320C6000、TMS320C5000、等）のプログラム可能なデジタル信号プロセッサ（DSP）において具現化される。これらの基礎的な原理に適合する種々の異なるDSPを使用することができる。

【0396】

これら実施形態は、上述した種々のステップを含むことができる。これらステップは、汎用プロセッサ又は特殊目的のプロセッサが幾つかのステップを遂行するようにさせるマシン実行可能なインストラクションにおいて実施することができる。これら基礎的な原理に関連しない種々の要素、例えば、コンピュータメモリ、ハードドライブ、入力装置等は、当該観点を不明瞭にしないために図面から除外してある。

【0397】

ここに開示する要旨の要素は、マシン実行可能なインストラクションを記憶するためのマシン読み取り可能なメディアとして与えられてもよい。マシン読み取り可能なメディアは、フラッシュメモリ、光学的ディスク、CD-ROM、DVD-ROM、RAM、EPROM、EEPROM、磁気又は光学カード、電子的インストラクションを記憶するのに適した他の形式のマシン読み取り可能なメディアを含むが、これらに限定されない。

【0398】

又、ここに開示する要旨の要素は、一連のオペレーションを遂行するようにコンピュータ（例えば、プロセッサ又は他の電子装置）をプログラムするのに使用されるインストラクションが記憶されたマシン読み取り可能なメディアを含むコンピュータプログラム製品として提供されてもよいことを理解されたい。或いは又、オペレーションは、ハードウェア及びソフトウェアの組合せにより遂行されてもよい。マシン読み取り可能なメディアは、フロッピー（登録商標）ディスク、光学的ディスク、CD-ROM、及び磁気・光学ディスク、ROM、RAM、EPROM、EEPROM、磁気又は光学カード、電子的インストラクションを記憶するのに適した他の形式のマシン読み取り可能なメディアを含むが、これらに限定されない。

【0399】

更に、ここに開示する要旨は、特定の実施形態に関連して説明したが、本開示の範囲内で多数の変更や修正がなされ得る。従って、本発明及び添付図面は、単なる例示に過ぎず、それに限定されるものではない。

【符号の説明】

【0400】

100：CPU/GPU

101：RAM

102：ディスプレイ装置

10

20

30

40

50

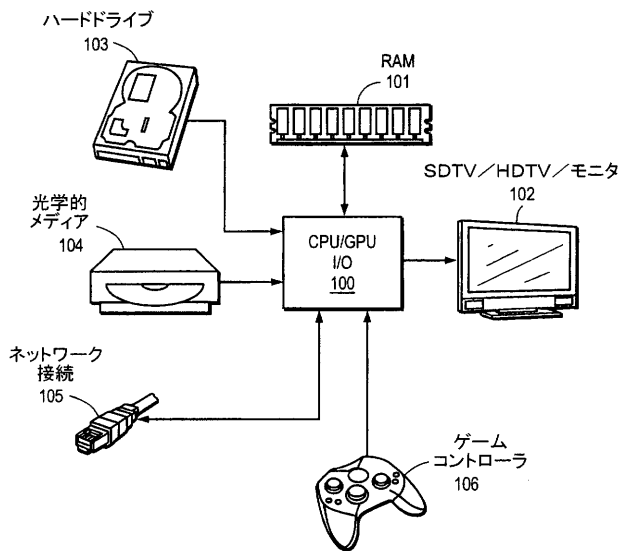
1 0 3	：ハードドライブ	
1 0 4	：光学的メディアドライブ	
1 0 5	：ネットワーク接続	
1 0 6	：ゲームコントローラ	
2 0 5	：クライアント装置	
2 0 6	：インターネット	
2 1 0	：ホスティングサービス	
2 1 1	：ユーザの家屋	
2 2 0	：ソフトウェア開発者	
2 2 1	：入力装置	10
2 2 2	：モニタ又はT V 受像機	
3 0 1	：最大データレート	
3 0 2	：実際に利用できる最大データレート	
3 0 3	：要求されたデータレート	
4 0 1	：ホスティングサービスコントロールシステム	
4 0 2	：サーバー	
4 0 3	：S A N	
4 0 4	：低待ち時間のビデオ圧縮	
4 0 5	：R A I D アレイ	
4 0 6	：コントロール信号	20
4 1 0	：インターネット	
4 1 2	：低待ち時間の解凍	
4 1 3	：コントロール信号ロジック	
4 1 5	：家庭又はオフィスクライアント	
4 2 1	：入力装置	
4 2 2	：モニタ又はH D T V	
4 4 1	：中央オフィス、ヘッドエンド、セルタワー、等	
4 4 2	：W A N インターフェイス	
4 4 3	：ファイアウォール／ルーター／N A T	
4 4 4	：W A N インターフェイス	30
4 5 1	：コントロール信号	
4 5 2	：ユーザの家屋ルーティング	
4 5 3	：ユーザ I S P	
4 5 4	：インターネット	
4 5 5	：サーバーセンタールーティング	
4 5 6	：フレーム計算	
4 5 7	：ビデオ圧縮	
4 5 8	：ビデオ解凍	
4 6 2	：パワーオーバーイーサネット	
4 6 3	：H D M I 出力	40
4 6 4	：ディスプレイ能力	
4 6 5	：イーサネット対H D M I クライアント	
4 6 6	：シャッター付き眼鏡	
4 6 8	：モニタ又はS D / H D T V	
4 6 9	：ブルーーツース入力装置	
4 7 6	：フラッシュ	
4 8 0	：バス	
4 8 1	：イーサネットインターフェイス	
4 8 3	：コントロールC P U	
4 8 4	：ブルーーツース	50

486:ビデオデコンプレッサ
 487:ビデオ出力
 488:オーディオデコンプレッサ
 489:オーディオ出力
 490:HDMI
 497:イーサネット
 499:電力
 2500:エンコーダ
 2501:フィードバックチャンネル
 2502:デコーダ
 2510:フレーム
 2511:デコードされたフレーム
 2600:エンコーダ
 2601:フィードバックチャンネル
 2602:デコーダ
 2605:未圧縮フレーム／タイル
 2606:圧縮されたフレーム／タイル
 2607:解凍されたフレーム／タイル
 2610、2620:メモリ
 2611:過去のエンコーダ状態
 2621:過去のデコーダ状態
 2900:デコーダロジック
 2901:ダウンローダー
 2902:エンコーダ

10

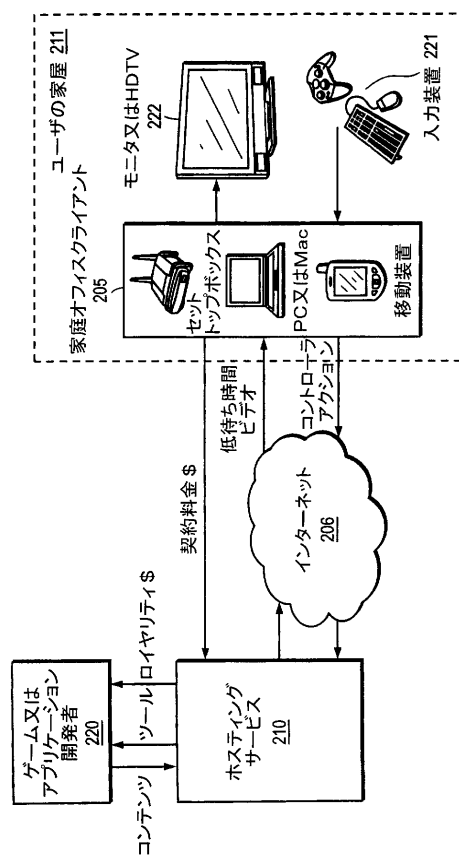
20

【図1】

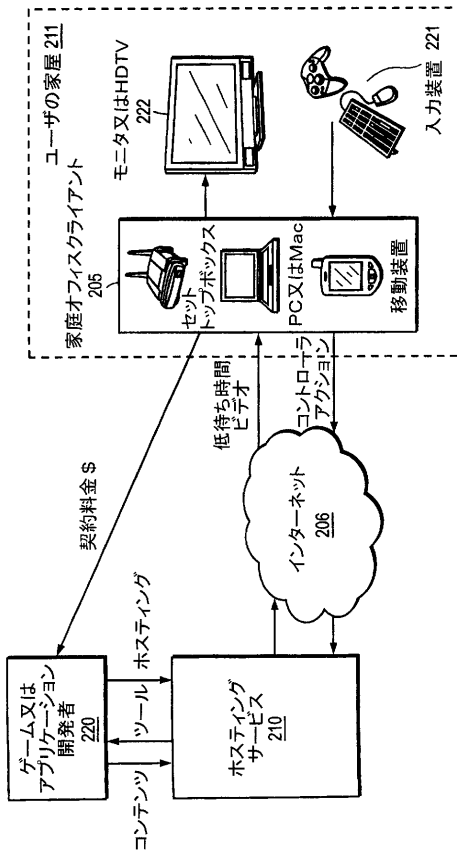


(従来技術)

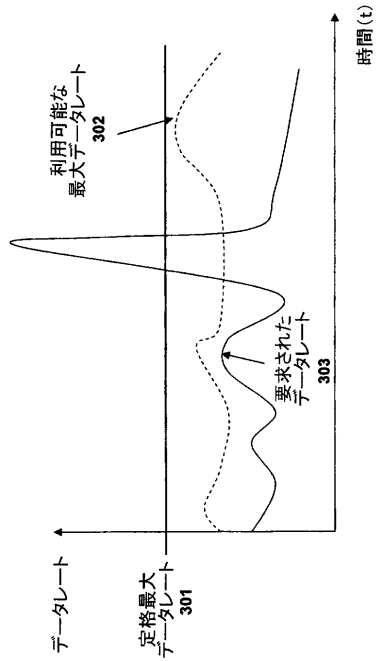
【図2a】



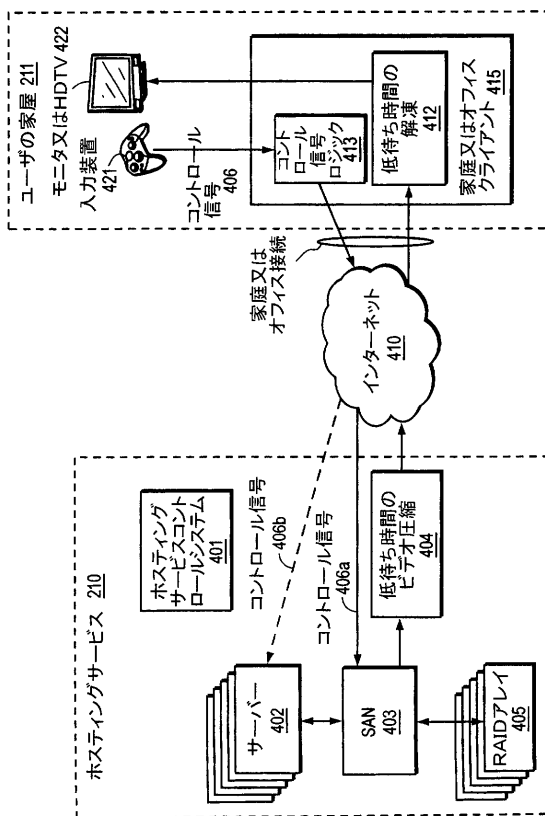
【 図 2 b 】



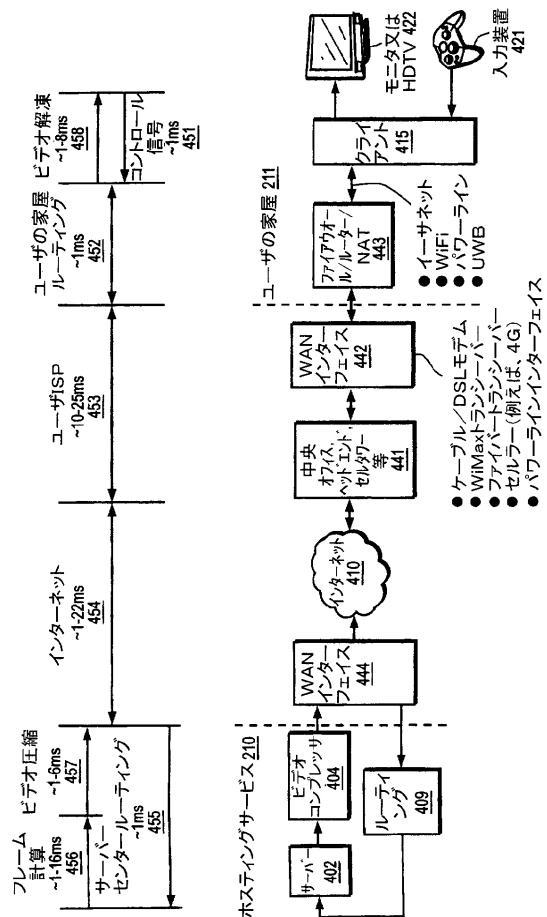
【圖 3】



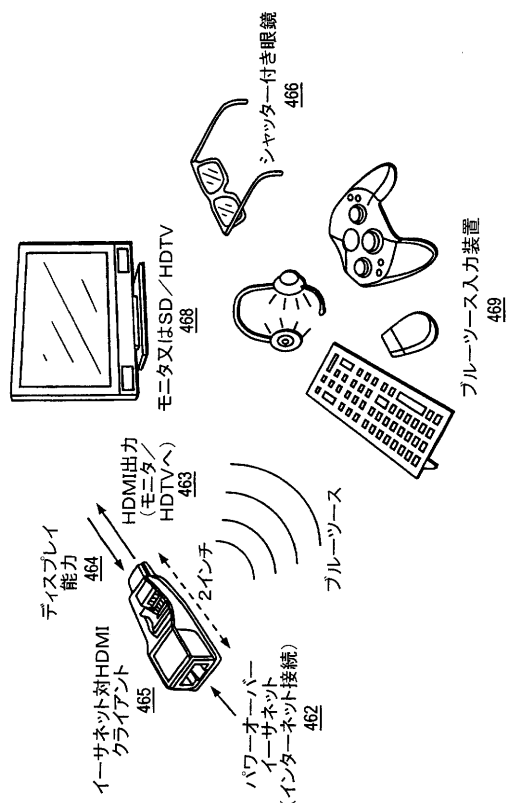
【図 4 a】



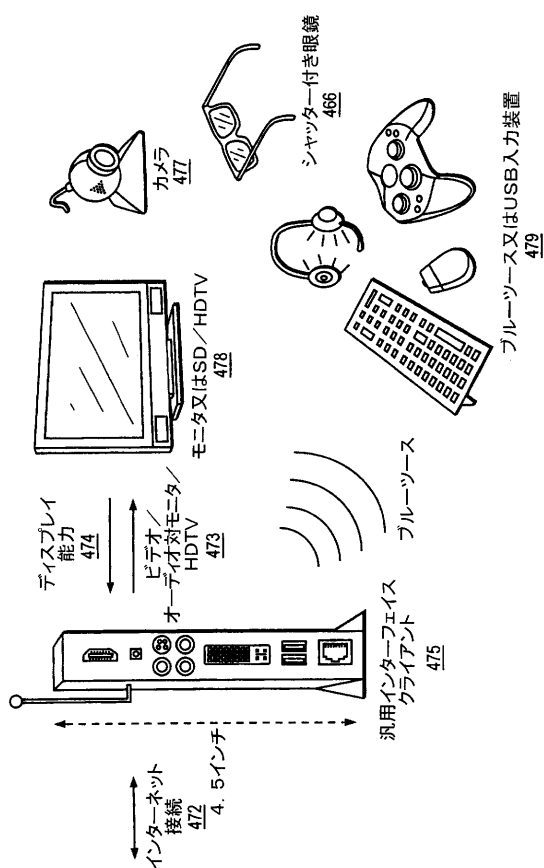
【図 4 b】



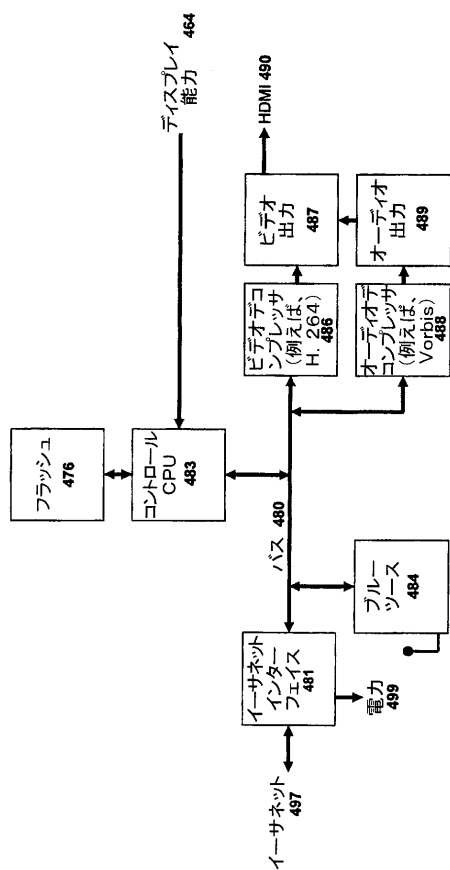
【図 4 c】



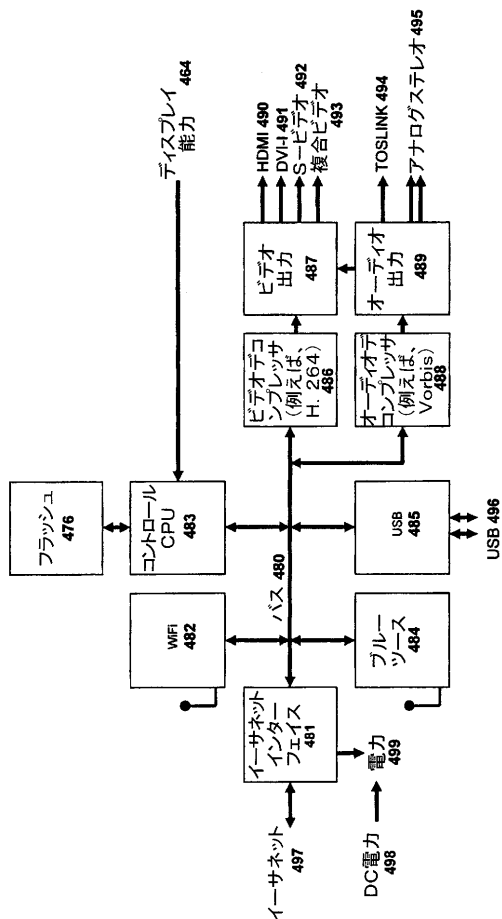
【 図 4 d 】



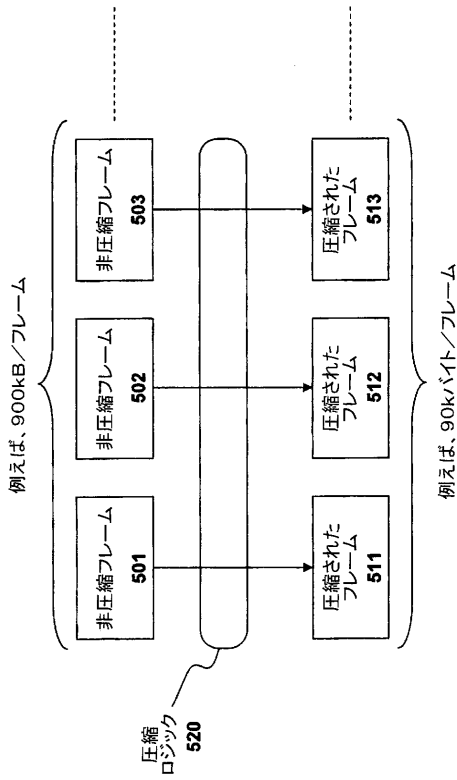
【図 4 e】



【図 4 f】

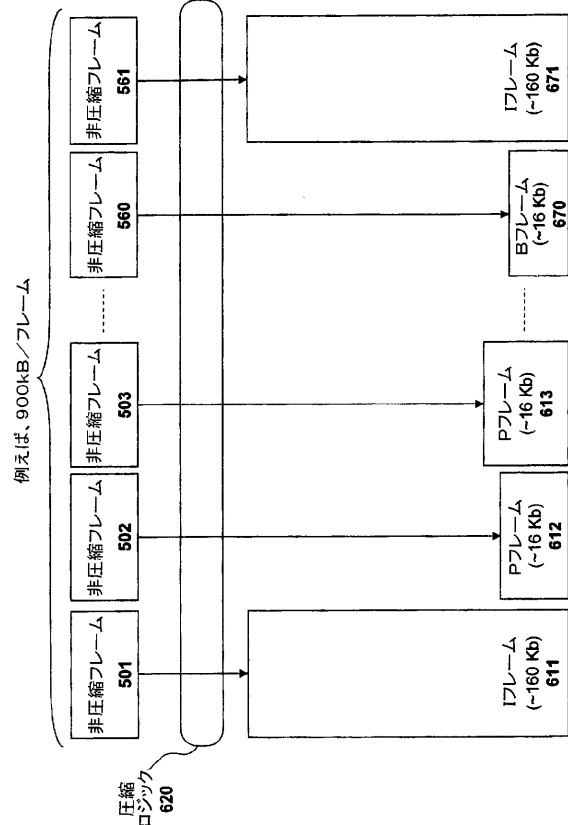


【図 5】



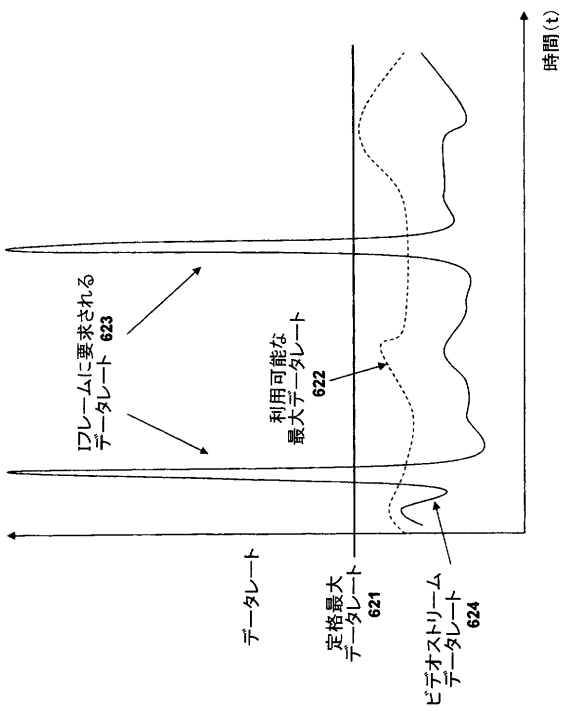
(従来技術)

【図 6 a】

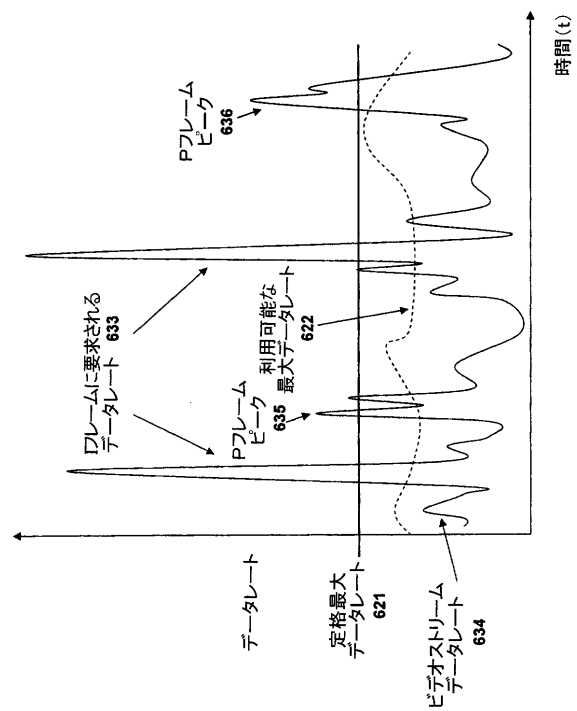


(従来技術)

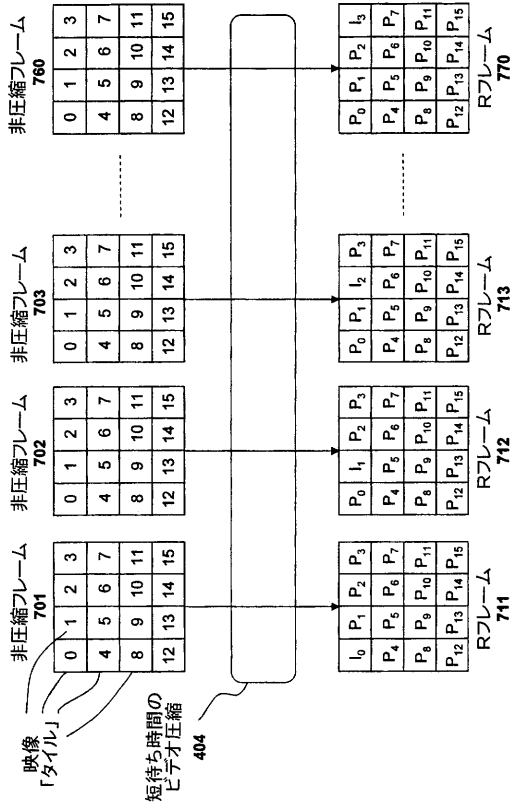
【図 6 b】



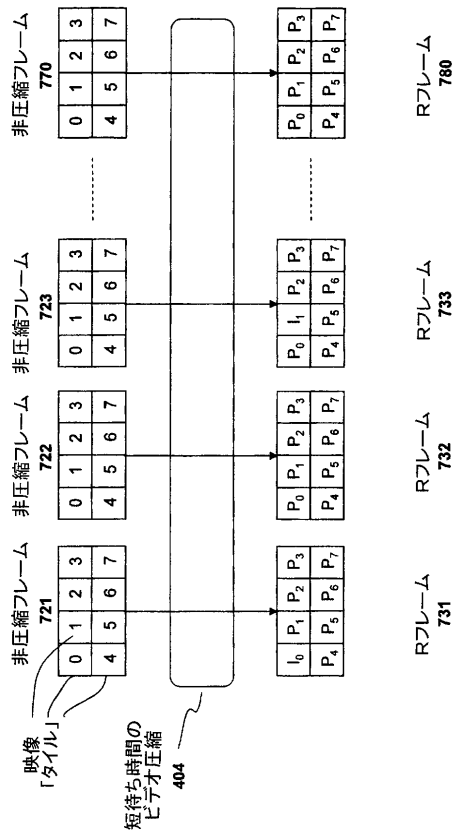
【図 6 c】



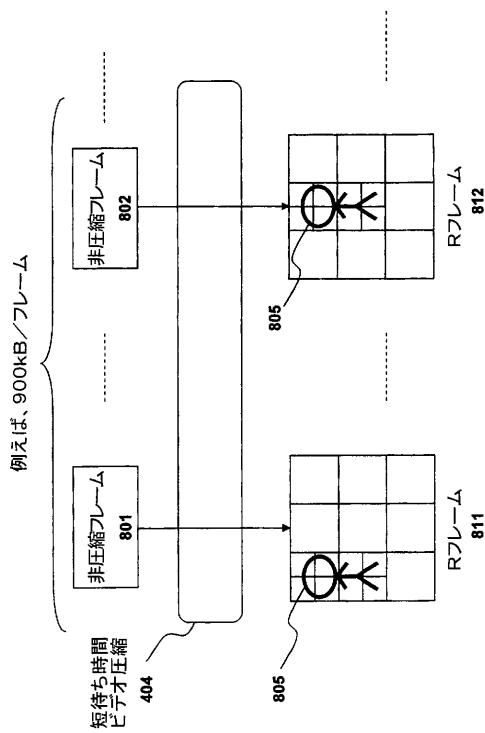
【図 7 a】



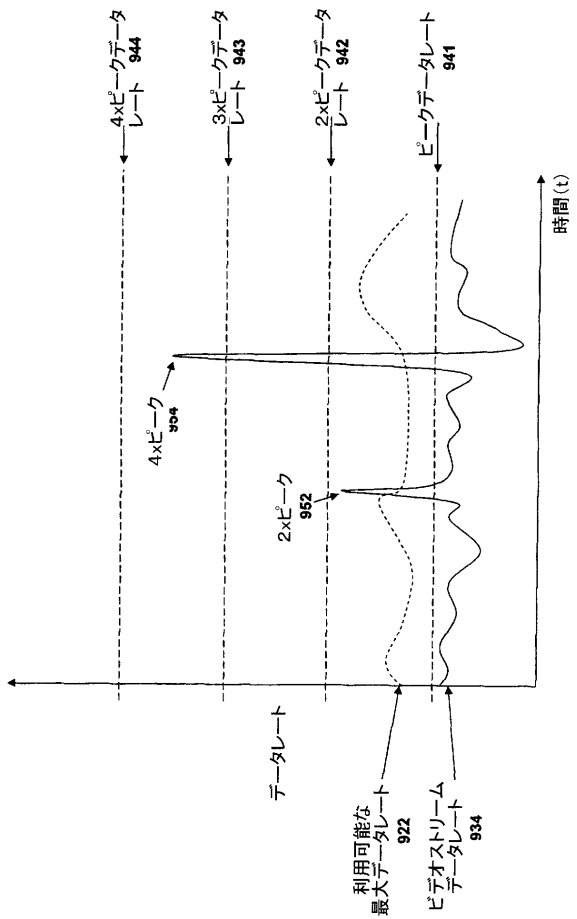
【図 7 b】



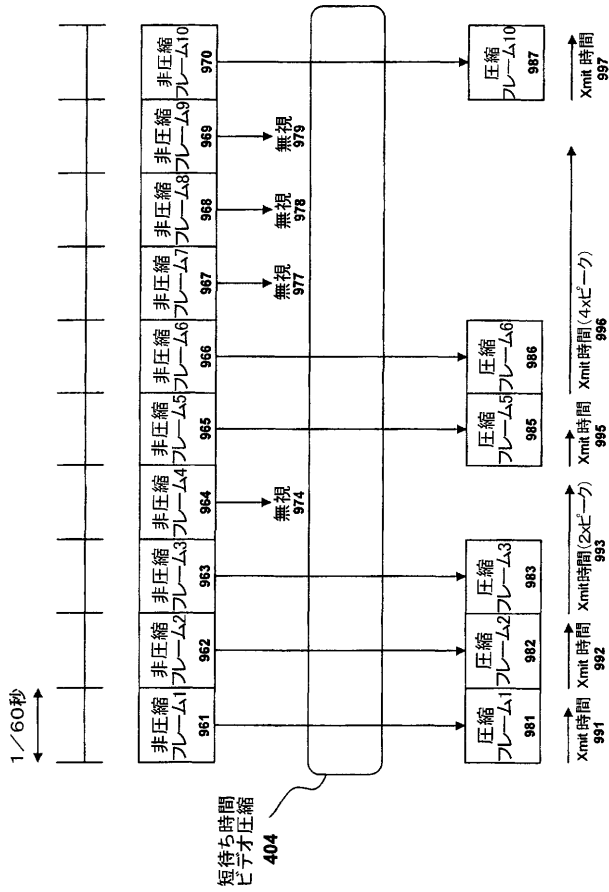
【図 8】



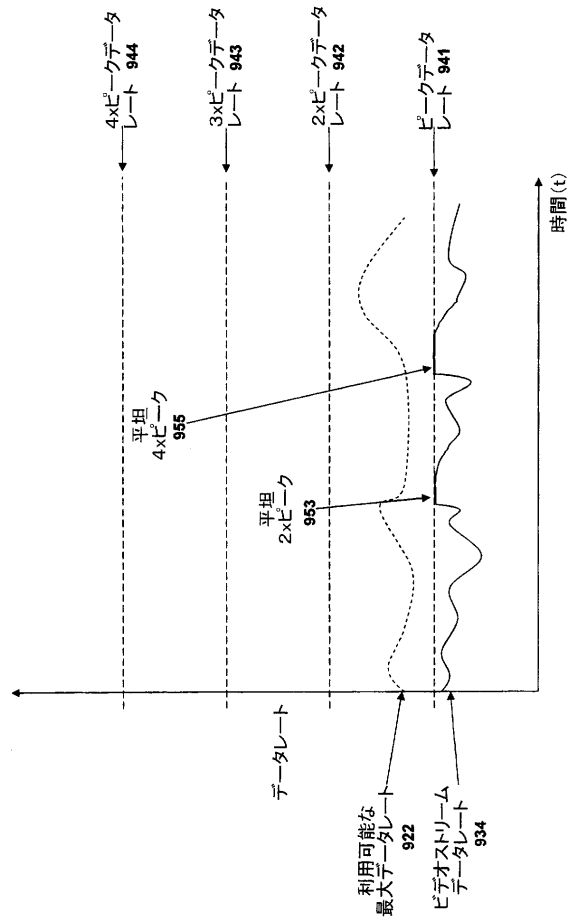
【図 9 a】



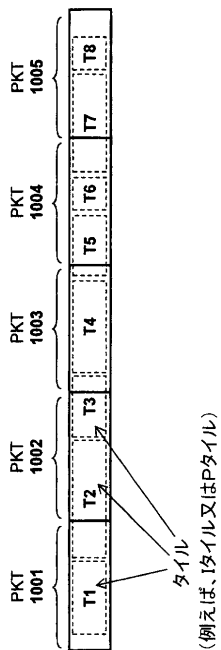
【図 9 b】



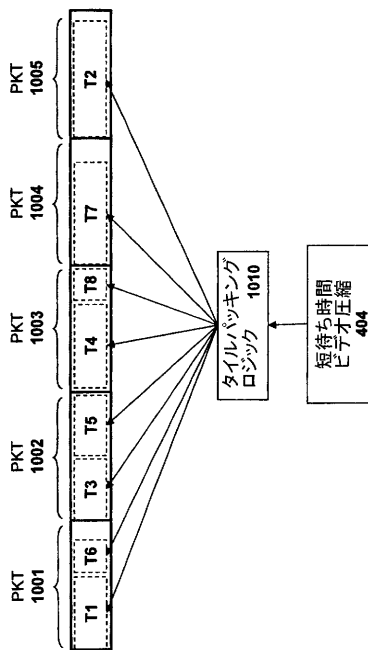
【図 9 c】



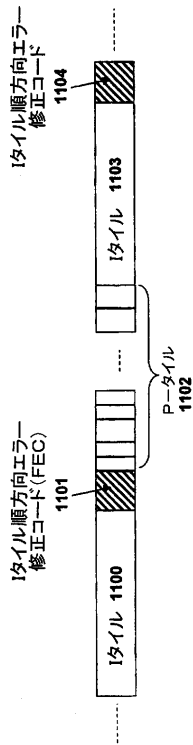
【図 10 a】



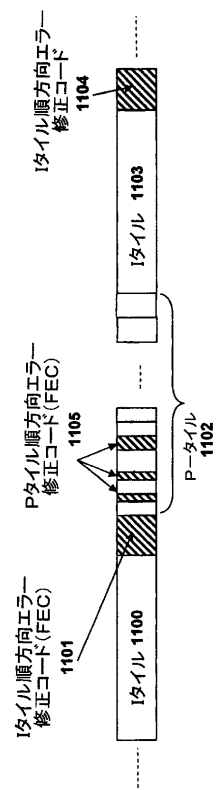
【図 10 b】



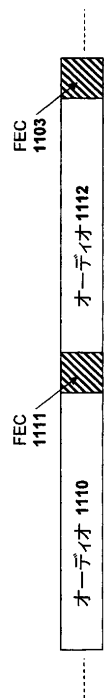
【図 111 a】



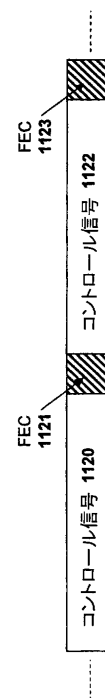
【図 111 b】



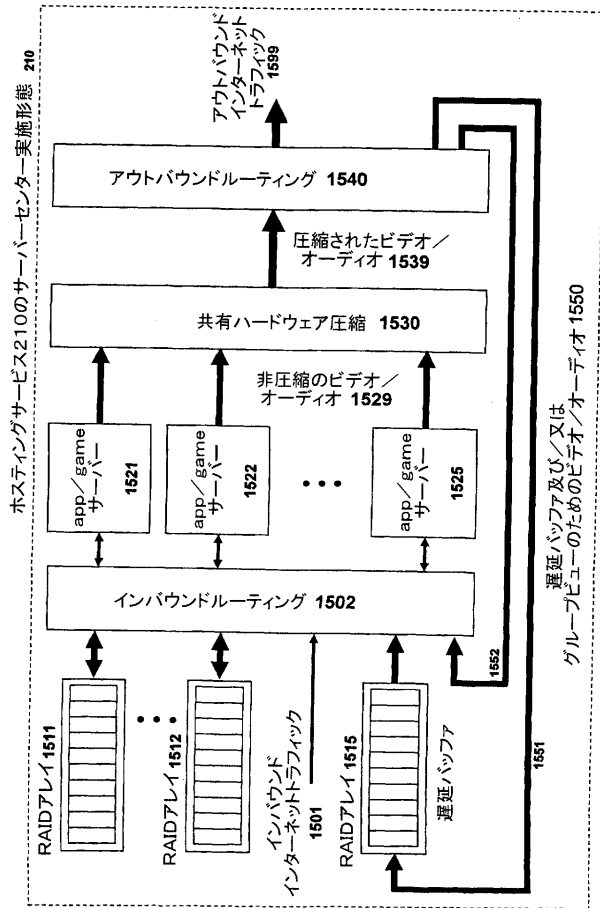
【図 111 c】



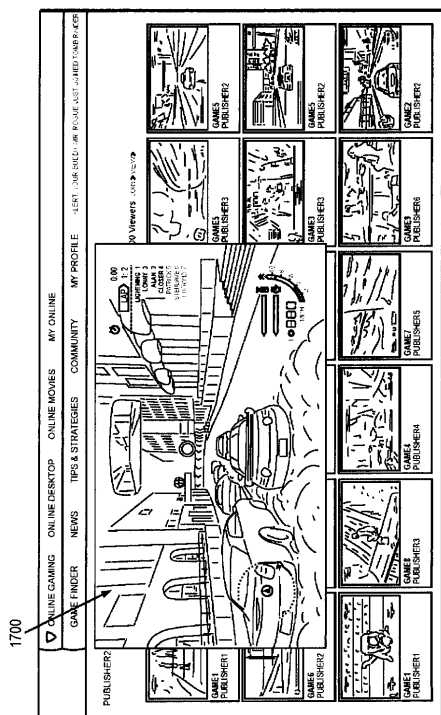
【図 111 d】



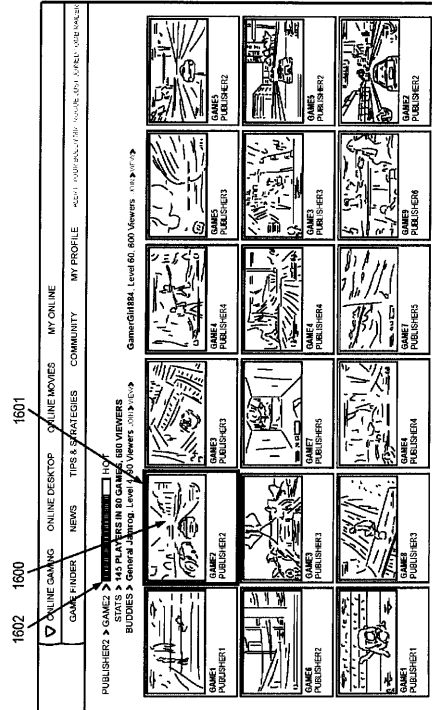
【図 15】



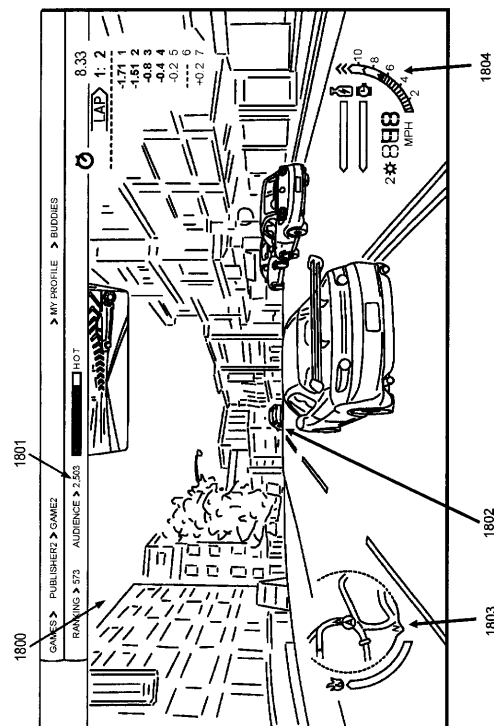
【図 17】



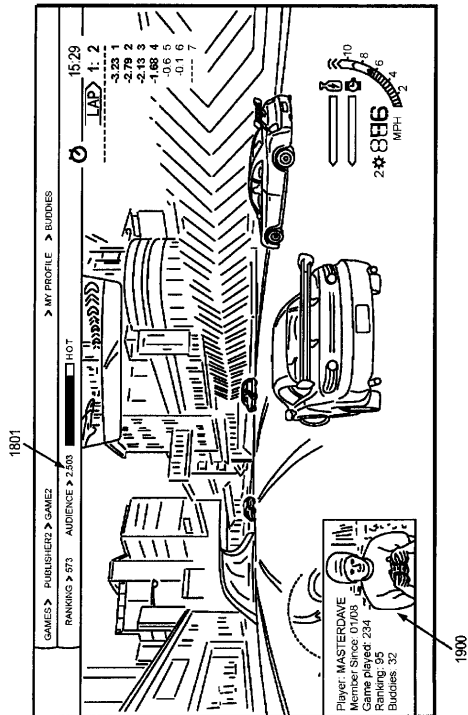
【図 16】



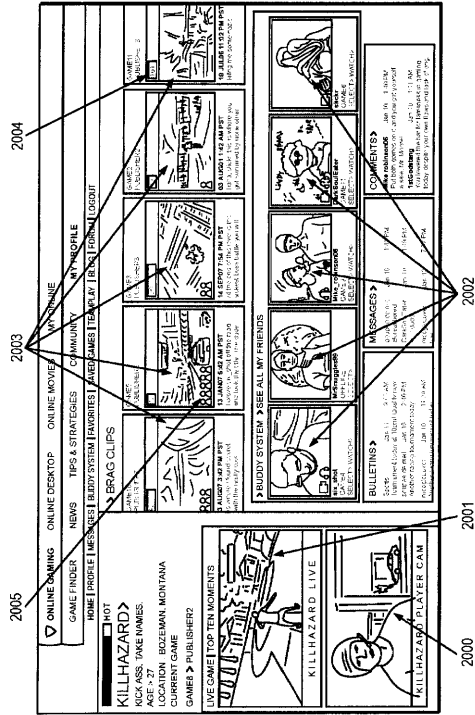
【図 18】



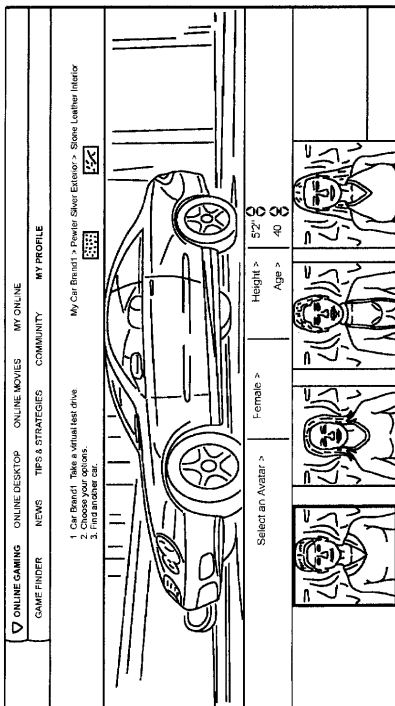
【 19 】



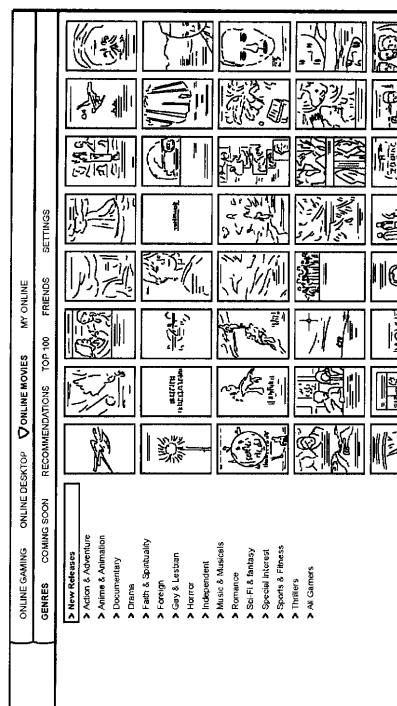
【 20 】



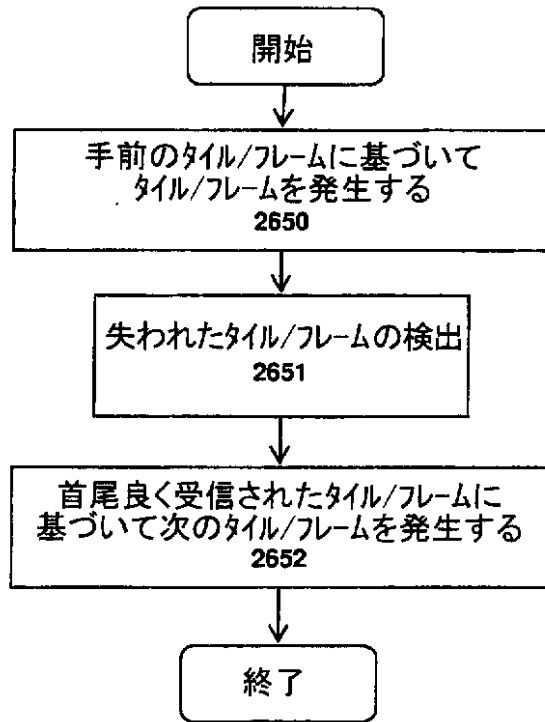
【 21 】



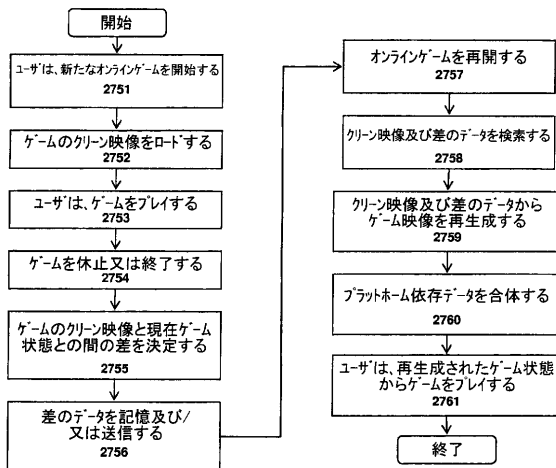
【 23 】



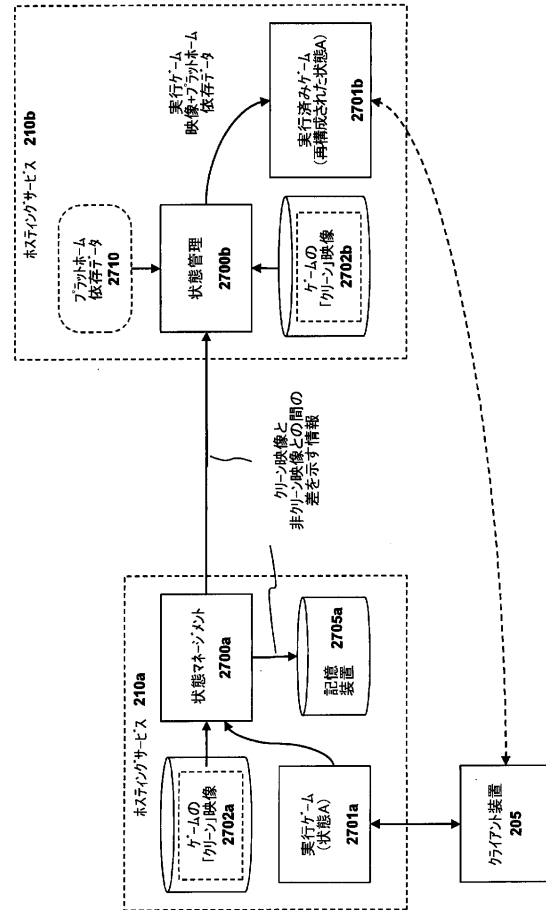
【図 26 b】



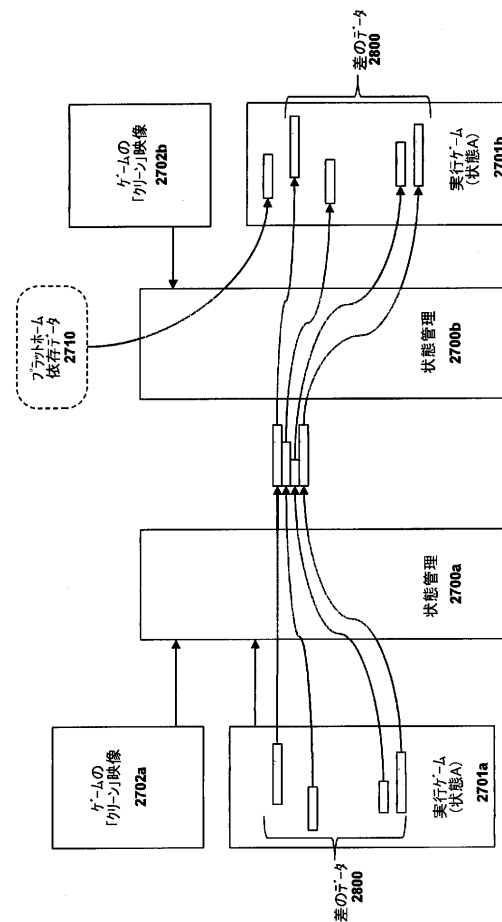
【図 27 b】



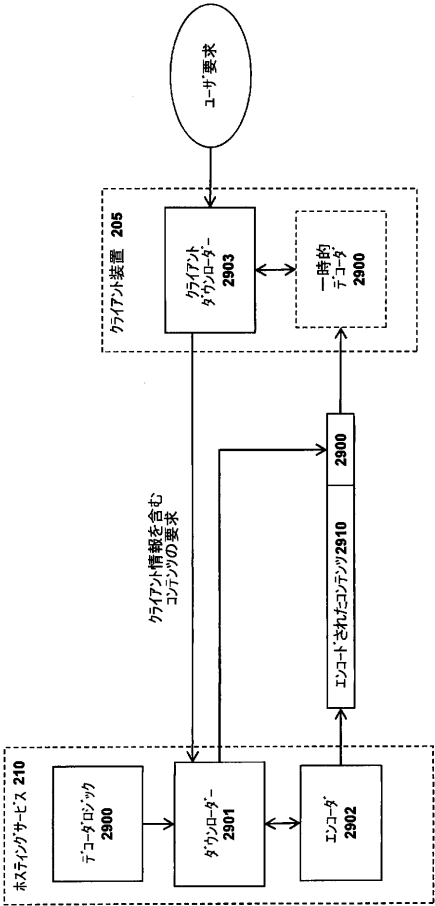
【図 27 a】



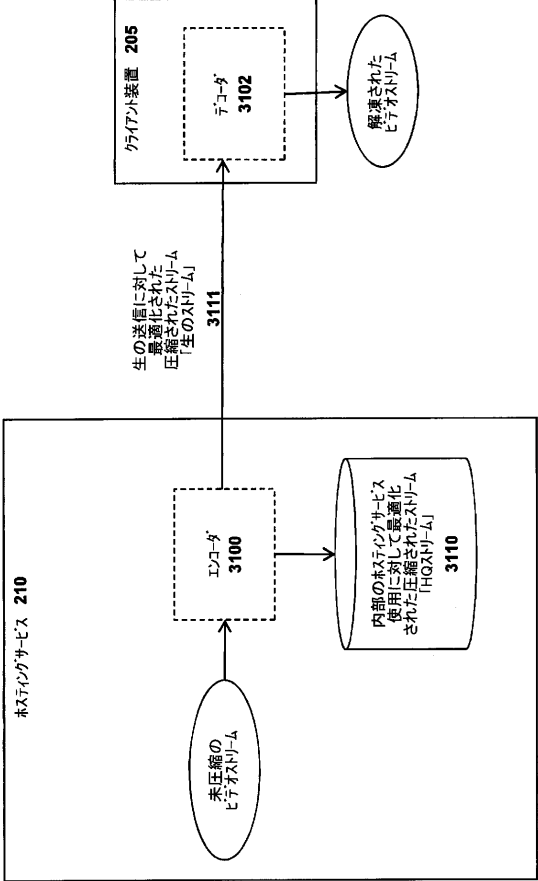
【図 28】



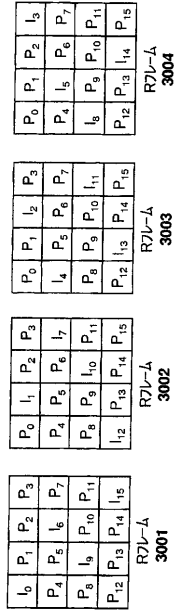
【図 29】



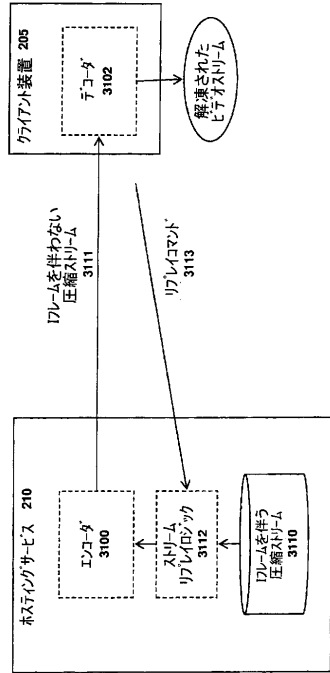
【図 31 a】



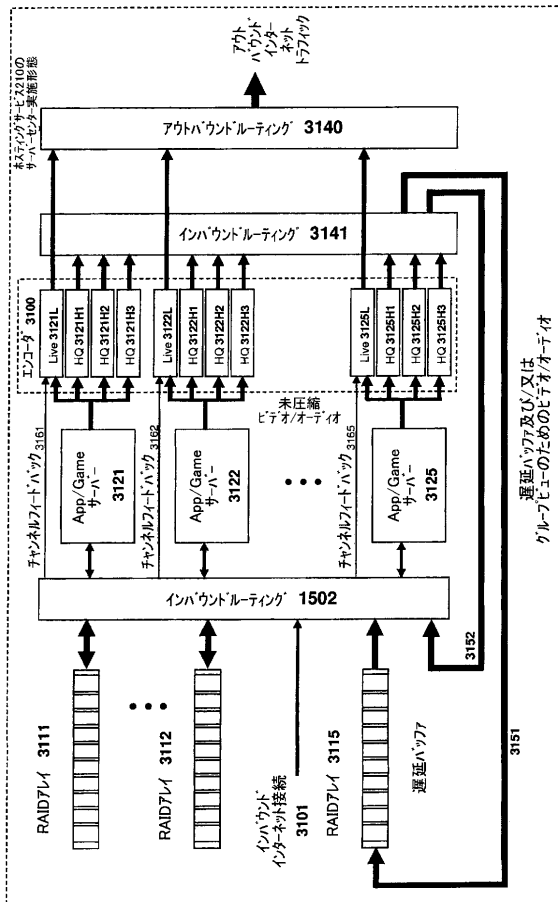
【図 30】



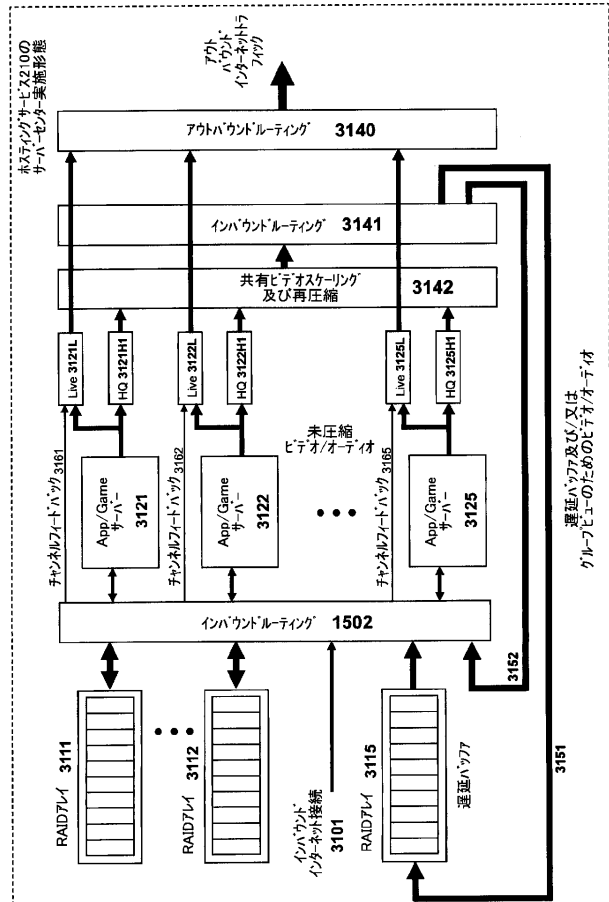
【図 31 b】



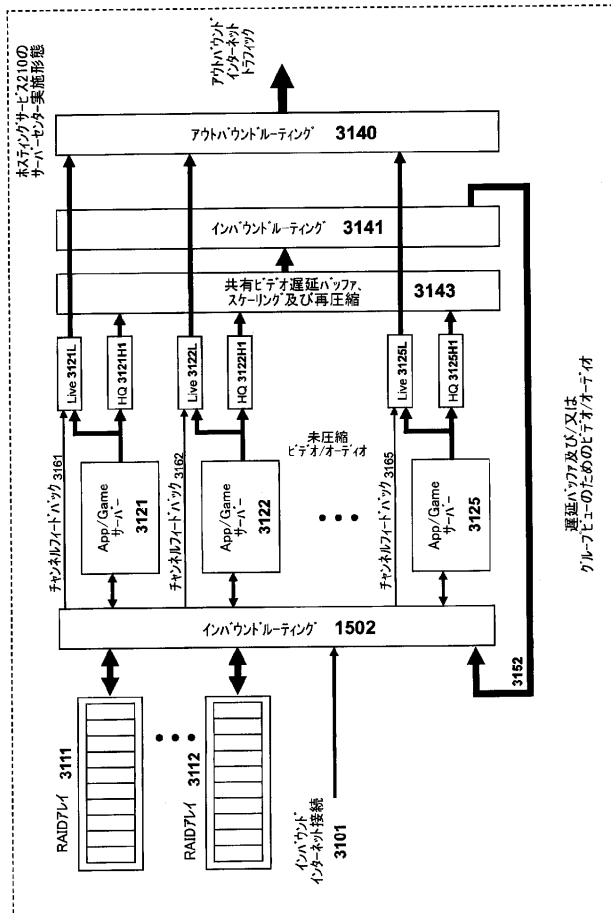
【図 31 c】



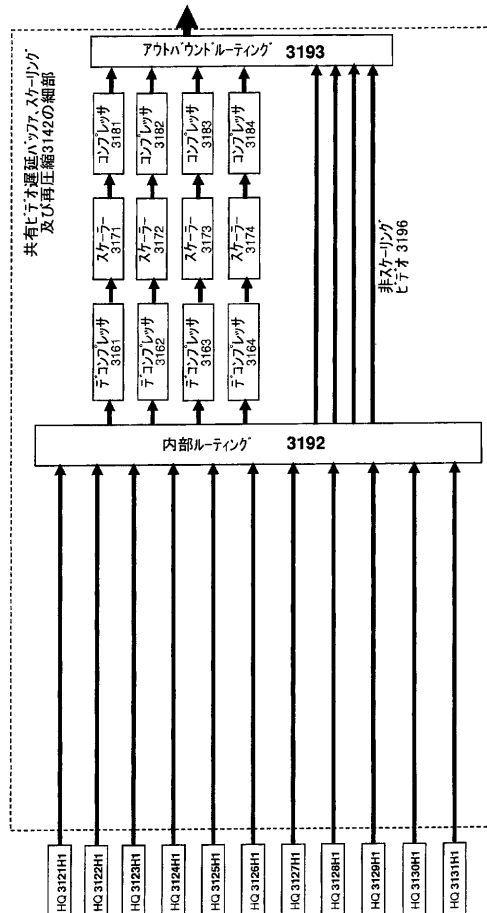
【図 31 d】



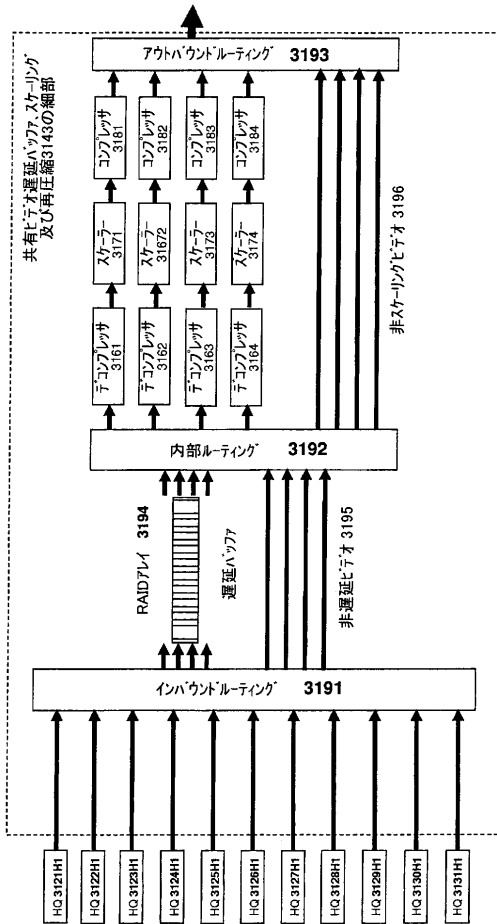
【図 31 e】



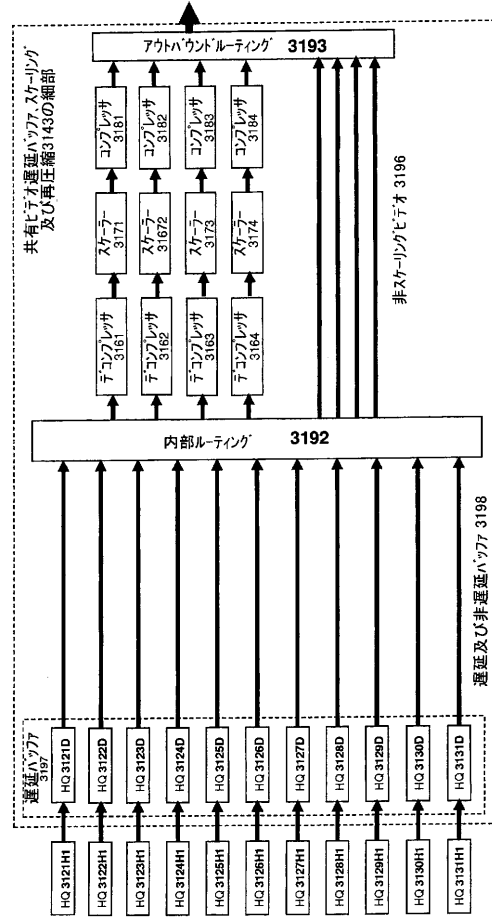
【図 31 f】



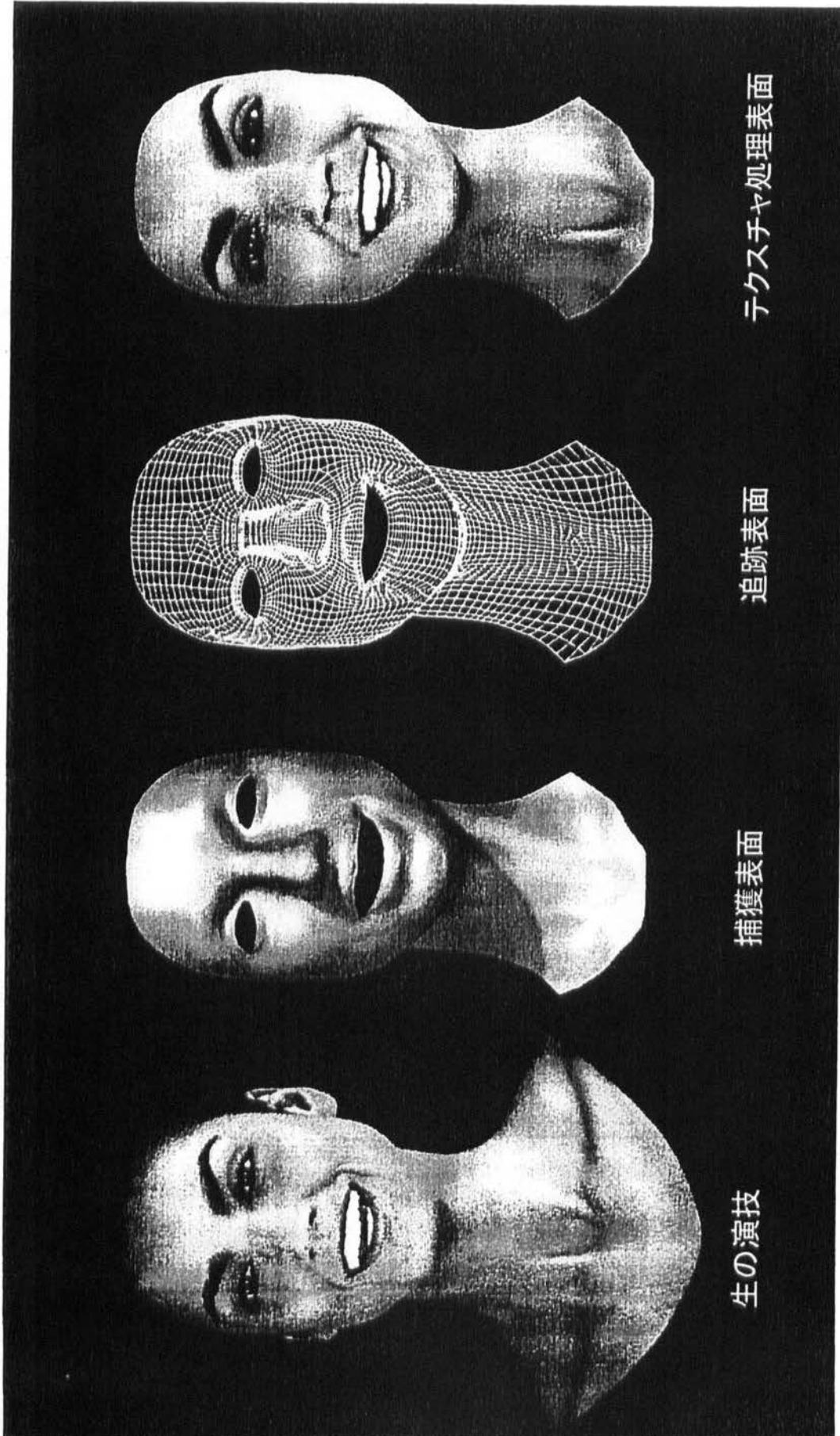
【図 31 g】



【図 31 h】



【図 2 2】



フロントページの続き

(74)代理人 100109335

弁理士 上杉 浩

(74)代理人 100121979

弁理士 岩崎 吉信

(72)発明者 スティーブン ジー パールマン

アメリカ合衆国 カリフォルニア州 パロ アルト

(72)発明者 マイケル トイ

アメリカ合衆国 カリフォルニア州 ロス アルトス

(72)発明者 ティモシー エス コター

アメリカ合衆国 カリフォルニア州 サニーヴェイル

(72)発明者 ジェローム ポワシェ

アメリカ合衆国 カリフォルニア州 クーパーティノ

(72)発明者 ポール アンドリュウ オルブリッチ

アメリカ合衆国 カリフォルニア州 パロ アルト

Fターム(参考) 5C164 FA06 SB08S SB21P SB29P TA08S UA43S UB26P UB41P UD65S YA11

【外国語明細書】

SYSTEM AND METHOD FOR REMOTE-HOSTED VIDEO EFFECTS

RELATED APPLICATION

This application claims the benefit of U.S. Provisional Patent Application No. 61/259,927 filed November 10, 2009. The present application is a continuation-in-part (CIP) application of Serial No. 11/999,461 filed December 05, 2007, which is a CIP application of Serial No. 10/315,460 filed December 10, 2002, each of which is assigned to the assignee of the present CIP application.

TECHNICAL FIELD

The present disclosure relates generally to the field of data processing systems that improve a users' ability to manipulate and access audio and video media.

BACKGROUND

Recorded audio and motion picture media has been an aspect of society since the days of Thomas Edison. At the start of the 20th century there was wide distribution of recorded audio media (cylinders and records) and motion picture media (nickelodeons and movies), but both technologies were still in their infancy. In the late 1920s motion pictures were combined with audio on a mass-market basis, followed by color motion pictures with audio. Radio broadcasting gradually evolved into a largely advertising-supported form of broadcast mass-market audio media. When a television (TV) broadcast standard was established in the mid-1940s, television joined radio as a form of broadcast mass-market media bringing previously recorded or live motion pictures into the home.

By the middle of the 20th century, a large percentage of US homes had phonograph record players for playing recorded audio media, a radio to receive live broadcast audio, and a television set to play live broadcast audio / video (A/V) media. Very often these 3 “media players” (record player, radio and TV) were combined into one cabinet sharing common speakers that became the “media center” for the home. Although the media choices were limited to the consumer, the media “ecosystem” was quite stable. Most consumers knew how to use the “media players” and were able to enjoy the full extent of their capabilities. At the same time, the publishers of the media (largely the motion picture and televisions studios, and the music companies) were able to distribute their media both to theaters and to the home without suffering from widespread piracy or “second sales”, i.e., the resale of used media. Typically publishers do not derive revenue from second sales, and as such, it reduces revenue that publishers might otherwise derive from the buyer of used media for new sales. Although there certainly were used records sold during the middle of the 20th century, such sales did not have a large impact on record publishers because, unlike a motion picture or video program -- which is typically watched once or only a few times by an adult -- a music track may be listened to hundreds or even thousands of times. So, music media is far less “perishable” (i.e., it has lasting value to an adult consumer) than motion picture/video media. Once a record was purchased, if the consumer liked the music, the consumer was likely to keep it a long time.

From the middle of the 20th century through the present day, the media ecosystem has undergone a series of radical changes, both to the benefit and the detriment of consumers and publishers. With the widespread introduction of audio recorders, especially cassette tapes with high-quality stereo sound, there certainly was a higher degree of consumer convenience. But it also marked the

beginning of what is now a widespread practice with consumer media: piracy. Certainly, many consumers used the cassette tapes for taping their own records purely for convenience, but increasingly consumers (e.g., students in a dormitory with ready access to each others' record collections) would make pirated copies. Also, consumers would tape music played over the radio rather than buying a record or tape from the publisher.

The advent of the consumer VCR led to even more consumer convenience, since now a VCR could be set to record a TV show which could be watched at a later time, and it also led to the creation of the video rental business, where movies as well as TV programming could be accessed on an "on demand" basis. The rapid development of mass-market home media devices since the mid-1980s has led to an unprecedented level of choice and convenience for the consumer, and also has led to a rapid expansion of the media publishing market.

Today, consumers are faced with a plethora of media choices as well as a plethora of media devices, many of which are tied to particular forms of media or particular publishers. An avid consumer of media may have a stack of devices connected to TVs and computers in various rooms of the house, resulting in a "rat's nest" of cables to one or more TV sets and/or personal computers (PCs) as well as a group of remote controls. (In the context of the present application, the term "personal computer" or "PC" refers to any sort of computer suitable for use in the home or office, including a desktop, a Macintosh® or other non-Windows computers, Windows-compatible devices, Unix variations, laptops, etc.) These devices may include a video game console, VCR, DVD player, audio surround-sound processor/amplifier, satellite set-top box, cable TV set-top box, etc. And, for an avid consumer, there may be multiple similar-function devices because of compatibility issues. For example, a consumer may own both a HD-DVD and a Blu-ray DVD

player, or both a Microsoft Xbox® and a Sony Playstation® video game system. Indeed, because of incompatibility of some games across versions of game consoles, the consumer may own both an XBox and a later version, such as an Xbox 360®. Frequently, consumers are befuddled as to which video input and which remote to use. Even after a disc is placed into the correct player (e.g., DVD, HD-DVD, Blu-ray, Xbox or Playstation), the video and audio input is selected for that the device, and the correct remote control is found, the consumer is still faced with technical challenges. For example, in the case of a wide-screen DVD, the user may need to first determine and then set the correct aspect ratio on his TV or monitor screen (e.g., 4:3, Full, Zoom, Wide Zoom, Cinema Wide, etc.). Similarly, the user may need to first determine and then set the correct audio surround sound system format (e.g., AC-3, Dolby Digital, DTS, etc.). Often times, the consumer is unaware that they may not be enjoying the media content to the full capability of their television or audio system (e.g., watching a movie squashed at the wrong aspect ratio, or listening to audio in stereo rather than in surround sound).

Increasingly, Internet-based media devices have been added to the stack of devices. Audio devices like the Sonos® Digital Music system stream audio directly from the Internet. Likewise, devices like the Slingbox™ entertainment player record video and stream it through a home network or out through the Internet where it can be watched remotely on a PC. And Internet Protocol Television (IPTV) services offer cable TV-like services through Digital Subscriber Line (DSL) or other home Internet connections. There have also been recent efforts to integrate multiple media functions into a single device, such as the Moxi® Media Center and PCs running Windows XP Media Center Edition. While each of these devices offers an element of convenience for the functions that it performs, each lacks ubiquitous and simple access to most media. Further, such devices frequently cost hundreds of

dollars to manufacture, often because of the need for expensive processing and/or local storage. Additionally, these modern consumer electronic devices typically consume a great deal of power, even while idle, which means they are expensive over time and wasteful of energy resources. For example, a device may continue to operate if the consumer neglects to turn it off or switches to a different video input. And, because none of the devices is a complete solution, it must be integrated with the other stack of devices in the home, which still leaves the user with a rat's nest of wires and a sea of remote controls.

Furthermore, when many newer Internet-based devices do work properly, they typically offer media in a more generic form than it might otherwise be available. For example, devices that stream video through the Internet often stream just the video material, not the interactive "extras" that often accompany DVDs, like the "making of" videos, games, or director's commentary. This is due to the fact that frequently the interactive material is produced in a particular format intended for a particular device that handles interactivity locally. For example, each of DVD, HD-DVDs and Blu-ray discs have their own particular interactive format. Any home media device or local computer that might be developed to support all of the popular formats would require a level of sophistication and flexibility that would likely make it prohibitively expensive and complex for the consumer to operate.

Adding to the problem, if a new format were introduced later in the future the local device may not have the hardware capability to support the new format, which would mean that the consumer would have to purchase an upgraded local media device. For example, if higher-resolution video or stereoscopic video (e.g., one video stream for each eye) were introduced at a later date, the local device may not have the computational capability to decode the video, or it may not have the hardware to output the video in the new format (e.g., assuming

stereoscopy is achieved through 120fps video synchronized with shuttered glasses, with 60fps delivered to each eye, if the consumer's video hardware can only support 60fps video, this option would be unavailable absent an upgraded hardware purchase).

The issue of media device obsolescence and complexity is a serious problem when it comes to sophisticated interactive media, especially video games.

Modern video game applications are largely divided into four major non-portable hardware platforms: Sony PlayStation® 1, 2 and 3 (PS1, PS2, and PS3); Microsoft Xbox® and Xbox 360®; and Nintendo Gamecube® and Wii™; and PC-based games. Each of these platforms is different than the others so that games written to run on one platform usually do not run on another platform. There may also be compatibility problems from one generation of device to the next. Even though the majority of software game developers create software games that are designed independent of a particular platform, in order to run a particular game on a specific platform a proprietary layer of software (frequently called a "game development engine") is needed to adapt the game for use on a specific platform. Each platform is sold to the consumer as a "console" (i.e., a standalone box attached to a TV or monitor/speakers) or it is a PC itself. Typically, the video games are sold on optical media such as a Blu-ray DVD, DVD-ROM or CD-ROM, which contains the video game embodied as a sophisticated real-time software application. As home broadband speeds have increased, video games are becoming increasingly available for download.

The specificity requirements to achieve platform-compatibility with video game software is extremely exacting due to the real-time nature and high computational requirements of advanced video games. For example, one might expect full game compatibility from one generation to the next of video games (e.g.,

from XBox to XBox 360, or from Playstation 2 ("PS2") to Playstation 3 ("PS3"), just as there is general compatibility of productivity applications (e.g., Microsoft Word) from one PC to another with a faster processing unit or core. However, this is not the case with video games. Because the video game manufacturers typically are seeking the highest possible performance for a given price point when a video game generation is released, dramatic architectural changes to the system are frequently made such that many games written for the prior generation system do not work on the later generation system. For example, XBox was based upon the x86-family of processors, whereas XBox 360 was based upon a PowerPC-family.

Techniques can be utilized to emulate a prior architecture, but given that video games are real-time applications, it is often unfeasible to achieve the exact same behavior in an emulation. This is a detriment to the consumer, the video game console manufacturer and the video game software publisher. For the consumer, it means the necessity of keeping both an old and new generation of video game consoles hooked up to the TV to be able to play all games. For the console manufacturer it means cost associated with emulation and slower adoption of new consoles. And for the publisher it means that multiple versions of new games may have to be released in order to reach all potential consumers -- not only releasing a version for each brand of video game (e.g., XBox, Playstation), but often a version for each version of a given brand (e.g., PS2 and PS3). For example, a separate version of Electronic Arts' "Madden NFL 08" was developed for XBox, XBox 360, PS2, PS3, Gamecube, Wii, and PC, among other platforms.

Portable devices, such as cellular ("cell") phones and portable media players also present challenges to game developers. Increasingly such devices are connected to wireless data networks and are able to download video games. But, there are a wide variety of cell phones and media devices in the market, with a wide

range of different display resolutions and computing capabilities. Also, because such devices typically have power consumption, cost and weight constraints, they typically lack advanced graphics acceleration hardware like a Graphics Processing Unit ("GPU"), such as devices made by NVIDIA of Santa Clara, CA. Consequently, game software developers typically develop a given game title simultaneously for many different types of portable devices. A user may find that a given game title is not available for his particular cell phone or portable media player.

In the case of home game consoles, hardware platform manufacturers typically charge a royalty to the software game developers for the ability to publish a game on their platform. Cell phone wireless carriers also typically charge a royalty to the game publisher to download a game into the cell phone. In the case of PC games, there is no royalty paid to publish games, but game developers typically face high costs due to the higher customer service burden to support the wide range of PC configurations and installation issues that may arise. Also, PCs typically present less barriers to the piracy of game software since they are readily reprogrammable by a technically-knowledgeable user and games can be more easily pirated and more easily distributed (e.g., through the Internet). Thus, for a software game developer, there are costs and disadvantages in publishing on game consoles, cell phones and PCs.

For game publishers of console and PC software, costs do not end there. To distribute games through retail channels, publishers charge a wholesale price below the selling price for the retailer to have a profit margin. The publisher also typically has to pay the cost of manufacturing and distributing the physical media holding the game. The publisher is also frequently charged a "price protection fee" by the retailer to cover possible contingencies such as where the game does not sell, or if the game's price is reduced, or if the retailer must refund part or all of

the wholesale price and/or take the game back from a buyer. Additionally, retailers also typically charge fees to publishers to help market the games in advertising flyers. Furthermore, retailers are increasingly buying back games from users who have finished playing them, and then sell them as used games, typically sharing none of the used game revenue with the game publisher. Adding to the cost burden placed upon game publishers is the fact that games are often pirated and distributed through the Internet for users to download and make free copies.

As Internet broadband speeds have been increasing and broadband connectivity has become more widespread in the US and worldwide, particularly to the home and to Internet "cafes" where Internet-connected PCs are rented, games are increasingly being distributed via downloads to PCs or consoles. Also, broadband connections are increasingly used for playing multiplayer and massively multiplayer online games (both of which are referred to in the present disclosure by the acronym "MMOG"). These changes mitigate some of the costs and issues associated with retail distribution. Downloading online games addresses some of the disadvantages to game publishers in that distribution costs typically are less and there are little or no costs from unsold media. But downloaded games are still subject to piracy, and because of their size (often many gigabytes in size) they can take a very long time to download. In addition, multiple games can fill up small disk drives, such as those sold with portable computers or with video game consoles. However, to the extent games or MMOGs require an online connection for the game to be playable, the piracy problem is mitigated since the user is usually required to have a valid user account. Unlike linear media (e.g., video and music) which can be copied by a camera shooting video of the display screen or a microphone recording audio from the speakers, each video game experience is unique, and can not be copied using simple video/audio recording. Thus, even in regions where copyright

laws are not strongly enforced and piracy is rampant, MMOGs can be shielded from piracy and therefore a business can be supported. For example, Vivendi SA's "World of Warcraft" MMOG has been successfully deployed without suffering from piracy throughout the world. And many online or MMOG games, such as Linden Lab's "Second Life" MMOG generate revenue for the games' operators through economic models built into the games where assets can be bought, sold, and even created using online tools. Thus, mechanisms in addition to conventional game software purchases or subscriptions can be used to pay for the use of online games.

While piracy can be often mitigated due to the nature of online or MMOGs, online game operator still face remaining challenges. Many games require substantial local (i.e., in-home) processing resources for online or MMOGs to work properly. If a user has a low performance local computer (e.g., one without a GPU, such as a low-end laptop), he may not be able to play the game. Additionally, as game consoles age, they fall further behind the state-of-the-art and may not be able to handle more advanced games. Even assuming the user's local PC is able to handle the computational requirements of a game, there are often installation complexities. There may be driver incompatibilities (e.g., if a new game is downloaded, it may install a new version of a graphics driver that renders a previously-installed game, reliant upon an old version of the graphics driver, inoperable). A console may run out of local disk space as more games are downloaded. Complex games typically receive downloaded patches over time from the game developer as bugs are found and fixed, or if modifications are made to the game (e.g., if the game developer finds that a level of the game is too hard or too easy to play). These patches require new downloads. But sometimes not all users complete downloading of all the patches. Other times, the downloaded patches introduce other compatibility or disk space consumption issues.

Also, during game play, large data downloads may be required to provide graphics or behavioral information to the local PC or console. For example, if the user enters a room in a MMOG and encounters a scene or a character made up of graphics data or with behaviors that are not available on the user's local machine, then that scene or character's data must be downloaded. This may result in a substantial delay during game play if the Internet connection is not fast enough. And, if the encountered scene or character requires storage space or computational capability beyond that of the local PC or console, it can create a situation where the user can not proceed in the game, or must continue with reduced-quality graphics. Thus, online or MMOG games often limit their storage and/or computational complexity requirements. Additionally, they often limit the amount of data transfers during the game. Online or MMOG games may also narrow the market of users that can play the games.

Furthermore, technically-knowledgeable users are increasingly reverse-engineering local copies of games and modifying the games so that they can cheat. The cheats maybe as simple as making a button press repeat faster than is humanly possible (e.g., so as to shoot a gun very rapidly). In games that support in-game asset transactions the cheating can reach a level of sophistication that results in fraudulent transactions involving assets of actual economic value. When an online or MMOGs economic model is based on such asset transactions, this can result in substantial detrimental consequences to the game operators.

The cost of developing a new game has grown as PCs and consoles are able to produce increasingly sophisticated games (e.g., with more realistic graphics, such as real-time ray-tracing, and more realistic behaviors, such as real-time physics simulation). In the early days of the video game industry, video game development was a very similar process to application software development; that is,

most of the development cost was in the development of the software, as opposed to the development of the graphical, audio, and behavioral elements or “assets”, such as those that may be developed for a motion picture with extensive special effects. Today, many sophisticated video game development efforts more closely resemble special effects-rich motion picture development than software development. For instance, many video games provide simulations of 3-D worlds, and generate increasingly photorealistic (i.e., computer graphics that seem as realistic as live action imagery shot photographically) characters, props, and environments. One of the most challenging aspects of photorealistic game development is creating a computer-generated human face that is indistinguishable from a live action human face. Facial capture technologies such as Contour™ Reality Capture developed by Mova of San Francisco, CA captures and tracks the precise geometry of a performer’s face at high resolution while it is in motion. This technology allows a 3D face to be rendered on a PC or game console that is virtually indistinguishable from a captured live action face. Capturing and rendering a “photoreal” human face precisely is useful in several respects. First, highly recognizable celebrities or athletes are often used in video games (often hired at a high cost), and imperfections may be apparent to the user, making the viewing experience distracting or unpleasant. Frequently, a high degree of detail is required to achieve a high degree of photorealism -- requiring the rendering of a large number of polygons and high-resolution textures, potentially with the polygons and/or textures changing on a frame-by-frame basis as the face moves.

When high polygon-count scenes with detailed textures change rapidly, the PC or game console supporting the game may not have sufficient RAM to store enough polygon and texture data for the required number of animation frames generated in the game segment. Further, the single optical drive or single disk drive

typically available on a PC or game console is usually much slower than the RAM, and typically can not keep up with the maximum data rate that the GPU can accept in rendering polygons and textures. Current games typically load most of the polygons and textures into RAM, which means that a given scene is largely limited in complexity and duration by the capacity of the RAM. In the case of facial animation, for example, this may limit a PC or a game console to either a low resolution face that is not photoreal, or to a photoreal face that can only be animated for a limited number of frames, before the game pauses, and loads polygons and textures (and other data) for more frames.

Watching a progress bar move slowly across the screen as a PC or console displays a message similar to "Loading..." is accepted as an inherent drawback by today's users of complex video games. The delay while the next scene loads from the disk ("disk" herein, unless otherwise qualified, refers to non-volatile optical or magnetic media, as well non-disk media such as semiconductor "Flash" memory) can take several seconds or even several minutes. This is a waste of time and can be quite frustrating to a game player. As previously discussed, much or all of the delay may be due to the load time for polygon, textures or other data from a disk, but it also may be the case that part of the load time is spent while the processor and/or GPU in the PC or console prepares data for the scene. For example, a soccer video game may allow the players to choose among a large number of players, teams, stadiums and weather conditions. So, depending on what particular combination is chosen, different polygons, textures and other data (collectively "objects") may be required for the scene (e.g., different teams have different colors and patterns on their uniforms). It may be possible to enumerate many or all of the various permutations and pre-compute many or all of the objects in advance and store the objects on the disk used to store the game. But, if the

number of permutations is large, the amount of storage required for all of the objects may be too large to fit on the disk (or too impractical to download). Thus, existing PC and console systems are typically constrained in both the complexity and play duration of given scenes and suffer from long load times for complex scenes.

Another significant limitation with prior art video game systems and application software systems is that they are increasingly using large databases, e.g., of 3D objects such as polygons and textures, that need to be loaded into the PC or game console for processing. As discussed above, such databases can take a long time to load when stored locally on a disk. Load time, however, is usually far more severe if the database is stored a remote location and is accessed through the Internet. In such a situation it may take minutes, hours, or even days to download a large database. Further, such databases are often created a great expense (e.g., a 3D model of a detailed tall-masted sailing ship for use in a game, movie, or historical documentary) and are intended for sale to the local end-user. However, the database is at risk of being pirated once it has been downloaded to the local user. In many cases, a user wants to download a database simply for the sake of evaluating it to see if it suits the user's needs (e.g., if a 3D costume for a game character has a satisfactory appearance or look when the user performs a particular move). A long load time can be a deterrent for the user evaluating the 3D database before deciding to make a purchase.

Similar issues occur in MMOGs, particularly as games that allow users to utilize increasingly customized characters. For a PC or game console to display a character it needs to have access to the database of 3D geometry (polygons, textures, etc.) as well as behaviors (e.g., if the character has a shield, whether the shield is strong enough to deflect a spear or not) for that character. Typically, when a MMOG is first played by a user, a large number of databases for characters are

already available with the initial copy of the game, which is available locally on the game's optical disk or downloaded to a disk. But, as the game progresses, if the user encounters a character or object whose database is not available locally (e.g., if another user has created a customized character), before that character or object can be displayed, its database must be downloaded. This can result in a substantial delay of the game.

Given the sophistication and complexity of video games, another challenge for video game developers and publishers with prior art video game consoles, is that it frequently takes 2 to 3 years to develop a video game at a cost of tens of millions of dollars. Given that new video game console platforms are introduced at a rate of roughly once every five years, game developers need to start development work on those games years in advance of the release of the new game console in order to have video games available concurrently when the new platform is released. Several consoles from competing manufactures are sometimes released around the same time (e.g., within a year or two of each other), but what remains to be seen is the popularity of each console, e.g., which console will produce the largest video game software sales. For example, in a recent console cycle, the Microsoft XBox 360, the Sony Playstation 3, and the Nintendo Wii were scheduled to be introduced around the same general timeframe. But years before the introductions the game developers essentially had to "place their bets" on which console platforms would be more successful than others, and devote their development resources accordingly. Motion picture production companies also have to apportion their limited production resources based on what they estimate to be the likely success of a movie well in advance of the release of the movie. Given the growing level of investment required for video games, game production is increasingly becoming like motion picture production, and game production

companies routinely devote their production resources based on their estimate of the future success of a particular video game. But, unlike the motion picture companies, this bet is not simply based on the success of the production itself; rather, it is predicated on the success of the game console the game is intended to run on. Releasing the game on multiple consoles at once may mitigate the risk, but this additional effort increases cost, and frequently delays the actual release of the game.

Application software and user environments on PCs are becoming more computationally intensive, dynamic and interactive, not only to make them more visually appealing to users, but also to make them more useful and intuitive. For example, both the new Windows Vista™ operating system and successive versions of the Macintosh® operating system incorporate visual animation effects. Advanced graphics tools such as Maya™ from Autodesk, Inc., provide very sophisticated 3D rendering and animation capability which push the limits of state-of-the-art CPUs and GPUs. However, the computational requirements of these new tools create a number of practical issues for users and software developers of such products.

Since the visual display of an operating system (OS) must work on a wide range of classes of computers -- including prior-generation computers no longer sold, but still upgradeable with the new OS -- the OS graphical requirements are limited to a large degree by a least common denominator of computers that the OS is targeted for, which typically includes computers that do not include a GPU. This severely limits the graphics capability of the OS. Furthermore, battery-powered portable computers (e.g., laptops) limit the visual display capability since high computational activity in a CPU or GPU typically results in higher power consumption and shorter battery life. Portable computers typically include software that automatically lowers processor activity to reduce power consumption when the

processor is not utilized. In some computer models the user may lower processor activity manually. For example, Sony's VGN-SZ280P laptop contains a switch labeled "Stamina" on one side (for low performance, more battery life) and "Speed" on the other (for high performance, less battery life). An OS running on a portable computer must be able to function useably even in the event the computer is running at a fraction of its peak performance capability. Thus, OS graphics performance often remains far below the state-of-the-art available computational capability.

High-end computationally-intense applications like Maya are frequently sold with the expectation that they will be used on high-performance PCs. This typically establishes a much higher performance, and more expensive and less portable, least common denominator requirement. As a consequence, such applications have a much more limited target audience than a general purpose OS (or general purpose productivity application, like Microsoft Office) and typically sell in much lower volume than general purpose OS software or general purpose application software. The potential audience is further limited because often times it is difficult for a prospective user to try out such computationally-intense applications in advance. For example, suppose a student wants to learn how to use Maya or a potential buyer already knowledgeable about such applications wants to try out Maya before making the investment in the purchase (which may well involve also buying a high-end computer capable of running Maya). While either the student or the potential buyer could download, or get a physical media copy of, a demo version of Maya, if they lack a computer capable of running Maya to its full potential (e.g., handling a complex 3D scene), then they will be unable to make an fully-informed assessment of the product. This substantially limits the audience for such high-end applications. It also contributes to a high selling price since the development cost is

usually amortized across a much smaller number of purchases than those of a general-purpose application.

High-priced applications also create more incentive for individuals and businesses to use pirated copies of the application software. As a result, high-end application software suffers from rampant piracy, despite significant efforts by publishers of such software to mitigate such piracy through various techniques. Still, even when using pirated high-end applications, users cannot obviate the need to invest in expensive state-of-the-art PCs to run the pirated copies. So, while they may obtain use of a software application for a fraction of its actual retail price, users of pirated software are still required to purchase or obtain an expensive PC in order to fully utilize the application.

The same is true for users of high-performance pirated video games. Although pirates may get the games at fraction of their actual price, they are still required to purchase expensive computing hardware (e.g., a GPU-enhanced PC, or a high-end video game console like the Xbox 360) needed to properly play the game. Given that video games are typically a pastime for consumers, the additional cost for a high-end video game system can be prohibitive. This situation is worse in countries (e.g., China) where the average annual income of workers currently is quite low relative to that of the United States. As a result, a much smaller percentage of the population owns a high-end video game system or a high-end PC. In such countries, "Internet cafes", in which users pay a fee to use a computer connected to the Internet, are quite common. Frequently, such Internet cafes have older model or low-end PCs without high performance features, such as a GPU, which might otherwise enable players to play computationally-intensive video games. This is a key factor in the success of games that run on low-end PCs, such as Vivendi's "World of Warcraft" which is highly successful in China, and is commonly played in

Internet cafes there. In contrast, a computationally-intensive game, like “Second Life” is much less likely to be playable on a PC installed in a Chinese Internet café. Such games are virtually inaccessible to users who only have access to low-performance PCs in Internet cafes.

Barriers also exist for users who are considering purchasing a video game and would first like to try out a demonstration version of the game by downloading the demo through the Internet to their home. A video game demo is often a full-fledged version of the game with some features disabled, or with limits placed on the amount of game play. This may involve a long process (perhaps hours) of downloading gigabytes of data before the game can be installed and executed on either a PC or a console. In the case of a PC, it may also involve figuring out what special drivers are needed (e.g., DirectX or OpenGL drivers) for the game, downloading the correct version, installing them, and then determining whether the PC is capable of playing the game. This latter step may involve determining whether the PC has enough processing (CPU and GPU) capability, sufficient RAM, and a compatible OS (e.g., some games run on Windows XP, but not Vista). Thus, after a long process of attempting to run a video game demo, the user may well find out that the video game demo can't be possibly played, given the user's PC configuration. Worse, once the user has downloaded new drivers in order to try the demo, these driver versions may be incompatible with other games or applications the user uses regularly on the PC, thus the installation of a demo may render previously operable games or applications inoperable. Not only are these barriers frustrating for the user, but they create barriers for video game software publishers and video game developers to market their games.

Another problem that results in economic inefficiency has to do with the fact that given PC or game console is usually designed to accommodate a

certain level of performance requirement for applications and/or games. For example, some PCs have more or less RAM, slower or faster CPUs, and slower or faster GPUs, if they have a GPUs at all. Some games or applications make take advantage of the full computing power of a given PC or console, while many games or applications do not. If a user's choice of game or application falls short of the peak performance capabilities of the local PC or console, then the user may have wasted money on the PC or console for unutilized features. In the case of a console, the console manufacturer may have paid more than was necessary to subsidize the console cost.

Another problem that exists in the marketing and enjoyment of video games involves allowing a user to watch others playing games before the user commits to the purchase of that game. Several prior art approaches exist for the recording of video games for replay at a later time. For example, U.S. Patent No. 5,558,339 teaches recording game state information, including game controller actions, during "gameplay" in the video game client computer (owned by the same or different user). This state information can be used at a later time to replay some or all of the game action on a video game client computer (e.g., PC or console). A significant drawback to this approach is that for a user to view the recorded game, the user must possess a video game client computer capable of playing the game and must have the video game application running on that computer, such that the gameplay is identical when the recorded game state is replayed. Beyond that, the video game application has to be written in such a way that there is no possible execution difference between the recorded game and the played back game.

For example, game graphics are generally computed on a frame-by-frame basis. For many games, the game logic sometimes may take shorter or longer than one frame time to compute the graphics displayed for the next frame,

depending on whether the scene is particularly complex, or if there are other delays that slow down execution (e.g., on a PC, another process may be running that takes away CPU cycles from the game applications). In such a game, a "threshold" frame that is computed in slightly less than one frame time (say a few CPU clock cycles less) can eventually occur. When that same scene is computed again using the exact same game state information, it could easily take a few CPU clock cycles more than one frame time (e.g., if an internal CPU bus is slightly out of phase with the an external DRAM bus and it introduces a few CPU cycle times of delay, even if there is no large delay from another process taking away milliseconds of CPU time from game processing). Therefore, when the game is played back the frame gets calculated in two frame times rather than a single frame time. Some behaviors are based on how often the game calculates a new frame (e.g., when the game samples the input from the game controllers). While the game is played, this discrepancy in the time reference for different behaviors does not impact game play, but it can result in the played-back game producing a different result. For example, if a basketball's ballistics are calculated at a steady 60 fps rate, but the game controller input is sampled based on rate of computed frames, the rate of computed frames may be 53 fps when the game was recorded, but 52 fps when the game is replayed, which can make the difference between whether the basketball is blocked from going into the basket or not, resulting in a different outcome. Thus, using game state to record video games requires very careful game software design to ensure that the replay, using the same game state information, produces the exact same outcome.

Another prior art approach for recording video game is to simply record the video output of a PC or video game system (e.g., to a VCR, DVD recorder, or to a video capture board on a PC). The video then can be rewound and replayed, or alternatively, the recorded video uploaded to the Internet, typically after being

compressed. A disadvantage to this approach is that when a 3D game sequence is played back, the user is limited to viewing the sequence from only the point of view from which the sequence was recorded. In other words, the user cannot change the point of view of the scene.

Further, when compressed video of a recorded game sequence played on a home PC or game console is made available to other users through the Internet, even if the video is compressed in real-time, it may be impossible to upload the compressed video in real-time to the Internet. The reason why is because many homes in the world that are connected to the Internet have highly asymmetric broadband connections (e.g., DSL and cable modems typically have far higher downstream bandwidth than upstream bandwidth). Compressed high resolution video sequences often have higher bandwidths than the upstream bandwidth capacity of the network, making them impossible to upload in real-time. Thus, there would be a significant delay after the game sequence is played (perhaps minutes or even hours) before another user on the Internet would be able to view the game. Although this delay is tolerable in certain situations (e.g., to watch a game player's accomplishments that occurred at a prior time), it eliminates the ability to watch a game live (e.g., a basketball tournament, played by champion players) or with "instant replay" capability as the game is played live.

Another prior art approach allows a viewer with a television receiver to watch video games live, but only under the control of the television production crew. Some television channels, in both the US and in other countries provide video game viewing channels, where the television viewing audience is able to watch certain video game users (e.g., top-rated players playing in tournaments) on video game channels. This is accomplished by having the video output of the video game systems (PCs and/or consoles) fed into the video distribution and processing

equipment for the television channel. This is not unlike when the television channel is broadcasting a live basketball game in which several cameras provide live feeds from different angles around the basketball court. The television channel then is able to make use of their video/audio processing and effects equipment to manipulate the output from the various video game systems. For example, the television channel can overlay text on top of the video from a video game that indicates the status of different players (just as they might overlay text during a live basketball game), and the television channel can overdub audio from a commentator who can discuss the action occurring during the games. Additionally, the video game output can be combined with cameras recording video of the actual players of the games (e.g., showing their emotional response to the game).

One problem with this approach is that such live video feeds must be available to the television channel's video distribution and processing equipment in real-time in order for it to have the excitement of a live broadcast. As previously discussed, however, this is often impossible when the video game system is running from the home, especially if part of the broadcast includes live video from a camera that is capturing real-world video of the game player. Further, in a tournament situation, there is a concern that an in-home gamer may modify the game and cheat, as previously described. For these reasons, such video game broadcasts on television channels are often arranged with players and video game systems aggregated at a common location (e.g., at a television studio or in an arena) where the television production equipment can accept video feeds from multiple video game systems and potentially live cameras.

Although such prior art video game television channels can provide a very exciting presentation to the television viewing audience that is an experience akin to a live sporting event, e.g., with the video game players presented as

“athletes”, both in terms of their actions in the video game world, and in terms of their actions in the real world, these video game systems are often limited to situations where players are in close physical proximity to one another. And, since television channels are broadcasted, each broadcasted channel can only show one video stream, which is selected by the television channel's production crew. Because of these limitations and the high cost of broadcast time, production equipment and production crews, such television channels typically only show top-rated players playing in top tournaments.

Additionally, a given television channel broadcasting a full-screen image of a video game to the entire television viewing audience shows only one video game at a time. This severely limits a television viewer's choices. For example, a television viewer may not be interested in the game(s) shown at a given time. Another viewer may only be interested in watching the game play of a particular player that is not featured by the television channel at a given time. In other cases, a viewer may only be interested in watching a how an expert player handles a particular level in a game. Still other viewers may wish to control the viewpoint that a video game is seen from, which is different from that chosen by the production team, etc. In short, a television viewer may have a myriad of preferences in watching video games that are not accommodated by the particular broadcast of a television network, even if several different television channels are available. For all of the aforementioned reasons, prior art video game television channels have significant limitations in presenting video games to television viewers.

Another drawback of prior art video games systems and application software systems is that they are complex, and commonly suffer from errors, crashes and/or unintended and undesired behaviors (collectively, “bugs”). Although games and applications typically go through a debugging and tuning process

(frequently called “Software Quality Assurance” or SQA) before release, almost invariably once the game or application is released to a wide audience in the field bugs crop up. Unfortunately, it is difficult for the software developer to identify and track down many of the bugs after release. It can be difficult for software developers to become aware of bugs. Even when they learn about a bug, there may only be a limited amount of information available to them to identify what caused the bug. For example, a user may call up a game developer’s customer service line and leave a message stating that when playing the game, the screen started to flash, then changed to a solid blue color and the PC froze. That provides the SQA team with very little information useful in tracking down a bug. Some games or applications that are connected online can sometimes provide more information in certain cases. For example, a “watchdog” process can sometimes be used to monitor the game or application for “crashes”. The watchdog process can gather statistics about the status of the game or applications process (e.g., the status of the stack, of the memory usage, how far the game or applications has progressed, etc.) when it crashes and then upload that information to the SQA team via the Internet. But in a complex game or application, such information can take a very long time to decipher in order to accurately determine what the user was doing at the time of the crash. Even then, it may be impossible to determine what sequence of events led to the crash.

Yet another problem associated with PCs and game consoles is that they are subject to service issues which greatly inconvenience the consumer. Service issues also impact the manufacturer of the PC or game console since they typically are required to send a special box to safely ship the broken PC or console, and then incur the cost of repair if the PC or console is in warranty. The game or

application software publisher can also be impacted by the loss of sales (or online service use) by PCs and/or consoles being in a state of repair.

Figure 1 illustrates a prior art video gaming system such as a Sony Playstation® 3, Microsoft Xbox 360®, Nintendo Wii™, Windows-based personal computer or Apple Macintosh. Each of these systems includes a central processing unit (CPU) for executing program code, typically a graphical processing unit (GPU) for performing advanced graphical operations, and multiple forms of input/output (I/O) for communicating with external devices and users. For simplicity, these components are shown combined together as a single unit 100. The prior art video gaming system of Figure 1 also is shown including an optical media drive 104 (e.g., a DVD-ROM drive); a hard drive 103 for storing video game program code and data; a network connection 105 for playing multi-player games, for downloading games, patches, demos or other media; a random access memory (RAM) 101 for storing program code currently being executed by the CPU/GPU 100; a game controller 106 for receiving input commands from the user during gameplay; and a display device 102 (e.g., a SDTV/HDTV or a computer monitor).

The prior art system shown in Figure 1 suffers from several limitations. First, optical drives 104 and hard drives 103 tend to have much slower access speeds as compared to that of RAM 101. When working directly through RAM 101, the CPU/GPU 100 can, in practice, process far more polygons per second than is possible when the program code and data is read directly off of hard drive 103 or optical drive 104 due to the fact that RAM 101 generally has much higher bandwidth and does not suffer from the relatively long seek delays of disc mechanisms. But only a limited amount of RAM is provided in these prior art systems (e.g., 256-512Mbytes). Therefore, a "Loading..." sequence in which RAM 101 is periodically filled up with the data for the next scene of the video game is often required.

Some systems attempt to overlap the loading of the program code concurrently with the gameplay, but this can only be done when there is a known sequence of events (e.g., if a car is driving down a road, the geometry for the approaching buildings on the roadside can be loaded while the car is driving). For complex and/or rapid scene changes, this type of overlapping usually does not work. For example, in the case where the user is in the midst of a battle and RAM 101 is completely filled with data representing the objects within view at that moment, if the user moves the view rapidly to the left to view objects that are not presently loaded in RAM 101, a discontinuity in the action will result since there not be enough time to load the new objects from Hard Drive 103 or Optical Media 104 into RAM 101.

Another problem with the system of Figure 1 arises due to limitations in the storage capacity of hard drives 103 and optical media 104. Although disk storage devices can be manufactured with a relatively large storage capacity (e.g., 50 gigabytes or more), they still do not provide enough storage capacity for certain scenarios encountered in current video games. For example, as previously mentioned, a soccer video game might allow the user to choose among dozens of teams, players and stadiums throughout the world. For each team, each player and each stadium a large number of texture maps and environment maps are needed to characterize the 3D surfaces in the world (e.g., each team has a unique jersey, with each requiring a unique texture map).

One technique used to address this latter problem is for the game to pre-compute texture and environment maps once they are selected by the user. This may involve a number of computationally-intensive processes, including decompressing images, 3D mapping, shading, organizing data structures, etc. As a result, there may be a delay for the user while the video game is performing these calculations. One way to reduce this delay, in principle, is to perform all of these

computations – including every permutation of team, player roster, and stadium – when the game was originally developed. The released version of the game would then include all of this pre-processed data stored on optical media 104, or on one or more servers on the Internet with just the selected pre-processed data for a given team, player roster, stadium selection downloaded through the Internet to hard drive 103 when the user makes a selection. As a practical matter, however, such pre-loaded data of every permutation possible in game play could easily be terabytes of data, which is far in excess of the capacity of today's optical media devices. Furthermore, the data for a given team, player roster, stadium selection could easily be hundreds of megabytes of data or more. With a home network connection of, say, 10Mbps, it would take longer to download this data through network connection 105 than it would to compute the data locally.

Thus, the prior art game architecture shown in Figure 1 subjects the user to significant delays between major scene transitions of complex games.

Another problem with prior art approaches such as that shown in Figure 1 is that over the years video games tend to become more advanced and require more CPU/GPU processing power. Thus, even assuming an unlimited amount of RAM, video games hardware requirements go beyond the peak level of processing power available in these systems. As a result, users are required to upgrade gaming hardware every few years to keep pace (or play newer games at lower quality levels). One consequence of the trend to ever more advanced video games is that video game playing machines for home use are typically economically inefficient because their cost is usually determined by the requirements of the highest performance game they can support. For example, an XBox 360 might be used to play a game like "Gears of War", which demands a high performance CPU, GPU, and hundreds of megabytes of RAM, or the XBox 360 might be used to play

Pac Man, a game from the 1970s that requires only kilobytes of RAM and a very low performance CPU. Indeed, an XBox 360 has enough computing power to host many simultaneous Pac Man games at once.

Video games machines are typically turned off for most of the hours of a week. According to a July 2006 Nielsen Entertainment study of active gamers 13 years and older, on average, active gamers spend fourteen hours/week playing console video games, or just 12% of the total hours in a week. This means that the average video game console is idle 88% of the time, which is an inefficient use of an expensive resource. This is particularly significant given that video game consoles are often subsidized by the manufacturer to bring down the purchase price (with the expectation that the subsidy will be earned back by royalties from future video game software purchases).

Video game consoles also incur costs associated with almost any consumer electronic device. For instance, the electronics and mechanisms of the systems need to be housed in an enclosure. The manufacturer needs to offer a service warranty. The retailer who sells the system needs to collect a margin on either the sale of the system and/or on the sale of video game software. All of these factors add to the cost of the video game console, which must either be subsidized by the manufacturer, passed along to the consumer, or both.

In addition, piracy is a major problem for the video game industry. The security mechanisms utilized on virtually every major video gaming system have been "cracked" over the years, resulting in unauthorized copying of video games. For example, the Xbox 360 security system was cracked in July 2006 and users are now able to download illegal copies online. Games that are downloadable (e.g., games for the PC or the Mac) are particularly vulnerable to piracy. In certain regions of the world where piracy is weakly policed there is essentially no viable market for

standalone video game software because users can buy pirated copies as readily as legal copies for a tiny fraction of the cost. Also, in many parts of the world the cost of a game console is such a high percentage of income that even if piracy were controlled, few people could afford a state-of-the-art gaming system.

In addition, the used game market reduces revenue for the video game industry. When a user has become tired of a game, they can sell the game to a store which will resell the game to other users. This unauthorized but common practice significantly reduces revenues of game publishers. Similarly, a reduction in sales on the order of 50% commonly occurs when there is a platform transition every few years. This is because users stop buying games for the older platforms when they know that the newer version platform is about to be released (e.g., when Playstation 3 is about to be released, users stop buying Playstation 2 games). Combined, the loss of sales and increased development costs associated with the new platforms can have a very significant adverse impact on the profitability of game developers.

New game consoles are also very expensive. The Xbox 360, the Nintendo Wii, and the Sony Playstation 3 all retail for hundreds of dollars. High powered personal computer gaming systems can cost up to \$8000. This represents a significant investment for users, particularly considering that the hardware becomes obsolete after a few years and the fact that many systems are purchased for children.

One approach to the foregoing problems is online gaming in which the gaming program code and data are hosted on a server and delivered to client machines on-demand as compressed video and audio streamed over a digital broadband network. Some companies such as G-Cluster in Finland (now a subsidiary of Japan's SOFTBANK Broadmedia) currently provide these services

online. Similar gaming services have become available in local networks, such as those within hotels and offered by DSL and cable television providers. A major drawback of these systems is the problem of latency, i.e., the time it takes for a signal to travel to and from the game server, which is typically located in an operator's "head-end". Fast action video games (also known as "twitch" video games) require very low latency between the time the user performs an action with the game controller and the time the display screen is updated showing the result of the user action. Low latency is needed so that the user has the perception that the game is responding "instantly". Users may be satisfied with different latency intervals depending on the type of game and the skill level of the user. For example, 100ms of latency may be tolerable for a slow casual game (like backgammon) or a slow-action role playing game, but in a fast action game a latency in excess of 70 or 80ms may cause the user to perform more poorly in the game, and thus is unacceptable. For instance, in a game that requires fast reaction time there is a sharp decline in accuracy as latency increases from 50 to 100ms.

When a game or application server is installed in a nearby, controlled network environment, or one where the network path to the user is predictable and/or can tolerate bandwidth peaks, it is far easier to control latency, both in terms of maximum latency and in terms of the consistency of the latency (e.g., so the user observes steady motion from digital video streaming through the network). Such level of control can be achieved between a cable TV network head-end to a cable TV subscriber's home, or from a DSL central office to DSL subscriber's home, or in a commercial office Local Area Network (LAN) environment from a server or a user. Also, it is possible to obtain specially-graded point-to-point private connections between businesses which have guaranteed bandwidth and latency. But in a game or application system that hosts games in a server center connected to the general

Internet and then streams compressed video to the user through a broadband connection, latency is incurred from many factors, resulting in severe limitations in the deployment of prior art systems.

In a typical broadband-connected home, a user may have a DSL or cable modem for broadband service. Such broadband services commonly incur as much as a 25ms round-trip latency (and at times more) between the user's home and the general Internet. In addition, there are round-trip latencies incurred from routing data through the Internet to a server center. The latency through the Internet varies based on the route that the data is given and the delays it incurs as it is routed. In addition to routing delays, round-trip latency is also incurred due to the speed of light traveling through the optical fiber that interconnects most of the Internet. For example, for each 1000 miles, approximately 22ms is incurred in round-trip latency due to the speed of light through the optical fiber and other overhead.

Additional latency can occur due to the data rate of the data streamed through the Internet. For example, if a user has DSL service that is sold as "6Mbps DSL service", in practice, the user will probably get less than 5Mbps of downstream throughput at best, and will likely see the connection degrade periodically due to various factors such as congestion during peak load times at the Digital Subscriber Line Access Multiplexer (DSLAM). A similar issue can occur reducing a the data rate of a cable modem is used for a connection sold as "6Mbps cable modem service" to far less than that, if there is congestion in the local shared coaxial cable looped through the neighborhood, or elsewhere in the cable modem system network. If data packets at a steady rate of 4Mbps are streamed as one-way in User Datagram Protocol (UDP) format from a server center through such connections, if everything is working well, the data packets will pass through without incurring additional latency, but if there is congestion (or other impediments) and only 3.5Mbps is

available to stream data to the user, then in a typical situation either packets will be dropped, resulting in lost data, or packets will queue up at the point of congestion, until they can be sent, thereby introducing additional latency. Different points of congestion have different queuing capacity to hold delayed packets, so in some cases packets that can't make it through the congestion are dropped immediately. In other cases, several megabits of data are queued up and eventually be sent. But, in almost all cases, queues at points of congestion have capacity limits, and once those limits are exceeded, the queues will overflow and packets will be dropped. Thus, to avoid incurring additional latency (or worse, loss of packets), it is necessary to avoid exceeding the data rate capacity from the game or application server to the user.

Latency is also incurred by the time required to compress video in the server and decompress video in the client device. Latency is further incurred while a video game running on a server is calculating the next frame to be displayed. Currently available video compression algorithms suffer from either high data rates or high latency. For example, motion JPEG is an intraframe-only lossy compression algorithm that is characterized by low-latency. Each frame of video is compressed independently of each other frame of video. When a client device receives a frame of compressed motion JPEG video, it can immediately decompress the frame and display it, resulting in very low latency. But because each frame is compressed separately, the algorithm is unable to exploit similarities between successive frames, and as a result intraframe-only video compression algorithms suffer from very high data rates. For example, 60 fps (frames per second) 640x480 motion JPEG video may require 40Mbps (megabits per second) or more of data. Such high data rates for such low resolution video windows would be prohibitively expensive in many broadband applications (and certainly for most consumer Internet-based

applications). Further, because each frame is compressed independently, artifacts in the frames that may result from the lossy compression are likely to appear in different places in successive frames. This can result in what appears to the viewer as a moving visual artifacts when the video is decompressed.

Other compression algorithms, such as MPEG2, H.264 or VC9 from Microsoft Corporation as they are used in prior art configurations, can achieve high compression ratios, but at the cost of high latency. Such algorithms utilize interframe as well as intraframe compression. Periodically, such algorithms perform an intraframe-only compression of a frame. Such a frame is known as a key frame (typically referred to as an "I" frame). Then, these algorithms typically compare the I frame with both prior frames and successive frames. Rather than compressing the prior frames and successive frames independently, the algorithm determines what has changed in the image from the I frame to the prior and successive frames, and then stores those changes as what are called "B" frames for the changes preceding the I frame and "P" frames for the changes following the I frame. This results in much lower data rates than intraframe-only compression. But, it typically comes at the cost of higher latency. An I frame is typically much larger than a B or P frame (often 10 times larger), and as a result, it takes proportionately longer to transmit at a given data rate.

Consider, for example, a situation where the I frames are 10X the size of B and P frames, and there are 29 B frames + 30 P frames = 59 interframes for every single I intraframe, or 60 frames total for each "Group of Frames" (GOP). So, at 60 fps, there is 1 60-frame GOP each second. Suppose the transmission channel has a maximum data rate of 2Mbps. To achieve the highest quality video in the channel, the compression algorithm would produce a 2Mbps data stream, and given the above ratios, this would result in $2 \text{ Megabits (Mb)} / (59+10) = 30,394$ bits per

intraframe and 303,935 bits per I frame. When the compressed video stream is received by the decompression algorithm, in order for the video to play steadily, each frame needs to be decompressed and displayed at a regular interval (e.g., 60 fps). To achieve this result, if any frame is subject to transmission latency, all of the frames need to be delayed by at least that latency, so the worst-case frame latency will define the latency for every video frame. The I frames introduce the longest transmission latencies since they are largest, and an entire I frame would have to be received before the I frame could be decompressed and displayed (or any interframe dependent on the I frame). Given that the channel data rate is 2Mbps, it will take $303,935/2\text{Mb} = 145\text{ms}$ to transmit an I frame.

An interframe video compression system as described above using a large percentage of the bandwidth of the transmission channel will be subject to long latencies due to the large size of an I frame relative to the average size of a frame. Or, to put it another way, while prior art interframe compression algorithms achieve a lower average per-frame data rate than intraframe-only compression algorithms (e.g., 2Mbps vs. 40Mbps), they still suffer from a high peak per-frame data rate (e.g., $303,935 * 60 = 18.2\text{Mbps}$) because of the large I frames. Bear in mind, though that the above analysis assumes that the P and B frames are all much smaller than the I frames. While this is generally true, it is not true for frames with high image complexity uncorrelated with the prior frame, high motion, or scene changes. In such situations, the P or B frames can become as large as I frames (if a P or B frame gets larger than an I frame, a sophisticated compression algorithm will typically “force” an I frame and replace the P or B frame with an I frame). So, I frame-sized data rate peaks can occur at any moment in a digital video stream. Thus, with compressed video, when the average video data rate approaches data rate capacity of the transmission channels (as is frequently the case, given the high data rate demands

for video) the high peak data rates from I frames or large P or B frames result in a high frame latency.

Of course, the above discussion only characterizes the compression algorithm latency created by large B, P or I frames in a GOP. If B frames are used, the latency will be even higher. The reason why is because before a B frame can be displayed, all of the B frames after the B frame and the I frame must be received. Thus, in a group of picture (GOP) sequence such as BBBBBIPPPPPBBBBBIPPPPP, where there are 5 B frames before each I frame, the first B frame can not be displayed by the video decompressor until the subsequent B frames and I frame are received. So, if video is being streamed at 60fps (i.e., 16.67ms/frame), before the first B frame can be decompressed, five B frames and the I frame will take $16.67 * 6 = 100\text{ms}$ to receive, no matter how fast the channel bandwidth is, and this is with just 5 B frames. Compressed video sequences with 30 B frames are quite common. And, at a low channel bandwidth like 2Mbps, the latency impact caused by the size of the I frame is largely additive to the latency impact due to waiting for B frames to arrive. Thus, on a 2Mbps channel, with a large number of B frames it is quite easy to exceed 500ms of latency or more using prior art video compression technology. If B frames are not used (at the cost of a lower compression ratio for given quality level), the B frame latency is not incurred, but the latency caused by the peak frame sizes, described above, is still incurred.

The problem is exacerbated by very the nature of many video games. Video compression algorithms utilizing the GOP structure described above have been largely optimized for use with live video or motion picture material intended for passive viewing. Typically, the camera (whether a real camera, or a virtual camera in the case of a computer-generated animation) and scene is relatively steady, simply because if the camera or scene moves around too jerkily, the video or movie

material is (a) typically unpleasant to watch and (b) if it is being watched, usually the viewer is not closely following the action when the camera jerks around suddenly (e.g., if the camera is bumped when shooting a child blowing out the candles on a birthday cake and suddenly jerks away from the cake and back again, the viewers are typically focused on the child and the cake, and disregard the brief interruption when the camera suddenly moves). In the case of a video interview, or a video teleconference, the camera may be held in a fixed position and not move at all, resulting in very few data peaks at all. But 3D high action video games are characterized by constant motion (e.g., consider a 3D racing, where the entire frame is in rapid motion for the duration of the race, or consider first-person shooters, where the virtual camera is constantly moving around jerkily). Such video games can result in frame sequences with large and frequent peaks where the user may need to clearly see what is happening during those sudden motions. As such, compression artifacts are far less tolerable in 3D high action video games. Thus, the video output of many video games, by their nature, produces a compressed video stream with very high and frequent peaks.

Given that users of fast-action video games have little tolerance for high latency, and given all of the above causes of latency, to date there have been limitations to server-hosted video games that stream video on the Internet. Further, users of applications that require a high degree of interactivity suffer from similar limitations if the applications are hosted on the general Internet and stream video. Such services require a network configuration in which the hosting servers are set up directly in a head end (in the case of cable broadband) or the central office (in the case of Digital Subscriber Lines (DSL)), or within a LAN (or on a specially-graded private connection) in a commercial setting, so that the route and distance from the client device to the server is controlled to minimize latency and peaks can be

accommodated without incurring latency. LANs (typically rated at 100Mbps-1Gbps) and leased lines with adequate bandwidth typically can support peak bandwidth requirements (e.g., 18Mbps peak bandwidth is a small fraction of a 100Mbps LAN capacity).

Peak bandwidth requirements can also be accommodated by residential broadband infrastructure if special accommodations are made. For example, on a cable TV system, digital video traffic can be given dedicated bandwidth which can handle peaks, such as large I frames. And, on a DSL system, a higher speed DSL modem can be provisioned, allowing for high peaks, or a specially-graded connection can be provisioned which can handle higher data rates. But, conventional cable modem and DSL infrastructure attached to the general Internet have far less tolerance for peak bandwidth requirements for compressed video. So, online services that host video games or applications in server centers a long distance from the client devices, and then stream the compressed video output over the Internet through conventional residential broadband connections suffer from significant latency and peak bandwidth limitations – particularly with respect to games and applications which require very low latency (e.g., first person shooters and other multi-user, interactive action games, or applications requiring a fast response time).

BRIEF DESCRIPTION OF THE DRAWINGS

The present disclosure will be understood more fully from the detailed description that follows and from the accompanying drawings, which however, should not be taken to limit the disclosed subject matter to the specific embodiments shown, but are for explanation and understanding only.

FIG. 1 illustrates an architecture of a prior art video gaming system.

FIGS. 2a-b illustrate a high level system architecture according to one embodiment.

FIG. 3 illustrates actual, rated, and required data rates for communication between a client and a server.

FIG. 4a illustrates a hosting service and a client employed according to one embodiment.

FIG. 4b illustrates exemplary latencies associated with communication between a client and hosting service.

FIG 4c illustrates a client device according to one embodiment.

FIG 4d illustrates a client device according to another embodiment.

FIG 4e illustrates an example block diagram of the client device in Figure 4c.

FIG 4f illustrates an example block diagram of the client device in Figure 4d.

FIG. 5 illustrates an example form of video compression which may be employed according to one embodiment.

FIG. 6a illustrates an example form of video compression which may be employed in another embodiment.

FIG. 6b illustrates peaks in data rate associated with transmitting a low complexity, low action video sequence.

FIG. 6c illustrates peaks in data rate associated with transmitting a high complexity, high action video sequence.

FIGS. 7a-b illustrate example video compression techniques employed in one embodiment.

FIG. 8 illustrates additional example video compression techniques employed in one embodiment.

FIGS. 9a-c illustrate frame rate processing techniques employed in one embodiment of the invention.

FIGS. 10a-b illustrate one embodiment which efficiently packs image tiles within packets.

FIGS. 11a-d illustrate embodiments which employ forward error correction techniques.

FIG. 12 illustrates one embodiment which uses multi-core processing units for compression.

FIGS. 13a-b illustrate geographical positioning and communication between hosting services according to various embodiments.

FIG. 14 illustrates exemplary latencies associated with communication between a client and a hosting service.

FIG. 15 illustrates an example hosting service server center architecture.

FIG. 16 illustrates an example screen shot of one embodiment of a user interface which includes a plurality of live video windows.

FIG. 17 illustrates the user interface of Figure 16 following the selection of a particular video window.

FIG. 18 illustrates the user interface of Figure 17 following zooming of the particular video window to full screen size.

FIG. 19 illustrates an example collaborative user video data overlaid on the screen of a multiplayer game.

FIG. 20 illustrates an example user page for a game player on a hosting service.

FIG. 21 illustrates an example 3D interactive advertisement.

FIG. 22 illustrates an example sequence of steps for producing a photoreal image having a textured surface from surface capture of a live performance.

FIG. 23 illustrates an example user interface page that allows for selection of linear media content.

FIG. 24 is a graph that illustrates the amount of time that elapses before the web page is live versus connection speed.

FIGS. 25a-b illustrates embodiments of the invention which employ a feedback channel from the client device to the hosting service.

FIGS. 26a-b illustrate an embodiment in which encodes tiles/frames based on the last known tile/frame to have been successfully received.

FIGS. 27a-b illustrate an embodiment in which the state of a game or application is ported from a first hosting service or server to a second hosting service or server.

FIG. 28 illustrates one embodiment in which the state of a game or application is ported using difference data.

FIG. 29 illustrates one embodiment of the invention which employs a temporary decoder on the client device.

FIG. 30 illustrates how "I tiles" are interspersed across "R frames" according to one embodiment of the invention.

FIGS. 31a-h illustrate embodiments of the invention which generate a live stream and/or one or more HQ streams.

DESCRIPTION OF EXAMPLE EMBODIMENTS

In the following description specific details are set forth, such as device types, system configurations, communication methods, etc., in order to provide a thorough understanding of the present disclosure. However, persons having ordinary skill in the relevant arts will appreciate that these specific details may not be needed to practice the embodiments described.

Figures 2a-b provide a high-level architecture of two embodiments in which video games and software applications are hosted by a hosting service 210 and accessed by client devices 205 at user premises 211 (note that the “user premises” means the place wherever the user is located, including outdoors if using a mobile device) over the Internet 206 (or other public or private network) under a subscription service. The client devices 205 may be general-purpose computers such as Microsoft Windows- or Linux-based PCs or Apple, Inc. Macintosh computers with a wired or wireless connection to the Internet either with internal or external display device 222, or they may be dedicated client devices such as a set-top box (with a wired or wireless connection to the Internet) that outputs video and audio to a monitor or TV set 222, or they may be mobile devices, presumably with a wireless connection to the Internet.

Any of these devices may have their own user input devices (e.g., keyboards, buttons, touch screens, track pads or inertial-sensing wands, video capture cameras and/or motion-tracking cameras, etc.), or they may use external input devices 221 (e.g., keyboards, mice, game controllers, inertial sensing wand, video capture cameras and/or motion tracking cameras, etc.), connected with wires or wirelessly. As described in greater detail below, the hosting service 210 includes servers of various levels of performance, including those with high-powered

CPU/GPU processing capabilities. During playing of a game or use of an application on the hosting service 210, a home or office client device 205 receives keyboard and/or controller input from the user, and then it transmits the controller input through the Internet 206 to the hosting service 210 that executes the gaming program code in response and generates successive frames of video output (a sequence of video images) for the game or application software (e.g., if the user presses a button which would direct a character on the screen to move to the right, the game program would then create a sequence of video images showing the character moving to the right). This sequence of video images is then compressed using a low-latency video compressor, and the hosting service 210 then transmits the low-latency video stream through the Internet 206. The home or office client device then decodes the compressed video stream and renders the decompressed video images on a monitor or TV. Consequently, the computing and graphical hardware requirements of the client device 205 are significantly reduced. The client 205 only needs to have the processing power to forward the keyboard/controller input to the Internet 206 and decode and decompress a compressed video stream received from the Internet 206, which virtually any personal computer is capable of doing today in software on its CPU (e.g., a Intel Corporation Core Duo CPU running at approximately 2GHz is capable of decompressing 720p HDTV encoded using compressors such as H.264 and Windows Media VC9). And, in the case of any client devices, dedicated chips can also perform video decompression for such standards in real-time at far lower cost and with far less power consumption than a general-purpose CPU such as would be required for a modern PC. Notably, to perform the function of forwarding controller input and decompressing video, home client devices 205 do not require any specialized graphics processing units (GPUs),

optical drive or hard drives, such as the prior art video game system shown in Figure 1.

As games and applications software become more complex and more photo-realistic, they will require higher-performance CPUs, GPUs, more RAM, and larger and faster disk drives, and the computing power at the hosting service 210 may be continually upgraded, but the end user will not be required to update the home or office client platform 205 since its processing requirements will remain constant for a display resolution and frame rate with a given video decompression algorithm. Thus, the hardware limitations and compatibility issues seen today do not exist in the system illustrated in Figures 2a-b.

Further, because the game and application software executes only in servers in the hosting service 210, there never is a copy of the game or application software (either in the form of optical media, or as downloaded software) in the user's home or office ("office" as used herein unless otherwise qualified shall include any non-residential setting, including, schoolrooms, for example). This significantly mitigates the likelihood of a game or application software being illegally copied (pirated), as well as mitigating the likelihood of a valuable database that might be used by a game or applications software being pirated. Indeed, if specialized servers are required (e.g., requiring very expensive, large or noisy equipment) to play the game or application software that are not practical for home or office use, then even if a pirated copy of the game or application software were obtained, it would not be operable in the home or office.

In one embodiment, the hosting service 210 provides software development tools to the game or application software developers (which refers generally to software development companies, game or movie studios, or game or applications software publishers) 220 which design video games so that they may

design games capable of being executed on the hosting service 210. Such tools allow developers to exploit features of the hosting service that would not normally be available in a standalone PC or game console (e.g., fast access to very large databases of complex geometry ("geometry" unless otherwise qualified shall be used herein to refer to polygons, textures, rigging, lighting, behaviors and other components and parameters that define 3D datasets)).

Different business models are possible under this architecture. Under one model, the hosting service 210 collects a subscription fee from the end user and pays a royalty to the developers 220, as shown in Figure 2a. In an alternate implementation, shown in Figure 2b, the developers 220 collect a subscription fee directly from the user and pay the hosting service 210 for hosting the game or application content. These underlying principles are not limited to any particular business model for providing online gaming or application hosting.

COMPRESSED VIDEO CHARACTERISTICS

As discussed previously, one significant problem with providing video game services or applications software services online is that of latency. A latency of 70-80ms (from the point a input device is actuated by the user to the point where a response is displayed on the display device) is at the upper limit for games and applications requiring a fast response time. However, this is very difficult to achieve in the context of the architecture shown in Figures 2a and 2b due to a number of practical and physical constraints.

As indicated in Figure 3, when a user subscribes to an Internet service, the connection is typically rated by a nominal maximum data rate 301 to the user's home or office. Depending on the provider's policies and routing equipment capabilities, that maximum data rate may be more or less strictly enforced, but typically the actual available data rate is lower for one of many different reasons.

For example, there may be too much network traffic at the DSL central office or on the local cable modem loop, or there may be noise on the cabling causing dropped packets, or the provider may establish a maximum number of bits per month per user. Currently, the maximum downstream data rate for cable and DSL services typically ranges from several hundred Kilobits/second (Kbps) to 30 Mbps. Cellular services are typically limited to hundreds of Kbps of downstream data. However, the speed of the broadband services and the number of users who subscribe to broadband services will increase dramatically over time. Currently, some analysts estimate that 33% of US broadband subscribers have a downstream data rate of 2Mbps or more. For example, some analysts predict that by 2010, over 85% of US broadband subscribers will have a data rate of 2Mbps or more.

As indicated in Figure 3, the actual available max data rate 302 may fluctuate over time. Thus, in a low-latency, online gaming or application software context it is sometimes difficult to predict the actual available data rate for a particular video stream. If the data rate 303 required to sustain a given level of quality at given number of frames-per-second (fps) at a given resolution (e.g., 640 x 480 @ 60 fps) for a certain amount of scene complexity and motion rises above the actual available max data rate 302 (as indicated by the peak in Figure 3), then several problems may occur. For example, some internet services will simply drop packets, resulting in lost data and distorted/lost images on the user's video screen. Other services will temporarily buffer (i.e., queue up) the additional packets and provide the packets to the client at the available data rate, resulting in an increase in latency – an unacceptable result for many video games and applications. Finally, some Internet service providers will view the increase in data rate as a malicious attack, such as a denial of service attack (a well known technique used by hackers to disable network connections), and will cut off the user's Internet connection for a

specified time period. Thus, the embodiments described herein take steps to ensure that the required data rate for a video game does not exceed the maximum available data rate.

HOSTING SERVICE ARCHITECTURE

Figure 4a illustrates an architecture of the hosting service 210 according to one embodiment. The hosting service 210 can either be located in a single server center, or can be distributed across a plurality of server centers (to provide for lower latency connections to users that have lower latency paths to certain server centers than others, to provide for load balancing amongst users, and to provide for redundancy in the case one or more server centers fail). The hosting service 210 may eventually include hundreds of thousands or even millions of servers 402, serving a very large user base. A hosting service control system 401 provides overall control for the hosting service 210, and directs routers, servers, video compression systems, billing and accounting systems, etc. In one embodiment, the hosting service control system 401 is implemented on a distributed processing Linux-based system tied to RAID arrays used to store the databases for user information, server information, and system statistics. In the foregoing descriptions, the various actions implemented by the hosting service 210, unless attributed to other specific systems, are initiated and controlled by the hosting service control system 401.

The hosting service 210 includes a number of servers 402 such as those currently available from Intel, IBM and Hewlett Packard, and others. Alternatively, the servers 402 can be assembled in a custom configuration of components, or can eventually be integrated so an entire server is implemented as a single chip. Although this diagram shows a small number of servers 402 for the sake of illustration, in an actual deployment there may be as few as one server 402

or as many as millions of servers 402 or more. The servers 402 may all be configured in the same way (as an example of some of the configuration parameters, with the same CPU type and performance; with or without a GPU, and if with a GPU, with the same GPU type and performance; with the same number of CPUs and GPUs; with the same amount of and type/speed of RAM; and with the same RAM configuration), or various subsets of the servers 402 may have the same configuration (e.g., 25% of the servers can be configured a certain way, 50% a different way, and 25% yet another way), or every server 402 may be different.

In one embodiment, the servers 402 are diskless, i.e., rather than having its own local mass storage (be it optical or magnetic storage, or semiconductor-based storage such as Flash memory or other mass storage means serving a similar function), each server accesses shared mass storage through fast backplane or network connection. In one embodiment, this fast connection is a Storage Area Network (SAN) 403 connected to a series of Redundant Arrays of Independent Disks (RAID) 405 with connections between devices implemented using Gigabit Ethernet. As is known by those of skill in the art, a SAN 403 may be used to combine many RAID arrays 405 together, resulting in extremely high bandwidth—approaching or potentially exceeding the bandwidth available from the RAM used in current gaming consoles and PCs. And, while RAID arrays based on rotating media, such as magnetic media, frequently have significant seek-time access latency, RAID arrays based on semiconductor storage can be implemented with much lower access latency. In another configuration, some or all of the servers 402 provide some or all of their own mass storage locally. For example, a server 402 may store frequently-accessed information such as its operating system and a copy of a video game or application on low-latency local Flash-based storage, but it may utilize the SAN to access RAID Arrays 405 based on rotating media with higher seek

latency to access large databases of geometry or game state information on a less frequent bases.

In addition, in one embodiment, the hosting service 210 employs low-latency video compression logic 404 described in detail below. The video compression logic 404 may be implemented in software, hardware, or any combination thereof (certain embodiments of which are described below). Video compression logic 404 includes logic for compressing audio as well as visual material.

In operation, while playing a video game or using an application at the user premises 211 via a keyboard, mouse, game controller or other input device 421, control signal logic 413 on the client 415 transmits control signals 406a-b (typically in the form of UDP packets) representing the button presses (and other types of user inputs) actuated by the user to the hosting service 210. The control signals from a given user are routed to the appropriate server (or servers, if multiple servers are responsive to the user's input device) 402. As illustrated in **Figure 4a**, control signals 406a may be routed to the servers 402 via the SAN. Alternatively or in addition, control signals 406b may be routed directly to the servers 402 over the hosting service network (e.g., an Ethernet-based local area network). Regardless of how they are transmitted, the server or servers execute the game or application software in response to the control signals 406a-b. Although not illustrated in **Figure 4a**, various networking components such as a firewall(s) and/or gateway(s) may process incoming and outgoing traffic at the edge of the hosting service 210 (e.g., between the hosting service 210 and the Internet 410) and/or at the edge of the user premises 211 between the Internet 410 and the home or office client 415. The graphical and audio output of the executed game or application software— i.e., new sequences of video images—are provided to the low-latency video compression

logic 404 which compresses the sequences of video images according to low-latency video compression techniques, such as those described herein and transmits a compressed video stream, typically with compressed or uncompressed audio, back to the client 415 over the Internet 410 (or, as described below, over an optimized high speed network service that bypasses the general Internet). Low-latency video decompression logic 412 on the client 415 then decompresses the video and audio streams and renders the decompressed video stream, and typically plays the decompressed audio stream, on a display device 422. Alternatively, the audio can be played on speakers separate from the display device 422 or not at all. Note that, despite the fact that input device 421 and display device 422 are shown as free-standing devices in Figures 2a and 2b, they may be integrated within client devices such as portable computers or mobile devices.

Home or office client 415 (described previously as home or office client 205 in Figures 2a and 2b) may be a very inexpensive and low-power device, with very limited computing or graphics performance and may well have very limited or no local mass storage. In contrast, each server 402, coupled to a SAN 403 and multiple RAIDs 405 can be an exceptionally high performance computing system, and indeed, if multiple servers are used cooperatively in a parallel-processing configuration, there is almost no limit to the amount of computing and graphics processing power that can be brought to bear. And, because of the low-latency video compression 404 and low-latency video decompression 412, perceptually to the user, the computing power of the servers 402 is being provided to the user. When the user presses a button on input device 421, the image on display 422 is updated in response to the button press perceptually with no meaningful delay, as if the game or application software were running locally. Thus, with a home or office client 415 that is a very low performance computer or just an inexpensive chip that

implements the low-latency video decompression and control signal logic 413, a user is provided with effectively arbitrary computing power from a remote location that appears to be available locally. This gives users the power to play the most advanced, processor-intensive (typically new) video games and the highest performance applications.

Figure 4c shows a very basic and inexpensive home or office client device 465. This device is an embodiment of home or office client 415 from Figures 4a and 4b. It is approximately 2 inches long. It has an Ethernet jack 462 that interfaces with an Ethernet cable with Power over Ethernet (PoE), from which it derives its power and its connectivity to the Internet. It is able to run Network Address Translation (NAT) within a network that supports NAT. In an office environment, many new Ethernet switches have PoE and bring PoE directly to a Ethernet jack in an office. In such a situation, all that is required is an Ethernet cable from the wall jack to the client 465. If the available Ethernet connection does not carry power (e.g., in a home with a DSL or cable modem, but no PoE), then there are inexpensive wall "bricks" (i.e., power supplies) available that will accept an unpowered Ethernet cable and output Ethernet with PoE.

The client 465 contains control signal logic 413 (of Figure 4a) that is coupled to a Bluetooth wireless interface, which interfaces with Bluetooth input devices 479, such as a keyboard, mouse, game controller and/or microphone and/or headset. Also, one embodiment of client 465 is capable of outputting video at 120fps coupled with a display device 468 able to support 120fps video and signal (typically through infrared) a pair of shuttered glasses 466 to alternately shutter one eye, then the other with each successive frame. The effect perceived by the user is that of a stereoscopic 3D image that "jumps out" of the display screen. One such display device 468 that supports such operation is the Samsung HL-T5076S. Since the

video stream for each eye is separate, in one embodiment two independent video streams are compressed by the hosting service 210, the frames are interleaved in time, and the frames are decompressed as two independent decompression processes within client 465.

The client 465 also contains low latency video decompression logic 412, which decompresses the incoming video and audio and output through the HDMI (High-Definition Multimedia Interface), connector 463 which plugs into an SDTV (Standard Definition Television) or HDTV (High Definition Television) 468, providing the TV with video and audio, or into a monitor 468 that supports HDMI. If the user's monitor 468 does not support HDMI, then an HDMI-to-DVI (Digital Visual Interface) can be used, but the audio will be lost. Under the HDMI standard, the display capabilities (e.g. supported resolutions, frame rates) 464 are communicated from the display device 468, and this information is then passed back through the Internet connection 462 back to the hosting service 210 so it can stream compressed video in a format suitable for the display device.

Figure 4d shows a home or office client device 475 that is the same as the home or office client device 465 shown in Figure 4c except that it has more external interfaces. Also, client 475 can accept either PoE for power, or it can run off of an external power supply adapter (not shown) that plugs in the wall. Using client 475 USB input, video camera 477 provides compressed video to client 475, which is uploaded by client 475 to hosting service 210 for use described below. Built into camera 477 is a low-latency compressor utilizing the compression techniques described below.

In addition to having an Ethernet connector for its Internet connection, client 475 also has an 802.11g wireless interface to the Internet. Both interfaces are able to use NAT within a network that supports NAT.

Also, in addition to having an HDMI connector to output video and audio, client 475 also has a Dual Link DVI-I connector, which includes analog output (and with a standard adapter cable will provide VGA output). It also has analog outputs for composite video and S-video.

For audio, the client 475 has left/right analog stereo RCA jacks, and for digital audio output it has a TOSLINK output.

In addition to a Bluetooth wireless interface to input devices 479, it also has USB jacks to interface to input devices.

Figure 4e shows one embodiment of the internal architecture of client 465. Either all or some of the devices shown in the diagram can be implemented in a Field Programmable Logic Array, a custom ASIC or in several discrete devices, either custom designed or off-the-shelf.

Ethernet with PoE 497 attaches to Ethernet Interface 481. Power 499 is derived from the Ethernet with PoE 497 and is connected to the rest of the devices in the client 465. Bus 480 is a common bus for communication between devices.

Control CPU 483 (almost any small CPU, such as a MIPS R4000 series CPU at 100MHz with embedded RAM is adequate) running a small client control application from Flash 476 implements the protocol stack for the network (i.e. Ethernet interface) and also communicates with the Hosting Service 210, and configures all of the devices in the client 465. It also handles interfaces with the input devices 469 and sends packets back to the hosting service 210 with user controller data, protected by Forward Error Correction, if necessary. Also, Control CPU 483 monitors the packet traffic (e.g. if packets are lost or delayed and also timestamps their arrival). This information is sent back to the hosting service 210 so that it can constantly monitor the network connection and adjust what it sends accordingly. Flash memory 476 is initially loaded at the time of manufacture with the control

program for Control CPU 483 and also with a serial number that is unique to the particular Client 465 unit. This serial number allows the hosting service 210 to uniquely identify the Client 465 unit.

Bluetooth interface 484 communicates to input devices 469 wirelessly through its antenna, internal to client 465.

Video decompressor 486 is a low-latency video decompressor configured to implement the video decompression described herein. A large number of video decompression devices exist, either off-the-shelf, or as Intellectual Property (IP) of a design that can be integrated into an FPGA or a custom ASIC. One company offering IP for an H.264 decoder is Ocean Logic of Manly, NSW Australia. The advantage of using IP is that the compression techniques used herein do not conform to compression standards. Some standard decompressors are flexible enough to be configured to accommodate the compression techniques herein, but some cannot. But, with IP, there is complete flexibility in redesigning the decompressor as needed.

The output of the video decompressor is coupled to the video output subsystem 487, which couples the video to the video output of the HDMI interface 490.

The audio decompression subsystem 488 is implemented either using a standard audio decompressor that is available, or it can be implemented as IP, or the audio decompression can be implemented within the control processor 483 which could, for example, implement the Vorbis audio decompressor (available at Vorbis.com).

The device that implements the audio decompression is coupled to the audio output subsystem 489 that couples the audio to the audio output of the HDMI interface 490

Figure 4f shows one embodiment of the internal architecture of client 475. As can be seen, the architecture is the same as that of client 465 except for additional interfaces and optional external DC power from a power supply adapter that plugs in the wall, and if so used, replaces power that would come from the Ethernet PoE 497. The functionality that is in common with client 465 will not be repeated below, but the additional functionality is described as follows.

CPU 483 communicates with and configures the additional devices.

WiFi subsystem 482 provides wireless Internet access as an alternative to Ethernet 497 through its antenna. WiFi subsystems are available from a wide range of manufacturers, including Atheros Communications of Santa Clara, CA.

USB subsystem 485 provides an alternative to Bluetooth communication for wired USB input devices 479. USB subsystems are quite standard and readily available for FPGAs and ASICs, as well as frequently built into off-the-shelf devices performing other functions, like video decompression.

Video output subsystem 487 produces a wider range of video outputs than within client 465. In addition to providing HDMI 490 video output, it provides DVI-I 491, S-video 492, and composite video 493. Also, when the DVI-I 491 interface is used for digital video, display capabilities 464 are passed back from the display device to the control CPU 483 so that it can notify the hosting service 210 of the display device 478 capabilities. All of the interfaces provided by the video output subsystem 487 are quite standard interfaces and readily available in many forms.

Audio output subsystem 489 outputs audio digitally through digital interface 494 (S/PDIF and/or Toslink) and audio in analog form through stereo analog interface 495.

ROUND-TRIP LATENCY ANALYSIS

Of course, for the benefits of the preceding paragraph to be realized, the round trip latency between a user's action using input device 421 and seeing the consequence of that action on display device 420 should be no more than 70-80ms. This latency must take into account all of the factors in the path from input device 421 in the user premises 211 to hosting service 210 and back again to the user premises 211 to display device 422. Figure 4b illustrates the various components and networks over which signals must travel, and above these components and networks is a timeline that lists exemplary latencies that can be expected in a practical implementation. Note that Figure 4b is simplified so that only the critical path routing is shown. Other routing of data used for other features of the system is described below. Double-headed arrows (e.g., arrow 453) indicate round-trip latency and a single-headed arrows (e.g., arrow 457) indicate one-way latency, and "~" denotes an approximate measure. It should be pointed out that there will be real-world situations where the latencies listed cannot be achieved, but in a large number of cases in the US, using DSL and cable modem connections to the user premises 211, these latencies can be achieved in the circumstances described in the next paragraph. Also, note that, while cellular wireless connectivity to the Internet will certainly work in the system shown, most current US cellular data systems (such as EVDO) incur very high latencies and would not be able to achieve the latencies shown in Figure 4b. However, these underlying principles may be implemented on future cellular technologies that may be capable of implementing this level of latency. Further, there are game and application scenarios (e.g., games that do not require fast user reaction time, such as chess) where the latency incurred through a current US cellular data system, while noticeable to the user, would be acceptable for the game or application.

Starting from the input device 421 at user premises 211, once the user actuates the input device 421, a user control signal is sent to client 415 (which may be a standalone device such a set-top box, or it may be software or hardware running in another device such as a PC or a mobile device), and is packetized (in UDP format in one embodiment) and the packet is given a destination address to reach hosting service 210. The packet will also contain information to indicate which user the control signals are coming from. The control signal packet(s) are then forwarded through Firewall/Router/NAT (Network Address Translation) device 443 to WAN interface 442. WAN interface 442 is the interface device provided to the user premises 211 by the User's ISP (Internet Service Provider). The WAN interface 442 may be a Cable or DSL modem, a WiMax transceiver, a Fiber transceiver, a Cellular data interface, an Internet Protocol-over-powerline interface, or any other of many interfaces to the Internet. Further, Firewall/Router/NAT device 443 (and potentially WAN interface 442) may be integrated into the client 415. An example of this would be a mobile phone, which includes software to implement the functionality of home or office client 415, as well as the means to route and connect to the Internet wirelessly through some standard (e.g., 802.11g).

WAN Interface 442 then routes the control signals to what shall be called herein the "point of presence" 441 for the user's Internet Service Provider (ISP) which is the facility that provides an interface between the WAN transport connected to the user premises 211 and the general Internet or private networks. The point of presence's characteristics will vary depending upon nature of the Internet service provided. For DSL, it typically will be a telephone company Central Office where a DSLAM is located. For cable modems, it typically will be a cable Multi-System Operator (MSO) head end. For cellular systems, it typically will be a control room associated with cellular tower. But whatever the point of presence's

nature, it will then route the control signal packet(s) to the general Internet 410. The control signal packet(s) will then be routed to the WAN Interface 441 to the hosting service 210, through what most likely will be a fiber transceiver interface. The WAN 441 will then route the control signal packets to routing logic 409 (which may be implemented in many different ways, including Ethernet switches and routing servers), which evaluates the user's address and routes the control signal(s) to the correct server 402 for the given user.

The server 402 then takes the control signals as input for the game or application software that is running on the server 402 and uses the control signals to process the next frame of the game or application. Once the next frame is generated, the video and audio is output from server 402 to video compressor 404. The video and audio may be output from server 402 to compressor 404 through various means. To start with, compressor 404 may be built into server 402, so the compression may be implemented locally within server 402. Or, the video and/or audio may be output in packetized form through a network connection such as an Ethernet connection to a network that is either a private network between server 402 and video compressor 404, or a through a shared network, such as SAN 403. Or, the video may be output through a video output connector from server 402, such as a DVI or VGA connector, and then captured by video compressor 404. Also, the audio may be output from server 402 as either digital audio (e.g., through a TOSLINK or S/PDIF connector) or as analog audio, which is digitized and encoded by audio compression logic within video compressor 404.

Once video compressor 404 has captured the video frame and the audio generated during that frame time from server 402, the video compressor will compress the video and audio using techniques described below. Once the video and audio is compressed it is packetized with an address to send it back to the

user's client 415, and it is routed to the WAN Interface 441, which then routes the video and audio packets through the general Internet 410, which then routes the video and audio packets to the user's ISP point of presence 441, which routes the video and audio packets to the WAN Interface 442 at the user's premises, which routes the video and audio packets to the Firewall/Router/NAT device 443, which then routes the video and audio packets to the client 415.

The client 415 decompresses the video and audio, and then displays the video on the display device 422 (or the client's built-in display device) and sends the audio to the display device 422 or to separate amplifier/speakers or to an amplifier/speakers built in the client.

For the user to perceive that the entire process just described is perceptually without lag, the round-trip delay needs be less than 70 or 80ms. Some of the latency delays in the described round-trip path are under the control of the hosting service 210 and/or the user and others are not. Nonetheless, based on analysis and testing of a large number of real-world scenarios, the following are approximate measurements.

The one-way transmission time to send the control signals 451 is typically less than 1ms, the roundtrip routing through the user premises 452 is typically accomplished, using readily available consumer-grade Firewall/Router/NAT switches over Ethernet in about 1ms. User ISPs vary widely in their round trip delays 453, but with DSL and cable modem providers, we typically see between 10 and 25ms. The round trip latency on the general Internet 410 can vary greatly depending on how traffic is routed and whether there are any failures on the route (and these issues are discussed below), but typically the general Internet provides fairly optimal routes and the latency is largely determined by speed of light through optical fiber, given the distance to the destination. As discussed further below, we have

established 1000 miles as a roughly the furthest distance that we expect to place a hosting service 210 away from user premises 211. At 1000 miles (2000 miles round trip) the practical transit time for a signal through the Internet is approximately 22ms. The WAN Interface 441 to the hosting service 210 is typically a commercial-grade fiber high speed interface with negligible latency. Thus, the general Internet latency 454 is typically between 1 and 10ms. The one-way routing 455 latency through the hosting service 210 can be achieved in less than 1ms. The server 402 will typically compute a new frame for a game or an application in less than one frame time (which at 60fps is 16.7ms) so 16ms is a reasonable maximum one-way latency 456 to use. In an optimized hardware implementation of the video compression and audio compression algorithms described herein, the compression 457 can be completed in 1ms. In less optimized versions, the compression may take as much as 6ms (of course even less optimized versions could take longer, but such implementations would impact the overall latency of the round trip and would require other latencies to be shorter (e.g., the allowable distance through the general Internet could be reduced) to maintain the 70-80ms latency target). The round trip latencies of the Internet 454, User ISP 453, and User Premises Routing 452 have already been considered, so what remains is the video decompression 458 latency which, depending on whether the video decompression 458 is implemented in dedicated hardware, or if implemented in software on a client device 415 (such as a PC or mobile device) it can vary depending upon the size of the display and the performance of the decompressing CPU. Typically, decompression 458 takes between 1 and 8ms.

Thus, by adding together all of the worst-case latencies seen in practice, we can determine the worst-case round trip latency that can be expected to be experienced by a user of the system shown in Figure 4a. They are:

$1+1+25+22+1+16+6+8 = 80\text{ms}$. And, indeed, in practice (with caveats discussed below), this is roughly the round trip latency seen using prototype versions of the system shown in Figure 4a, using off-the-shelf Windows PCs as client devices and home DSL and cable modem connections within the US. Of course, scenarios better than worst case can result in much shorter latencies, but they cannot be relied upon in developing a commercial service that is used widely.

To achieve the latencies listed in Figures 4b over the general Internet requires the video compressor 404 and video decompressor 412 from Figure 4a in the client 415 to generate a packet stream which very particular characteristics, such that the packet sequence generated through entire path from the hosting service 210 to the display device 422 is not subject to delays or excessive packet loss and, in particular, consistently falls with the constraints of the bandwidth available to the user over the user's Internet connection through WAN interface 442 and Firewall/Router/NAT 443. Further, the video compressor must create a packet stream which is sufficiently robust so that it can tolerate the inevitable packet loss and packet reordering that occurs in normal Internet and network transmissions.

LOW-LATENCY VIDEO COMPRESSION

To accomplish the foregoing goals, one embodiment takes a new approach to video compression which decreases the latency and the peak bandwidth requirements for transmitting video. Prior to the description of these embodiments, an analysis of current video compression techniques will be provided with respect to Figure 5 and Figures 6a-b. Of course, these techniques may be employed in accordance with underlying principles if the user is provided with sufficient bandwidth to handle the data rate required by these techniques. Note that audio compression is not addressed herein other than to state that it is implemented

simultaneously and in synchrony with the video compression. Prior art audio compression techniques exist that satisfy the requirements for this system.

Figure 5 illustrates one particular prior art technique for compressing video in which each individual video frame 501-503 is compressed by compression logic 520 using a particular compression algorithm to generate a series of compressed frames 511-513. One embodiment of this technique is "motion JPEG" in which each frame is compressed according to a Joint Pictures Expert Group (JPEG) compression algorithm, based upon the discrete cosine transform (DCT). Various different types of compression algorithms may be employed, however, while still complying with these underlying principles (e.g., wavelet-based compression algorithms such as JPEG-2000).

One problem with this type of compression is that it reduces the data rate of each frame, but it does not exploit similarities between successive frames to reduce the data rate of the overall video stream. For example, as illustrated in Figure 5, assuming a frame rate of $640 \times 480 \times 24 \text{ bits/pixel} = 640 \times 480 \times 24 / 8 / 1024 = 900$ Kilobytes/frame (KB/frame), for a given quality of image, motion JPEG may only compress the stream by a factor of 10, resulting in a data stream of 90 KB/frame. At 60 frames/sec, this would require a channel bandwidth of $90 \text{ KB} \times 8 \text{ bits} \times 60 \text{ frames/sec} = 42.2 \text{ Mbps}$, which would be far too high bandwidth for almost all home Internet connections in the US today, and too high bandwidth for many office Internet connections. Indeed, given that it would demand a constant data stream at such a high bandwidth, and it would be just serving one user, even in an office LAN environment, it would consume a large percentage of a 100Mbps Ethernet LAN's bandwidth and heavily burden Ethernet switches supporting the LAN. Thus, the compression for motion video is inefficient when compared with other compression techniques (such as those described below). Moreover, single frame compression

algorithms like JPEG and JPEG-2000 that use lossy compression algorithms produce compression artifacts that may not be noticeable in still images (e.g., an artifact within dense foliage in the scene may not appear as an artifact since the eye does not know exactly how the dense foliage should appear). But, once the scene is in motion, an artifact can stand out because the eye detects that the artifact changed from frame-to-frame, despite the fact the artifact is in an area of the scene where it might not have been noticeable in a still image. This results in the perception of “background noise” in the sequence of frames, similar in appearance to the “snow” noise visible during marginal analog TV reception. Of course, this type of compression may still be used in certain embodiments described herein, but generally speaking, to avoid background noise in the scene, a high data rate (i.e., a low compression ratio) is required for a given perceptual quality.

Other types of compression, such as H.264, or Windows Media VC9, MPEG2 and MPEG4 are all more efficient at compressing a video stream because they exploit the similarities between successive frames. These techniques all rely upon the same general techniques to compress video. Thus, although the H.264 standard will be described, the same general principles apply to various other compression algorithms. A large number of H.264 compressors and decompressor are available, including the x264 open source software library for compressing H.264 and the FFmpeg open source software libraries for decompressing H.264.

Figures 6a and 6b illustrate an exemplary prior art compression technique in which a series of uncompressed video frames 501-503, 559-561 are compressed by compression logic 620 into a series of “I frames” 611, 671; “P frames” 612-613; and “B frames” 670. The vertical axis in Figure 6a generally signifies the resulting size of each of the encoded frames (although the frames are not drawn to scale). As described above, video coding using I frames, B frames and

P frames is well understood by those of skill in the art. Briefly, an I frame 611 is a DCT-based compression of a complete uncompressed frame 501 (similar to a compressed JPEG image as described above). P frames 612-613 generally are significantly smaller in size than I frames 611 because they take advantage of the data in the previous I frame or P frame; that is, they contain data indicating the changes between the previous I frame or P frame. B frames 670 are similar to that of P frames except that B frames use the frame in the following reference frame as well as potentially the frame in the preceding reference frame.

For the following discussion, it will be assumed that the desired frame rate is 60 frames/second, that each I frame is approximately 160 Kb, the average P frame and B frame is 16 Kb and that a new I frame is generated every second. With this set of parameters, the average data rate would be: $160 \text{ Kb} + 16 \text{ Kb} * 59 = 1.1 \text{ Mbps}$. This data rate falls well within the maximum data rate for many current broadband Internet connections to homes and offices. This technique also tends to avoid the background noise problem from intraframe-only encoding because the P and B frames track differences between the frames, so compression artifacts tend not to appear and disappear from frame-to-frame, thereby reducing the background noise problem described above.

One problem with the foregoing types of compression is that although the average data rate is relatively low (e.g., 1.1Mbps), a single I frame may take several frame times to transmit. For example, using prior art techniques a 2.2 Mbps network connection (e.g., DSL or cable modem with 2.2Mbps peak of max available data rate 302 from Figure 3a) would typically be adequate to stream video at 1.1 Mbps with a 160Kbps I frame each 60 frames. This would be accomplished by having the decompressor queue up 1 second of video before decompressing the video. In 1 second, 1.1Mb of data would be transmitted, which would be easily

accommodated by a 2.2Mbps max available data rate, even assuming that the available data rate might dip periodically by as much as 50%. Unfortunately, this prior art approach would result in a 1-second latency for the video because of the 1-second video buffer at the receiver. Such a delay is adequate for many prior art applications (e.g., the playback of linear video), but is far too long a latency for fast action video games which cannot tolerate more than 70-80ms of latency.

If an attempt were made to eliminate the 1-second video buffer, it still would not result in an adequate reduction in latency for fast action video games. For one, the use of B frames, as previously described, would necessitate the reception of all of the B frames preceding an I frame as well as the I frame. If we assume the 59 non-I frames are roughly split between P and B frames, then there would be at least 29 B frames and an I frame received before any B frame could be displayed. Thus, regardless of the available bandwidth of the channel, it would necessitate a delay of $29+1=30$ frames of $1/60^{\text{th}}$ second duration each, or 500ms of latency. Clearly that is far too long.

Thus, another approach would be to eliminate B frames and only use I and P frames. (One consequence of this is the data rate would increase for a given quality level, but for the sake of consistency in this example, let's continue to assume that each I frame is 160Kb and the average P frame is 16Kb in size, and thus the data rate is still 1.1Mbps) This approach eliminates the unavoidable latency introduced by B frames, since the decoding of each P frame is only reliant upon the prior received frame. A problem that remains with this approach is that an I frame is so much larger than an average P frame, that on a low bandwidth channel, as is typical in most homes and in many offices, the transmission of the I frame adds substantial latency. This is illustrated in Figure 6b. The video stream data rate 624 is below the available max data rate 621 except for the I frames, where the peak

data rate required for the I frames 623 far exceeds the available max data rate 622 (and even the rated max data rate 621). The data rate required by the P frames is less than the available max data rate. Even if the available max data rate peaks at 2.2Mbps remains steadily at its 2.2Mbps peak rate, it will take $160\text{Kb}/2.2\text{Mb}=71\text{ms}$ to transmit the I frame, and if the available max data rate 622 dips by 50% (1.1Mbps), it will take 142ms to transmit the I frame. So, the latency in transmitting the I frame will fall somewhere in between 71-142ms. This latency is additive to the latencies identified in Figure 4b, which in the worst case added up to 70 ms, so this would result in a total round trip latency of 141-222ms from the point the user actuates input device 421 until an image appears on display device 422, which is far too high. And if the available max data rate dips below 2.2Mbps, the latency will increase further.

Note also that there generally are severe consequences to “jamming” an ISP with peak data rate 623 that are far in excess of the available data rate 622. The equipment in different ISPs will behave differently, but the following behaviors are quite common among DSL and cable modem ISPs when receiving packets at much higher data rate than the available data rate 622: (a) delaying the packets by queuing them (introducing latency), (b) dropping some or all of the packets, (c) disabling the connection for a period of time (most likely because the ISP is concerned it is a malicious attack, such as “denial of service” attack). Thus, transmitting a packet stream at full data rate with characteristics such as those shown in Figure 6b is not a viable option. The peaks 623 may be queued up at the hosting service 210 and sent at a data rate below the available max data rate, introducing the unacceptable latency described in the preceding paragraph.

Further, the video stream data rate sequence 624 shown in Figure 6b is a very “tame” video stream data rate sequence and would be the sort of data rate

sequence that one would expect to result from compressing the video from a video sequence that does not change very much and has very little motion (e.g., as would be common in video teleconferencing where the cameras are in a fixed position and have little motion, and the objects, in the scene, e.g., seated people talking, show little motion).

The video stream data rate sequence 634 shown in Figure 6c is a sequence typical to what one would expect to see from video with far more action, such as might be generated in a motion picture or a video game, or in some application software. Note that in addition to the I frame peaks 633, there are also P frame peaks such as 635 and 636 that are quite large and exceed the available max data rate on many occasions. Although these P frame peaks are not quite as large as the I frame peaks, they still are far too large to be carried by the channel at full data rate, and as with the I frame peaks, they P frame peaks must be transmitted slowly (thereby increasing latency).

On a high bandwidth channel (e.g., a 100Mbps LAN, or a high bandwidth 100Mbps private connection) the network would be able to tolerate large peaks, such as I frame peaks 633 or P frame peaks 636, and in principle, low latency could be maintained. But, such networks are frequently shared amongst many users (e.g., in an office environment), and such “peaky” data would impact the performance of the LAN, particularly if the network traffic was routed to a private shared connection (e.g., from a remote data center to an office). To start with, bear in mind that this example is of a relatively low resolution video stream of 640x480 pixels at 60fps. HDTV streams of 1920x1080 at 60fps are readily handled by modern computers and displays, and 2560x1440 resolution displays at 60fps are increasingly available (e.g., Apple, Inc.’s 30” display). A high action video sequence at 1920x1080 at 60fps may require 4.5 Mbps using H.264 compression for a

reasonable quality level. If we assume the I frames peak at 10X the nominal data rate, that would result in 45Mbps peaks, as well as smaller, but still considerable, P frame peak. If several users were receiving video streams on the same 100Mbps network (e.g., a private network connection between an office and data center), it is easy to see how the peaks from several users' video stream could happen to align, overwhelming the bandwidth of the network, and potentially overwhelming the bandwidth of the backplanes of the switches supporting the users on the network. Even in the case of a Gigabit Ethernet network, if enough users had enough peaks aligned at once, it could overwhelm the network or the network switches. And, once 2560x1440 resolution video becomes more commonplace, the average video stream data rate may be 9.5Mbps, resulting in perhaps a 95Mbps peak data rate. Needless to say, a 100Mbps connection between a data center and an office (which today is an exceptionally fast connection) would be completely swamped by the peak traffic from a single user. Thus, even though LANs and private network connections can be more tolerant of peaky streaming video, the streaming video with high peaks is not desirable and might require special planning and accommodation by an office's IT department.

Of course, for standard linear video applications these issues are not a problem because the data rate is "smoothed" at the point of transmission and the data for each frame below the max available data rate 622, and a buffer in the client stores a sequence of I, P and B frames before they are decompressed. Thus, the data rate over the network remains close to the average data rate of the video stream. Unfortunately, this introduces latency, even if B frames are not used, that is unacceptable for low-latency applications such as video games and applications require fast response time.

One prior art solution to mitigating video streams that have high peaks is to use a technique often referred to as "Constant Bit Rate" (CBR) encoding. Although the term CBR would seem to imply that all frames are compressed to have the same bit rate (i.e., size), what it usually refers to is a compression paradigm where a maximum bit rate across a certain number of frames (in our case, 1 frame) is allowed. For example, in the case of Figure 6c, if a CBR constraint were applied to the encoding that limited the bit rate to, for example, 70% of the rated max data rate 621, then the compression algorithm would limit the compression of each of the frames so that any frame that would normally be compressed using more than 70% of the rated max data rate 621 would be compressed with less bits. The result of this is that frames that would normally require more bits to maintain a given quality level would be "starved" of bits and the image quality of those frames would be worse than that of other frames that do not require more bits than the 70% of the rate max data rate 621. This approach can produce acceptable results for certain types of compressed video where there (a) little motion or scene changes are expected and (b) the users can accept periodic quality degradation. A good example of a CBR-suited application is video teleconferencing since there are few peaks, and if the quality degrades briefly (for example, if the camera is panned, resulting in significant scene motion and large peaks, during the panning there may not be enough bits for high-quality image compression, which would result in degraded image quality), it is acceptable for most users. Unfortunately, CBR is not well-suited for many other applications which have scenes of high complexity or a great deal of motion and/or where a reasonably constant level of quality is required.

The low-latency compression logic 404 employed in one embodiment uses several different techniques to address the range of problems with streaming low-latency compressed video, while maintaining high quality. First, the low-latency

compression logic 404 generates only I frames and P frames, thereby alleviating the need to wait several frame times to decode each B frame. In addition, as illustrated in Figure 7a, in one embodiment, the low-latency compression logic 404 subdivides each uncompressed frame 701-760 into a series of “tiles” and individually encodes each tile as either an I frame or a P frame. The group of compressed I frames and P frames are referred to herein as “R frames” 711-770. In the specific example shown in Figure 7a, each uncompressed frame is subdivided into a 4 x 4 matrix of 16 tiles. However, these underlying principles are not limited to any particular subdivision scheme.

In one embodiment, the low-latency compression logic 404 divides up the video frame into a number of tiles, and encodes (i.e., compresses) one tile from each frame as an I frame (i.e., the tile is compressed as if it is a separate video frame of $1/16^{\text{th}}$ the size of the full image, and the compression used for this “mini” frame is I frame compression) and the remaining tiles as P frames (i.e., the compression used for each “mini” $1/16^{\text{th}}$ frame is P frame compression). Tiles compressed as I frames and as P frames shall be referred to as “I tiles” and “P tiles”, respectively. With each successive video frame, the tile to be encoded as an I tile is changed. Thus, in a given frame time, only one tile of the tiles in the video frame is an I tile, and the remainder of the tiles are P tiles. For example, in Figure 7a, tile 0 of uncompressed frame 701 is encoded as I tile I_0 and the remaining 1-15 tiles are encoded as P tiles P_1 through P_{15} to produce R frame 711. In the next uncompressed video frame 702, tile 1 of uncompressed frame 701 is encoded as I tile I_1 and the remaining tiles 0 and 2 through 15 are encoded as P tiles, P_0 and P_2 through P_{15} , to produce R frame 712. Thus, the I tiles and P tiles for tiles are progressively interleaved in time over successive frames. The process continues until a R tile 770 is generated with the last tile in the matrix encoded as an I tile (i.e.,

I₁₅). The process then starts over, generating another R frame such as frame 711 (i.e., encoding an I tile for tile 0) etc. Although not illustrated in Figure 7a, in one embodiment, the first R frame of the video sequence of R frames contains only I tiles (i.e., so that subsequent P frames have reference image data from which to calculate motion). Alternatively, in one embodiment, the startup sequence uses the same I tile pattern as normal, but does not include P tiles for those tiles that have not yet been encoded with an I tile. In other words, certain tiles are not encoded with any data until the first I tile arrives, thereby avoiding startup peaks in the video stream data rate 934 in **Figure 9a**, which is explained in further detail below. Moreover, as described below, various different sizes and shapes may be used for the tiles while still complying with these underlying principles.

The video decompression logic 412 running on the client 415 decompresses each tile as if it is a separate video sequence of small I and P frames, and then renders each tile to the frame buffer driving display device 422. For example, I₀ and P₀ from R frames 711 to 770 are used to decompress and render tile 0 of the video image. Similarly, I₁ and P₁ from R frames 711 to 770 are used to reconstruct tile 1, and so on. As mentioned above, decompression of I frames and P frames is well known in the art, and decompression of I tiles and P tiles can be accomplished by having a multiple instances of a video decompressor running in the client 415. Although multiplying processes would seem to increase the computational burden on client 415, it actually doesn't because the tile themselves are proportionally smaller relative to the number of additional processes, so the number of pixels displayed is the same as if there were one process and using conventional full sized I and P frames.

This R frame technique significantly mitigates the bandwidth peaks typically associated with I frames illustrated in Figures 6b and 6c because any given

frame is mostly made up of P frames which are typically smaller than I frames. For example, assuming again that a typical I frame is 160Kb, then the I tiles of each of the frames illustrated in Figure 7a would be roughly 1/16 of this amount or 10Kb. Similarly, assuming that a typical P frame is 16 Kb, then the P frames for each of the tiles illustrated in Figure 7a may be roughly 1Kb. The end result is an R frame of approximately $10\text{Kb} + 15 * 1\text{Kb} = 25\text{Kb}$. So, each 60-frame sequence would be $25\text{Kb} * 60 = 1.5\text{Mbps}$. So, at 60 frames/second, this would require a channel capable of sustaining a bandwidth of 1.5Mbps, but with much lower peaks due to I tiles being distributed throughout the 60-frame interval.

Note that in previous examples with the same assumed data rates for I frames and P frames, the average data rate was 1.1Mbps. This is because in the previous examples, a new I frame was only introduced once every 60 frame times, whereas in this example, the 16 tiles that make up an I frame cycle through in 16 frames times, and as such the equivalent of an I frame is introduced every 16 frame times, resulting in a slightly higher average data rate. In practice, though, introducing more frequent I frames does not increase the data rate linearly. This is due to the fact that a P frame (or a P tile) primarily encodes the difference from the prior frame to the next. So, if the prior frame is quite similar to the next frame, the P frame will be very small, if the prior frame is quite different from the next frame, the P frame will be very large. But because a P frame is largely derived from the previous frame, rather than from the actual frame, the resulting encoded frame may contain more errors (e.g., visual artifacts) than an I frame with an adequate number of bits. And, when one P frame follows another P frame, what can occur is an accumulation of errors that gets worse when there is a long sequence of P frames. Now, a sophisticated video compressor will detect the fact that the quality of the image is degrading after a sequence of P frames and, if necessary, it will allocate more bits to subsequent P

frames to bring up the quality or, if it is the most efficient course of action, replace a P frame with an I frame. So, when long sequences of P frames are used (e.g., 59 P frames, as in prior examples above) particularly when the scene has a great deal of complexity and/or motion, typically, more bits are needed for P frames as they get further removed from an I frame.

Or, to look at P frames from the opposite point of view, P frames that closely follow an I frame tend to require less bits than P frames that are further removed from an I frame. So, in the example shown in Figure 7a, no P frame is further than 15 frames removed from an I frame that precedes it, where as in the prior example, a P frame could be 59 frames removed from an I frame. Thus, with more frequent I frames, the P frames are smaller. Of course, the exact relative sizes will vary based on the nature of the video stream, but in the example of Figure 7a, if an I tile is 10Kb, P tiles on average, may be only 0.75kb in size resulting in $10\text{Kb} + 15 * 0.75\text{Kb} = 21.25\text{Kb}$, or at 60 frames per second, the data rate would be $21.25\text{Kb} * 60 = 1.3\text{Mbps}$, or about 16% higher data rate than a stream with an I frame followed by 59 P frames at 1.1Mbps. Once, again, the relative results between these two approaches to video compression will vary depending up on the video sequence, but typically, we have found empirically that using R-frames require about 20% more bits for a given level of quality than using I /P frame sequences. But, of course, R frames dramatically reduce the peaks which make the video sequences usable with far less latency than I/P frame sequences.

R frames can be configured in a variety of different ways, depending upon the nature of the video sequence, the reliability of the channel, and the available data rate. In an alternative embodiment, a different number of tiles is used than 16 in a 4x4 configuration. For example 2 tiles may be used in a 2x1 or 1x2 configuration, 4 tiles may be used in a 2x2, 4x1 or 1x4 configuration, 6 tiles may be

used in a 3x2, 2x3, 6x1 or 1x6 configurations or 8 tiles may be used in a 4x2 (as shown in Figure 7b), 2x4, 8x1 or 1x8 configuration. Note that the tiles need not be square, nor must the video frame be square, or even rectangular. The tiles can be broken up into whatever shape best suits the video stream and the application used.

In another embodiment, the cycling of the I and P tiles is not locked to the number of tiles. For example, in an 8-tile 4x2 configuration, a 16-cycle sequence can still be used as illustrated in Figure 7b. Sequential uncompressed frames 721, 722, 723 are each divided into 8 tiles, 0-7 and each tile is compressed individually. From R frame 731, only tile 0 is compressed as an I tile, and the remaining tiles are compressed as P tiles. For subsequent R frame 732 all of the 8 tiles are compressed as P tiles, and then for subsequent R frame 733, tile 1 is compressed as an I tile and the other tiles are all compressed as P tiles. And, so the sequencing continues for 16 frames, with an I tile generated only every other frame, so the last I tile is generated for tile 7 during the 15th frame time (not shown in Figure 7b) and during the 16th frame time R frame 780 is compressed using all P tiles. Then, the sequence begins again with tile 0 compressed as an I tile and the other tiles compressed as P tiles. As in the prior embodiment, the very first frame of the entire video sequence would typically be all I tiles, to provide a reference for P tiles from that point forward. The cycling of I tiles and P tiles need not even be an even multiple of the number of tiles. For example, with 8 tiles, each frame with an I tile can be followed by 2 frames with all P tiles, before another I tile is used. In yet another embodiment, certain tiles may be sequenced with I tiles more often than other tiles if, for example, certain areas of the screen are known to have more motion requiring from frequent I tiles, while others are more static (e.g., showing a score for a game) requiring less frequent I tiles. Moreover, although each frame is illustrated in Figures 7a-b with a single I tile, multiple I tiles may be encoded in a single frame (depending on the bandwidth of the

transmission channel). Conversely, certain frames or frame sequences may be transmitted with no I tiles (i.e., only P tiles).

The reason the approaches of the preceding paragraph works well is that while not having I tiles distributed across every single frame would seem to be result in larger peaks, the behavior of the system is not that simple. Since each tile is compressed separately from the other tiles, as the tiles get smaller the encoding of each tile can become less efficient, because the compressor of a given tile is not able to exploit similar image features and similar motion from the other tiles. Thus, dividing up the screen into 16 tiles generally will result in a less efficient encoding than dividing up the screen into 8 tiles. But, if the screen is divided into 8 tiles and it causes the data of a full I frame to be introduced every 8 frames instead of every 16 frames, it results in a much higher data rate overall. So, by introducing a full I frame every 16 frames instead of every 8 frames, the overall data rate is reduced. Also, by using 8 larger tiles instead of 16 smaller tiles, the overall data rate is reduced, which also mitigates to some degree the data peaks caused by the larger tiles.

In another embodiment, the low-latency video compression logic 404 in Figures 7a and 7b controls the allocation of bits to the various tiles in the R frames either by being pre-configured by settings, based on known characteristics of the video sequence to be compressed, or automatically, based upon an ongoing analysis of the image quality in each tile. For example, in some racing video games, the front of the player's car (which is relatively motionless in the scene) takes up a large part of the lower half of the screen, whereas the upper half of the screen is entirely filled with the oncoming roadway, buildings and scenery, which is almost always in motion. If the compression logic 404 allocates an equal number of bits to each tile, then the tiles on the bottom half of the screen (tiles 4-7) in uncompressed frame 721 in Figure 7b, will generally be compressed with higher quality than tiles

than the tiles in the upper half of the screen (tiles 0-3) in uncompressed frame 721 in Figure 7b. If this particular game, or this particular scene of the game is known to have such characteristics, then the operators of the hosting service 210 can configure the compression logic 404 to allocate more bits to the tiles in the top of the screen than to tiles at the bottom of the screen. Or, the compression logic 404 can evaluate the quality of the compression of the tiles after frames are compressed (using one or more of many compression quality metrics, such as Peak Signal-To-Noise Ratio (PSNR)) and if it determines that over a certain window of time, certain tiles are consistently producing better quality results, then it gradually allocates more bits to tiles that are producing lower quality results, until the various tiles reach a similar level of quality. In an alternative embodiment, the compressor logic 404 allocates bits to achieve higher quality in a particular tile or group of tiles. For example, it may provide a better overall perceptual appearance to have higher quality in the center of the screen than at the edges.

In one embodiment, to improve resolution of certain regions of the video stream, the video compression logic 404 uses smaller tiles to encode areas of the video stream with relatively more scene complexity and/or motion than areas of the video stream with relatively less scene complexity and/or motion. For example, as illustrated in Figure 8, smaller tiles are employed around a moving character 805 in one area of one R frame 811 (potentially followed by a series of R frames with the same tile sizes (not shown)). Then, when the character 805 moves to a new area of the image, smaller tiles are used around this new area within another R frame 812, as illustrated. As mentioned above, various different sizes and shapes may be employed as “tiles” while still complying with these underlying principles.

While the cyclic I/P tiles described above substantially reduce the peaks in the data rate of a video stream, they do not eliminate the peaks entirely,

particularly in the case of rapidly-changing or highly complex video imagery, such as occurs with motion pictures, video games, and some application software. For example, during a sudden scene transition, a complex frame may be followed by another complex frame that is completely different. Even though several I tiles may have preceded the scene transition by only a few frame times, they don't help in this situation because the new frame's material has no relation to the previous I tiles. In such a situation (and in other situations where even though not everything changes, much of the image changes), the video compressor 404 will determine that many, if not all, of the P tiles are more efficiently coded as I tiles, and what results is a very large peak in the data rate for that frame.

As discussed previously, it is simply the case that with most consumer-grade Internet connections (and many office connections), it simply is not feasible to "jam" data that exceeds the available maximum data rate shown as 622 in Figure 6c, along with the rated maximum data rate 621. Note that the rated maximum data rate 621 (e.g., "6Mbps DSL") is essentially a marketing number for users considering the purchase of an Internet connection, but generally it does not guarantee a level of performance. For the purposes of this application, it is irrelevant, since our only concern is the available maximum data rate 622 at the time the video is streamed through the connection. Consequently, in Figures 9a and 9c, as we describe a solution to the peaking problem, the rated maximum data rate is omitted from the graph, and only the available maximum data rate 922 is shown. The video stream data rate must not exceed the available maximum data rate 922.

To address this, the first thing that the video compressor 404 does is determine a peak data rate 941, which is a data rate the channel is able to handle steadily. This rate can be determined by a number of techniques. One such technique is by gradually sending an increasingly higher data rate test stream from

the hosting service 210 to the client 415 in Figures 4a and 4b, and having the client provide feedback to the hosting service as to the level of packet loss and latency. As the packet loss and/or latency begins to show a sharp increase, that is an indication that the available maximum data rate 922 is being reached. After that, the hosting service 210 can gradually reduce the data rate of the test stream until the client 415 reports that for a reasonable period of time the test stream has been received with an acceptable level of packet loss and the latency is near minimal. This establishes a peak maximum data rate 941, which will then be used as a peak data rate for streaming video. Over time, the peak data rate 941 will fluctuate (e.g., if another user in a household starts to heavily use the Internet connection), and the client 415 will need to constantly monitor it to see whether packet loss or latency increases, indicating the available max data rate 922 is dropping below the previously established peak data rate 941, and if so the peak data rate 941. Similarly, if over time the client 415 finds that the packet loss and latency remain at optimal levels, it can request that the video compressor slowly increases the data rate to see whether the available maximum data rate has increased (e.g., if another user in a household has stopped heavy use of the Internet connection), and again waiting until packet loss and/or higher latency indicates that the available maximum data rate 922 has been exceeded, and again a lower level can be found for the peak data rate 941, but one that is perhaps higher than the level before testing an increased data rate. So, by using this technique (and other techniques like it) a peak data rate 941 can be found, and adjusted periodically as needed. The peak data rate 941 establishes the maximum data rate that can be used by the video compressor 404 to stream video to the user. The logic for determining the peak data rate may be implemented at the user premises 211 and/or on the hosting service 210. At the user premises 211, the client device 415 performs the calculations to determine the peak data rate and

transmits this information back to the hosting service 210; at the hosting service 210, a server 402 at the hosting service performs the calculations to determine the peak data rate based on statistics received from the client 415 (e.g., packet loss, latency, max data rate, etc).

Figure 9a shows an example video stream data rate 934 that has substantial scene complexity and/or motion that has been generated using the cyclic I/P tile compression techniques described previously and illustrated in Figures 7a, 7b and 8. The video compressor 404 has been configured to output compressed video at an average data rate that is below the peak data rate 941, and note that, most of the time, the video stream data rate remains below the peak data rate 941. A comparison of data rate 934 with video stream data rate 634 shown in Figure 6c created using I/P/B or I/P frames shows that the cyclic I/P tile compression produces a much smoother data rate. Still, at frame 2x peak 952 (which approaches 2x the peak data rate 942) and frame 4x peak 954 (which approaches 4x the peak data rate 944), the data rate exceeds the peak data rate 941, which is unacceptable. In practice, even with high action video from rapidly changing video games, peaks in excess of peak data rate 941 occur in less than 2% of frames, peaks in excess of 2x peak data rate 942 occur rarely, and peaks in excess of 3x peak data rate 943 occur hardly ever. But, when they do occur (e.g., during a scene transition), the data rate required by them is necessary to produce a good quality video image.

One way to solve this problem is simply to configure the video compressor 404 such that its maximum data rate output is the peak data rate 941. Unfortunately, the resulting video output quality during the peak frames is poor since the compression algorithm is "starved" for bits. What results is the appearance of compression artifacts when there are sudden transitions or fast motion, and in time,

the user comes to realize that the artifacts always crop up when there is sudden changes or rapid motion, and they can become quite annoying.

Although the human visual system is quite sensitive to visual artifacts that appear during sudden changes or rapid motion, it is not very sensitive to detecting a reduction in frame rate in such situations. In fact, when such sudden changes occur, it appears that the human visual system is preoccupied with tracking the changes, and it doesn't notice if the frame rate briefly drops from 60fps to 30fps, and then returns immediately to 60fps. And, in the case of a very dramatic transition, like a sudden scene change, the human visual system doesn't notice if the frame rate drops to 20fps or even 15fps, and then immediately returns to 60fps. So long as the frame rate reduction only occurs infrequently, to a human observer, it appears that the video has been continuously running at 60fps.

This property of the human visual system is exploited by the techniques illustrated in Figure 9b. A server 402 (from Figures 4a and 4b) produces an uncompressed video output stream at a steady frame rate (at 60fps in one embodiment). A timeline shows each frame 961-970 output each $1/60^{\text{th}}$ second. Each uncompressed video frame, starting with frame 961, is output to the low-latency video compressor 404, which compresses the frame in less than a frame time, producing for the first frame compressed frame 1 981. The data produced for the compressed frame 1 981 may be larger or smaller, depending upon many factors, as previously described. If the data is small enough that it can be transmitted to the client 415 in a frame time ($1/60^{\text{th}}$ second) or less at the peak data rate 941, then it is transmitted during transmit time (xmit time) 991 (the length of the arrow indicates the duration of the transmit time). In the next frame time, server 402 produces uncompressed frame 2 962, it is compressed to compressed frame 2 982,

and it is transmitted to client 415 during transmit time 992, which is less than a frame time at peak data rate 941.

Then, in the next frame time, server 402 produces uncompressed frame 3 963. When it is compressed by video compressor 404, the resulting compressed frame 3 983 is more data than can be transmitted at the peak data rate 941 in one frame time. So, it is transmitted during transmit time (2x peak) 993, which takes up all of the frame time and part of the next frame time. Now, during the next frame time, server 402 produces another uncompressed frame 4 964 and outputs it to video compressor 404 but the data is ignored and illustrated with 974. This is because video compressor 404 is configured to ignore further uncompressed video frames that arrive while it is still transmitting a prior compressed frame. Of course client 415's video decompressor will fail to receive frame 4, but it simply continues to display on display device 422 frame 3 for 2 frame times (i.e., briefly reduces the frame rate from 60fps to 30fps).

For the next frame 5, server 402 outputs uncompressed frame 5 965, is compressed to compressed frame 5 985 and transmitted within 1 frame during transmit time 995. Client 415's video decompressor decompresses frame 5 and displays it on display device 422. Next, server 402 outputs uncompressed frame 6 966, video compressor 404 compresses it to compressed frame 6 986, but this time the resulting data is very large. The compressed frame is transmitted during transmit time (4x peak) 996 at the peak data rate 941, but it takes almost 4 frame times to transmit the frame. During the next 3 frame times, video compressor 404 ignores 3 frames from server 402, and client 415's decompressor holds frame 6 steadily on the display device 422 for 4 frames times (i.e., briefly reduces the frame rate from 60fps to 15fps). Then finally, server 402 outputs frame 10 970, video compressor 404 compresses it into compressed frame 10 987, and it is transmitted during

transmit time 997, and client 415's decompressor decompresses frame 10 and displays it on display device 422 and once again the video resumes at 60fps.

Note that although video compressor 404 drops video frames from the video stream generated by server 402, it does not drop audio data, regardless of what form the audio comes in, and it continues to compress the audio data when video frames are dropped and transmit them to client 415, which continues to decompress the audio data and provide the audio to whatever device is used by the user to playback the audio. Thus audio continues unabated during periods when frames are dropped. Compressed audio consumes a relatively small percentage of bandwidth, compared to compressed video, and as result does not have a major impact on the overall data rate. Although it is not illustrated in any of the data rate diagrams, there is always data rate capacity reserved for the compressed audio stream within the peak data rate 941.

The example just described in Figure 9b was chosen to illustrate how the frame rate drops during data rate peaks, but what it does not illustrate is that when the cyclic I/P tile techniques described previously are used, such data rate peaks, and the consequential dropped frames are rare, even during high scene complexity/high action sequences such as those that occur in video games, motion pictures and some application software. Consequently, the reduced frame rates are infrequent and brief, and the human visual system does not detect them.

If the frame rate reduction mechanism just described is applied to the video stream data rate illustrated in Figure 9a, the resulting video stream data rate is illustrated in Figure 9c. In this example, 2x peak 952 has been reduced to flattened 2x peak 953, and 4x peak 955 has been reduced to flattened 4x peak 955, and the entire video stream data rate 934 remains at or below the peak data rate 941.

Thus, using the techniques described above, a high action video stream can be transmitted with low latency through the general Internet and through a consumer-grade Internet connection. Further, in an office environment on a LAN (e.g., 100Mbps Ethernet or 802.11g wireless) or on a private network (e.g., 100Mbps connection between a data center and offices) a high action video stream can be transmitted without peaks so that multiple users (e.g., transmitting 1920x1080 at 60fps at 4.5Mbps) can use the LAN or shared private data connection without having overlapping peaks overwhelming the network or the network switch backplanes.

DATA RATE ADJUSTMENT

In one embodiment, the hosting service 210 initially assesses the available maximum data rate 622 and latency of the channel to determine an appropriate data rate for the video stream and then dynamically adjusts the data rate in response. To adjust the data rate, the hosting service 210 may, for example, modify the image resolution and/or the number of frames/second of the video stream to be sent to the client 415. Also, the hosting service can adjust the quality level of the compressed video. When changing the resolution of the video stream, e.g., from a 1280 x 720 resolution to a 640 x 360 the video decompression logic 412 on the client 415 can scale up the image to maintain the same image size on the display screen.

In one embodiment, in a situation where the channel completely drops out, the hosting service 210 pauses the game. In the case of a multiplayer game, the hosting service reports to the other users that the user has dropped out of the game and/or pauses the game for the other users.

DROPPED OR DELAYED PACKETS

In one embodiment, if data is lost due to packet loss between the video compressor 404 and client 415 in Figures 4a or 4b, or due to a packet being

received out of order that arrives too late to decompress and meet the latency requirements of the decompressed frame, the video decompression logic 412 is able to mitigate the visual artifacts. In a streaming I/P frame implementation, if there is a lost/delayed packet, the entire screen is impacted, potentially causing the screen to completely freeze for a period of time or show other screen-wide visual artifacts. For example, if a lost/delayed packet causes the loss of an I frame, then the decompressor will lack a reference for all of the P frames that follow until a new I frame is received. If a P frame is lost, then it will impact the P frames for the entire screen that follow. Depending on how long it will be before an I frame appears, this will have a longer or shorter visual impact. Using interleaved I/P tiles as shown in Figures 7a and 7b, a lost/delayed packet is much less likely to impact the entire screen since it will only affect the tiles contained in the affected packet. If each tile's data is sent within an individual packet, then if a packet is lost, it will only affect one tile. Of course, the duration of the visual artifact will depend on whether an I tile packet is lost and, if a P tile is lost, how many frames it will take until an I tile appears. But, given that different tiles on the screen are being updated with I frames very frequently (potentially every frame), even if one tile on the screen is affected, other tiles may not be. Further, if some event cause a loss of several packets at once (e.g., a spike in power next to a DSL line that briefly disrupts the data flow), then some of the tiles will be affected more than others, but because some tiles will quickly be renewed with a new I tile, they will be only briefly affected. Also, with a streaming I/P frame implementation, not only are the I frames the most critical frame, but the I frames are extremely large, so if there is an event that causes a dropped/delayed packet, there is a higher probability that an I frame will be affected (i.e., if any part of an I frame is lost, it is unlikely that the I frame can be decompressed at all) than a much smaller I tile. For all of these reasons, using I/P

tiles results in far fewer visual artifacts when packets are dropped/delayed than with I/P frames.

One embodiment attempts to reduce the effect of lost packets by intelligently packaging the compressed tiles within the TCP (transmission control protocol) packets or UDP (user datagram protocol) packets. For example, in one embodiment, tiles are aligned with packet boundaries whenever possible. Figure 10a illustrates how tiles might be packed within a series of packets 1001-1005 without implementing this feature. Specifically, in Figure 10a, tiles cross packet boundaries and are packed inefficiently so that the loss of a single packet results in the loss of multiple frames. For example, if packets 1003 or 1004 are lost, three tiles are lost, resulting in visual artifacts.

By contrast, Figure 10b illustrates tile packing logic 1010 for intelligently packing tiles within packets to reduce the effect of packet loss. First, the tile packing logic 1010 aligns tiles with packet boundaries. Thus, tiles T1, T3, T4, T7, and T2 are aligned with the boundaries of packets 1001-1005, respectively. The tile packing logic also attempts to fit tiles within packets in the most efficient manner possible, without crossing packet boundaries. Based on the size of each of the tiles, tiles T1 and T6 are combined in one packet 1001; T3 and T5 are combined in one packet 1002; tiles T4 and T8 are combined in one packet 1003; tile T8 is added to packet 1004; and tile T2 is added to packet 1005. Thus, under this scheme, a single packet loss will result in the loss of no more than 2 tiles (rather than 3 tiles as illustrated in Figure 10a).

One additional benefit to the embodiment shown in Figure 10b is that the tiles are transmitted in a different order in which they are displayed within the image. This way, if adjacent packets are lost from the same event interfering with

the transmission it will affect areas which are not near each other on the screen, creating a less noticeable artifacting on the display.

One embodiment employs forward error correction (FEC) techniques to protect certain portions of the video stream from channel errors. As is known in the art, FEC techniques such as Reed-Solomon and Viterbi generate and append error correction data information to data transmitted over a communications channel. If an error occurs in the underlying data (e.g., an I frame), then the FEC may be used to correct the error.

FEC codes increase the data rate of the transmission, so ideally, they are only used where they are most needed. If data is being sent that would not result in a very noticeable visual artifact, it may be preferable to not use FEC codes to protect the data. For example, a P tile that immediately precedes an I tile that is lost will only create a visual artifact (i.e., on tile on the screen will not be updated) for $1/60^{\text{th}}$ of second on the screen. Such a visual artifact is barely detectable by the human eye. As P tiles are further back from an I tile, losing a P tile becomes increasingly more noticeable. For example, if a tile cycle pattern is an I tile followed by 15 P tiles before an I tile is available again, then if the P tile immediately following an I tile is lost, it will result in that tile showing an incorrect image for 15 frame times (at 60 fps, that would be 250ms). The human eye will readily detect a disruption in a stream for 250ms. So, the further back a P tile is from a new I tile (i.e., the closer a P tiles follows an I tile), the more noticeable the artifact. As previously discussed, though, in general, the closer a P tile follows an I tile, the smaller the data for that P tile. Thus, P tiles following I tiles not only are more critical to protect from being lost, but they are smaller in size. And, in general, the smaller the data is that needs to be protected, the smaller the FEC code needs to be to protect it.

So, as illustrated in Figure 11a, in one embodiment, because of the importance of I tiles in the video stream, only I tiles are provided with FEC codes. Thus, FEC 1101 contains error correction code for I tile 1100 and FEC 1104 contains error correction code for I tile 1103. In this embodiment, no FEC is generated for the P tiles.

In one embodiment illustrated in Figure 11b FEC codes are also generated for P tiles which are most likely to cause visual artifacts if lost. In this embodiment, FECs 1105 provide error correction codes for the first 3 P tiles, but not for the P tiles that follow. In another embodiment, FEC codes are generated for P tiles which are smallest in data size (which will tend to self-select P tiles occurring the soonest after an I tile, which are the most critical to protect).

In another embodiment, rather than sending an FEC code with a tile, the tile is transmitted twice, each time in a different packet. If one packet is lost/delayed, the other packet is used.

In one embodiment, shown in Figure 11c, FEC codes 1111 and 1113 are generated for audio packets, 1110 and 1112, respectively, transmitted from the hosting service concurrently with the video. It is particularly important to maintain the integrity of the audio in a video stream because distorted audio (e.g., clicking or hissing) will result in a particularly undesirable user experience. The FEC codes help to ensure that the audio content is rendered at the client computer 415 without distortion.

In another embodiment, rather than sending an FEC code with audio data, the audio data is transmitted twice, each time in a different packet. If one packet is lost/delayed, the other packet is used.

In addition, in one embodiment illustrated in Figure 11d, FEC codes 1121 and 1123 are used for user input commands 1120 and 1122, respectively (e.g.,

button presses) transmitted upstream from the client 415 to the hosting service 210. This is important because missing a button press or a mouse movement in a video game or an application could result in an undesirable user experience.

In another embodiment, rather than sending an FEC code with user input command data, the user input command data is transmitted twice, each time in a different packet. If one packet is lost/delayed, the other packet is used.

In one embodiment, the hosting service 210 assesses the quality of the communication channel with the client 415 to determine whether to use FEC and, if so, what portions of the video, audio and user commands to which FEC should be applied. Assessing the "quality" of the channel may include functions such as evaluating packet loss, latency, etc, as described above. If the channel is particularly unreliable, then the hosting service 210 may apply FEC to all of I tiles, P tiles, audio and user commands. By contrast, if the channel is reliable, then the hosting service 210 may apply FEC only to audio and user commands, or may not apply FEC to audio or video, or may not use FEC at all. Various other permutations of the application of FEC may be employed while still complying with these underlying principles. In one embodiment, the hosting service 210 continually monitors the conditions of the channel and changes the FEC policy accordingly.

In another embodiment, referring to Figures 4a and 4b, when a packet is lost/delayed resulting in the loss of tile data or if, perhaps because of a particularly bad packet loss, the FEC is unable to correct lost tile data, the client 415 assesses how many frames are left before a new I tile will be received and compares it to the round-trip latency from the client 415 to hosting service 210. If the round-trip latency is less than the number of frames before a new I tile is due to arrive, then the client 415 sends a message to the hosting service 210 requesting a new I tile. This message is routed to the video compressor 404, and rather than generating a P tile

for the tile whose data had been lost, it generates an I tile. Given that the system shown in Figs. 4a and 4b is designed to provide a round-trip latency that is typically less than 80ms, this results in a tile being corrected within 80ms (at 60fps, frames are 16.67ms of duration, thus in full frame times, 80ms latency would result in a corrected a tile within 83.33ms, which is 5 frame times—a noticeable disruption, but far less noticeable than, for example, a 250ms disruption for 15 frames). When the compressor 404 generates such an I tile out of its usual cyclic order, if the I tile would cause the bandwidth of that frame to exceed the available bandwidth, then the compressor 404 will delay the cycles of the other tiles so that the other tiles receive P tiles during that frame time (even if one tile would normally be due an I tile during that frame), and then starting with the next frame the usual cycling will continue, and the tile that normally would have received an I tile in the preceding frame will receive an I tile. Although this action briefly delays the phase of the R frame cycling, it normally will not be noticeable visually.

VIDEO AND AUDIO COMPRESSOR/DECOMPRESSOR IMPLEMENTATION

Figure 12 illustrates one particular embodiment in which a multi-core and/or multi-processor 1200 is used to compress 8 tiles in parallel. In one embodiment, a dual processor, quad core Xeon CPU computer system running at 2.66 GHz or higher is used, with each core implementing the open source x264 H.264 compressor as an independent process. However, various other hardware/software configurations may be used while still complying with these underlying principles. For example, each of the CPU cores can be replaced with an H.264 compressor implemented in an FPGA. In the example shown in Figure 12, cores 1201-1208 are used to concurrently process the I tiles and P tiles as eight independent threads. As is well known in the art, current multi-core and multi-processor computer systems are inherently capable of multi-threading when

integrated with multi-threading operating systems such as Microsoft Windows XP Professional Edition (either 64-bit or the 32-bit edition) and Linux.

In the embodiment illustrated in Figure 12, since each of the 8 cores is responsible for just one tile, it operates largely independently from the other cores, each running a separate instantiation of x264. A PCI Express x1-based DVI capture card, such as the Sendero Video Imaging IP Development Board from Microtronix of Oosterhout, The Netherlands is used to capture uncompressed video at 640x480, 800x600, or 1280x720 resolution, and the FPGA on the card uses Direct Memory Access (DMA) to transfer the captured video through the DVI bus into system RAM. The tiles are arranged in a 4x2 arrangement 1205 (although they are illustrated as square tiles, in this embodiment they are of 160x240 resolution). Each instantiation of x264 is configured to compress one of the 8 160x240 tiles, and they are synchronized such that, after an initial I tile compression, each core enters into a cycle, each one frame out of phase with the other, to compress one I tile followed by seven P tiles, and illustrated in Figure 12.

Each frame time, the resulting compressed tiles are combined into a packet stream, using the techniques previously described, and then the compressed tiles are transmitted to a destination client 415.

Although not illustrated in Figure 12, if the data rate of the combined 8 tiles exceeds a specified peak data rate 941, then all 8 x264 processes are suspended for as many frame times as are necessary until the data for the combined 8 tiles has been transmitted.

In one embodiment, client 415 is implemented as software on a PC running 8 instantiations of FFmpeg. A receiving process receives the 8 tiles, and each tile is routed to an FFmpeg instantiation, which decompresses the tile and renders it to an appropriate tile location on the display device 422.

The client 415 receives keyboard, mouse, or game controller input from the PC's input device drivers and transmits it to the server 402. The server 402 then applies the received input device data and applies it to the game or application running on the server 402, which is a PC running Windows using an Intel 2.16GHz Core Duo CPU. The server 402 then produces a new frame and outputs it through its DVI output, either from a motherboard-based graphics system, or through a NVIDIA 8800GTX PCI Express card's DVI output.

Simultaneously, the server 402 outputs the audio produced by game or applications through its digital audio output (e.g., S/PDIF), which is coupled to the digital audio input on the dual quad-core Xeon-based PC that is implementing the video compression. A Vorbis open source audio compressor is used to compress the audio simultaneously with the video using whatever core is available for the process thread. In one embodiment, the core that completes compressing its tile first executes the audio compression. The compressed audio is then transmitted along with the compressed video, and is decompressed on the client 415 using a Vorbis audio decompressor.

HOSTING SERVICE SERVER CENTER DISTRIBUTION

Light through glass, such as optical fiber, travels at some fraction of the speed of light in a vacuum, and so an exact propagation speed for light in optical fiber could be determined. But, in practice, allowing time for routing delays, transmission inefficiencies, and other overhead, we have observed that optimal latencies on the Internet reflect transmission speeds closer to 50% the speed of light. Thus, an optimal 1000 mile round trip latency is approximately 22ms, and an optimal 3000 mile round trip latency is about 64ms. Thus, a single server on one US coast will be too far away to serve clients on the other coast (which can be as far as 3000 miles away) with the desired latency. However, as illustrated in Figure 13a, if the

hosting service 210 server center 1300 is located in the center of the US (e.g., Kansas, Nebraska, etc.), such that the distance to any point in the continental US is approximately 1500 miles or less, the round trip Internet latency could be as low as 32 ms. Referring to Figure 4b, note that although the worst-case latencies allowed for the user ISP 453 is 25ms, typically, we have observed latencies closer to 10-15ms with DSL and cable modem systems. Also, Figure 4b assumes a maximum distance from the user premises 211 to the hosting center 210 of 1000 miles. Thus, with a typical user ISP round trip latency of 15ms used and a maximum Internet distance of 1500 miles for a round trip latency of 32ms, the total round trip latency from the point a user actuates input device 421 and sees a response on display device 422 is $1+1+15+32+1+16+6+8 = 80\text{ms}$. So, the 80ms response time can be typically achieved over an Internet distance of 1500 miles. This would allow any user premises with a short enough user ISP latency 453 in the continental US to access a single server center that is centrally located.

In another embodiment, illustrated in Figure 13b, the hosting service 210 server centers, HS1-HS6, are strategically positioned around the United States (or other geographical region), with certain larger hosting service server centers positioned close to high population centers (e.g., HS2 and HS5). In one embodiment, the server centers HS1-HS6 exchange information via a network 1301 which may be the Internet or a private network or a combination of both. With multiple server centers, services can be provided at lower latency to users that have high user ISP latency 453.

Although distance on the Internet is certainly a factor that contributes to round trip latency through the Internet, sometimes other factors come into play that are largely unrelated to latency. Sometimes a packet stream is routed through the Internet to a far away location and back again, resulting in latency from the long

loop. Sometimes there is routing equipment on the path that is not operating properly, resulting in a delay of the transmission. Sometimes there is a traffic overloading a path which introduces delay. And, sometimes, there is a failure that prevents the user's ISP from routing to a given destination at all. Thus, while the general Internet usually provides connections from one point to another with a fairly reliable and optimal route and latency that is largely determined by distance (especially with long distance connections that result in routing outside of the user's local area) such reliability and latency is by no means guaranteed and often cannot be achieved from a user's premises to a given destination on the general Internet.

In one embodiment, when a user client 415 initially connects to the hosting service 210 to play a video game or use an application, the client communicates with each of the hosting service server centers HS1-HS6 available upon startup (e.g., using the techniques described above). If the latency is low enough for a particular connection, then that connection is used. In one embodiment, the client communicates with all, or a subset, of the hosting service server centers and the one with the lowest latency connection is selected. The client may select the service center with the lowest latency connection or the service centers may identify the one with the lowest latency connection and provide this information (e.g., in the form of an Internet address) to the client.

If a particular hosting service server center is overloaded and/or the user's game or application can tolerate the latency to another, less loaded hosting service server center, then the client 415 may be redirected to the other hosting service server center. In such a situation, the game or application the user is running would be paused on the server 402 at the user's overloaded server center, and the game or application state data would be transferred to a server 402 at another hosting service server center. The game or application would then be resumed. In

one embodiment, the hosting service 210 would wait until the game or application has either reached a natural pausing point (e.g., between levels in a game, or after the user initiates a "save" operation in application) to do the transfer. In yet another embodiment, the hosting service 210 would wait until user activity ceases for a specified period of time (e.g., 1 minute) and then would initiate the transfer at that time.

As described above, in one embodiment, the hosting service 210 subscribes to an Internet bypass service 440 of Figure 14 to attempt to provide guaranteed latency to its clients. Internet bypass services, as used herein, are services that provide private network routes from one point to another on the Internet with guaranteed characteristics (e.g., latency, data rate, etc.). For example, if the hosting service 210 was receiving large amount of traffic from users using AT&T's DSL service offering in San Francisco, rather than routing to AT&T's San Francisco-based central offices, the hosting service 210 could lease a high-capacity private data connection from a service provider (perhaps AT&T itself or another provider) between the San Francisco-based central offices and one or more of the server centers for hosting service 210. Then, if routes from all hosting service server centers HS1-HS6 through the general Internet to a user in San Francisco using AT&T DSL result in too high latency, then private data connection could be used instead. Although private data connections are generally more expensive than the routes through the general Internet, so long as they remain a small percentage of the hosting service 210 connections to users, the overall cost impact will be low, and users will experience a more consistent service experience.

Server centers often have two layers of backup power in the event of power failure. The first layer typically is backup power from batteries (or from an alternative immediately available energy source, such a flywheel that is kept running

and is attached to a generator), which provides power immediately when the power mains fail and keeps the server center running. If the power failure is brief, and the power mains return quickly (e.g., within a minute), then the batteries are all that is needed to keep the server center running. But if the power failure is for a longer period of time, then typically generators (e.g., diesel-powered) are started up that take over for the batteries and can run for as long as they have fuel. Such generators are extremely expensive since they must be capable of producing as much power as the server center normally gets from the power mains.

In one embodiment, each of the hosting services HS1-HS5 share user data with one another so that if one server center has a power failure, it can pause the games and applications that are in process, and then transfer the game or application state data from each server 402 to servers 402 at other server centers, and then will notify the client 415 of each user to direct its communications to the new server 402. Given that such situations occur infrequently, it may be acceptable to transfer a user to a hosting service server center which is not able to provide optimal latency (i.e., the user will simply have to tolerate higher latency for the duration of the power failure), which will allow for a much wider range of options for transferring users. For example, given the time zone differences across the US, users on the East Coast may be going to sleep at 11:30PM while users on the West Coast at 8:30PM are starting to peak in video game usage. If there is a power failure in a hosting service server center on the West Coast at that time, there may not be enough West Coast servers 402 at other hosting service server centers to handle all of the users. In such a situation, some of the users can be transferred to hosting service server centers on the East Coast which have available servers 402, and the only consequence to the users would be higher latency. Once the users have been transferred from the server center that has lost power, the server center can then

commence an orderly shutdown of its servers and equipment, such that all of the equipment has been shut down before the batteries (or other immediate power backup) is exhausted. In this way, the cost of a generator for the server center can be avoided.

In one embodiment, during times of heavy loading of the hosting service 210 (either due to peak user loading, or because one or more server centers have failed) users are transferred to other server centers on the basis of the latency requirements of the game or application they are using. So, users using games or applications that require low latency would be given preference to available low latency server connections when there is a limited supply.

HOSTING SERVICE FEATURES

Figure 15 illustrates an embodiment of components of a server center for hosting service 210 utilized in the following feature descriptions. As with the hosting service 210 illustrated in Figure 2a, the components of this server center are controlled and coordinated by a hosting service 210 control system 401 unless otherwise qualified.

Inbound internet traffic 1501 from user clients 415 is directed to inbound routing 1502. Typically, inbound internet traffic 1501 will enter the server center via a high-speed fiber optic connection to the Internet, but any network connection means of adequate bandwidth, reliability and low latency will suffice. Inbound routing 1502 is a system of network (the network can be implemented as an Ethernet network, a fiber channel network, or through any other transport means) switches and routing servers supporting the switches which takes the arriving packets and routes each packet to the appropriate application/game ("app/game") server 1521-1525. In one embodiment, a packet which is delivered to a particular app/game server represents a subset of the data received from the client and/or may

be translated/changed by other components (e.g., networking components such as gateways and routers) within the data center. In some cases, packets will be routed to more than one server 1521-1525 at a time, for example, if a game or application is running on multiple servers at once in parallel. RAID arrays 1511-1512 are connected to the inbound routing network 1502, such that the app/game servers 1521-1525 can read and write to the RAID arrays 1511-1512. Further, a RAID array 1515 (which may be implemented as multiple RAID arrays) is also connected to the inbound routing 1502 and data from RAID array 1515 can be read from app/game servers 1521-1525. The inbound routing 1502 may be implemented in a wide range of prior art network architectures, including a tree structure of switches, with the inbound internet traffic 1501 at its root; in a mesh structure interconnecting all of the various devices; or as an interconnected series of subnets, with concentrated traffic amongst intercommunicating device segregated from concentrated traffic amongst other devices. One type of network configuration is a SAN which, although typically used for storage devices, it can also be used for general high-speed data transfer among devices. Also, the app/game servers 1521-1525 may each have multiple network connections to the inbound routing 1502. For example, a server 1521-1525 may have a network connection to a subnet attached to RAID Arrays 1511-1512 and another network connection to a subnet attached to other devices.

The app/game servers 1521-1525 may all be configured the same, some differently, or all differently, as previously described in relation to servers 402 in the embodiment illustrated in Figure 4a. In one embodiment, each user, when using the hosting service is typically using at least one app/game server 1521-1525. For the sake of simplicity of explanation, we shall assume a given user is using app/game server 1521, but multiple servers could be used by one user, and multiple users could share a single app/game server 1521-1525. The user's control input,

sent from client 415 as previously described is received as inbound Internet traffic 1501, and is routed through inbound routing 1502 to app/game server 1521.

App/game server 1521 uses the user's control input as control input to the game or application running on the server, and computes the next frame of video and the audio associated with it. App/game server 1521 then outputs the uncompressed video/audio 1529 to shared video compression 1530. App/game server may output the uncompressed video via any means, including one or more Gigabit Ethernet connections, but in one embodiment the video is output via a DVI connection and the audio and other compression and communication channel state information is output via a Universal Serial Bus (USB) connection.

The shared video compression 1530 compresses the uncompressed video and audio from the app/game servers 1521-1525. The compression maybe implemented entirely in hardware, or in hardware running software. There may a dedicated compressor for each app/game server 1521-1525, or if the compressors are fast enough, a given compressor can be used to compress the video/audio from more than one app/game server 1521-1525. For example, at 60fps a video frame time is 16.67ms. If a compressor is able to compress a frame in 1ms, then that compressor could be used to compress the video/audio from as many as 16 app/game servers 1521-1525 by taking input from one server after another, with the compressor saving the state of each video/audio compression process and switching context as it cycles amongst the video/audio streams from the servers. This results in substantial cost savings in compression hardware. Since different servers will be completing frames at different times, in one embodiment, the compressor resources are in a shared pool 1530 with shared storage means (e.g., RAM, Flash) for storing the state of each compression process, and when a server 1521-1525 frame is complete and ready to be compressed, a control means

determines which compression resource is available at that time, provides the compression resource with the state of the server's compression process and the frame of uncompressed video/audio to compress.

Note that part of the state for each server's compression process includes information about the compression itself, such as the previous frame's decompressed frame buffer data which may be used as a reference for P tiles, the resolution of the video output; the quality of the compression; the tiling structure; the allocation of bits per tiles; the compression quality, the audio format (e.g., stereo, surround sound, Dolby® AC-3). But the compression process state also includes communication channel state information regarding the peak data rate 941 and whether a previous frame (as illustrated in Fig 9b) is currently being output (and as result the current frame should be ignored), and potentially whether there are channel characteristics which should be considered in the compression, such as excessive packet loss, which affect decisions for the compression (e.g., in terms of the frequency of I tiles, etc). As the peak data rate 941 or other channel characteristics change over time, as determined by an app/game server 1521-1525 supporting each user monitoring data sent from the client 415, the app/game server 1521-1525 sends the relevant information to the shared hardware compression 1530.

The shared hardware compression 1530 also packetizes the compressed video/audio using means such as those previously described, and if appropriate, applying FEC codes, duplicating certain data, or taking other steps to as to adequately ensure the ability of the video/audio data stream to be received by the client 415 and decompressed with as high a quality and reliability as feasible.

Some applications, such as those described below, require the video/audio output of a given app/game server 1521-1525 to be available at multiple resolutions (or in other multiple formats) simultaneously. If the app/game server

1521-1525 so notifies the shared hardware compression 1530 resource, then the uncompressed video audio 1529 of that app/game server 1521-1525 will be simultaneously compressed in different formats, different resolutions, and/or in different packet/error correction structures. In some cases, some compression resources can be shared amongst multiple compression processes compressing the same video/audio (e.g., in many compression algorithms, there is a step whereby the image is scaled to multiple sizes before applying compression. If different size images are required to be output, then this step can be used to serve several compression processes at once). In other cases, separate compression resources will be required for each format. In any case, the compressed video/audio 1539 of all of the various resolutions and formats required for a given app/game server 1521-1525 (be it one or many) will be output at once to outbound routing 1540. In one embodiment the output of the compressed video/audio 1539 is in UDP format, so it is a unidirectional stream of packets.

The outbound routing network 1540 comprises a series of routing servers and switches which direct each compressed video/audio stream to the intended user(s) or other destinations through outbound Internet traffic 1599 interface (which typically would connect to a fiber interface to the Internet) and/or back to the delay buffer 1515, and/or back to the inbound routing 1502, and/or out through a private network (not shown) for video distribution. Note that (as described below) the outbound routing 1540 may output a given video/audio stream to multiple destinations at once. In one embodiment this is implemented using Internet Protocol (IP) multicast in which a given UDP stream intended to be streamed to multiple destinations at once is broadcasted, and the broadcast is repeated by the routing servers and switches in the outbound routing 1540. The multiple destinations of the broadcast may be to multiple users' clients 415 via the Internet, to multiple

app/game servers 1521-1525 via inbound routing 1502, and/or to one or more delay buffers 1515. Thus, the output of a given server 1521-1522 is compressed into one or multiple formats, and each compressed stream is directed to one or multiple destinations.

Further, in another embodiment, if multiple app/game servers 1521-1525 are used simultaneously by one user (e.g., in a parallel processing configuration to create the 3D output of a complex scene) and each server is producing part of the resulting image, the video output of multiple servers 1521-1525 can be combined by the shared hardware compression 1530 into a combined frame, and from that point forward it is handled as described above as if it came from a single app/game server 1521-1525.

Note that in one embodiment, a copy (in at least the resolution or higher of video viewed by the user) of all video generated by app/game servers 1521-1525 is recorded in delay buffer 1515 for at least some number of minutes (15 minutes in one embodiment). This allows each user to "rewind" the video from each session in order to review previous work or exploits (in the case of a game). Thus, in one embodiment, each compressed video/audio output 1539 stream being routed to a user client 415 is also being multicasted to a delay buffer 1515. When the video/audio is stored on a delay buffer 1515, a directory on the delay buffer 1515 provides a cross reference between the network address of the app/game server 1521-1525 that is the source of the delayed video/audio and the location on the delay buffer 1515 where the delayed video/audio can be found.

LIVE, INSTANTLY-VIEWABLE, INSTANTLY-PLAYABLE GAMES

App/game servers 1521-1525 may not only be used for running a given application or video game for a user, but they may also be used for creating the user interface applications for the hosting service 210 that supports navigation

through hosting service 210 and other features. A screen shot of one such user interface application is shown in Figure 16, a “Game Finder” screen. This particular user interface screen allows a user to watch 15 games that are being played live (or delayed) by other users. Each of the “thumbnail” video windows, such as 1600 is a live video window in motion showing the video from one user’s game. The view shown in the thumbnail may be the same view that the user is seeing, or it may be a delayed view (e.g., if a user is playing a combat game, a user may not want other users to see where she is hiding and she may choose to delay any view of her gameplay by a period of time, say 10 minutes). The view may also be a camera view of a game that is different from any user’s view. Through menu selections (not shown in this illustration), a user may choose a selection of games to view at once, based on a variety of criteria. As a small sampling of exemplary choices, the user may select a random selection of games (such as those shown in Figure 16), all of one kind of games (all being played by different players), only the top-ranked players of a game, players at a given level in the game, or lower-ranked players (e.g., if the player is learning the basics), players who are “buddies” (or are rivals), games that have the most number of viewers, etc.

Note that generally, each user will decide whether the video from his or her game or application can be viewed by others and, if so, which others, and when it may be viewed by others, whether it is only viewable with a delay.

The app/game server 1521-1525 that is generating the user interface screen shown in Figure 16 acquires the 15 video/audio feeds by sending a message to the app/game server 1521-1525 for each user whose game it is requesting from. The message is sent through the inbound routing 1502 or another network. The message will include the size and format of the video/audio requested, and will identify the user viewing the user interface screen. A given user may choose to

select "privacy" mode and not permit any other users to view video/audio of his game (either from his point of view or from another point of view), or as described in the previous paragraph, a user may choose to allow viewing of video/audio from her game, but delay the video/audio viewed. A user app/game server 1521-1525 receiving and accepting a request to allow its video/audio to be viewed will acknowledge as such to the requesting server, and it will also notify the shared hardware compression 1530 of the need to generate an additional compressed video stream in the requested format or screen size (assuming the format and screen size is different than one already being generated), and it will also indicate the destination for the compressed video (i.e., the requesting server). If the requested video/audio is only delayed, then the requesting app/game server 1521-1525 will be so notified, and it will acquire the delayed video/audio from a delay buffer 1515 by looking up the video/audio's location in the directory on the delay buffer 1515 and the network address of the app/game server 1521-1525 that is the source of the delayed video/audio. Once all of these requests have been generated and handled, up to 15 live thumbnail-sized video streams will be routed from the outbound routing 1540 to the inbound routing 1502 to the app/game server 1521-1525 generating the user interface screen, and will be decompressed and displayed by the server. Delayed video/audio streams may be in too large a screen size, and if so, the app/game server 1521-1525 will decompress the streams and scale down the video streams to thumbnail size. In one embodiment, requests for audio/video are sent to (and managed by) a central "management" service similar to the hosting service control system of Figure 4a (not shown in Figure 15) which then redirects the requests to the appropriate app/game server 1521-1525. Moreover, in one embodiment, no request may be required because the thumbnails are "pushed" to the clients of those users that allow it.

The audio from 15 games all mixed simultaneously might create a cacophony of sound. The user may choose to mix all of the sounds together in this way (perhaps just to get a sense of the “din” created by all the action being viewed), or the user may choose to just listen to the audio from one game at a time. The selection of a single game is accomplished by moving the yellow selection box 1601 (appearing as a black rectangular outline in the black-and-white rendering of Figure 16) to a given game (the yellow box movement can be accomplished by using arrow keys on a keyboard, by moving a mouse, by moving a joystick, or by pushing directional buttons on another device such as a mobile phone). Once a single game is selected, just the audio from that game plays. Also, game information 1602 is shown. In the case of this game, for example, the publisher logo (e.g., “EA” for “Electronic Arts”) and the game logo, “e.g., Need for Speed Carbon” and an orange horizontal bar (rendered in Figure 16 as a bar with vertical stripes) indicates in relative terms the number of people playing or viewing the game at that particular moment (many, in this case, so the game is “Hot”). Further “Stats” (i.e. statistics) are provided, indicating that there are 145 players actively playing 80 different instantiations of the Need for Speed Game (i.e., it can be played either by an individual player game or multiplayer game), and there are 680 viewers (of which this user is one). Note that these statistics (and other statistics) are collected by hosting service control system 401 and are stored on RAID arrays 1511-1512, for keeping logs of the hosting service 210 operation and for appropriately billing users and paying publishers who provide content. Some of the statistics are recorded due to actions by the service control system 401, and some are reported to the service control system 401 by the individual app/game server 1521-1525. For example, the app/game server 1521-1525 running this Game Finder application sends messages to the hosting service control system 401 when games are being viewed (and when

they are ceased to be viewed) so that it may update the statistics of how many games are in view. Some of the statistics are available for user interface applications such as this Game Finder application.

If the user clicks an activation button on their input device, they will see the thumbnail video in the yellow box zoom up while continuing to play live video to full screen size. This effect is shown in process in Figure 17. Note that video window 1700 has grown in size. To implement this effect, the app/game server 1521-1525 requests from the app/game server 1521-1525 running the game selected to have a copy of the video stream for a full screen size (at the resolution of the user's display device 422) of the game routed to it. The app/game server 1521-1525 running the game notifies the shared hardware compressor 1530 that a thumbnail-sized copy of the game is no longer needed (unless another app/game server 1521-1525 requires such a thumbnail), and then it directs it to send a full-screen size copy of the video to the app/game server 1521-1525 zooming the video. The user playing the game may or may not have a display device 422 that is the same resolution as that of the user zooming up the game. Further, other viewers of the game may or may not have display devices 422 that are the same resolution as the user zooming up the game (and may have different audio playback means, e.g., stereo or surround sound). Thus, the shared hardware compressor 1530 determines whether a suitable compressed video/audio stream is already being generated that meets the requirements of the user requesting the video/audio stream and if one does exist, it notifies the outbound routing 1540 to route a copy of the stream to the app/game server 1521-1525 zooming the video, and if not compresses another copy of the video that is suitable for that user and instructs the outbound routing to send the stream back to the inbound routing 1502 and the app/game server 1521-1525

zooming the video. This server, now receiving a full screen version of the selected video will decompress it and gradually scale it up to full size.

Figure 18 illustrates how the screen looks after the game has completely zoomed up to full screen and the game is shown at the full resolution of the user's display device 422 as indicated by the image pointed to by arrow 1800. The app/game server 1521-1525 running the game finder application sends messages to the other app/game servers 1521-1525 that had been providing thumbnails that they are no longer needed and messages to the hosting service control server 401 that the other games are no longer being viewed. At this point the only display it is generating is an overlay 1801 at the top of the screen which provides information and menu controls to the user. Note that as this game has progressed, the audience has grown to 2,503 viewers. With so many viewers, there are bound to be many viewers with display devices 422 that have the same or nearly the same resolution (each app/game server 1521-1525 has the ability to scale the video for adjusting the fitting).

Because the game shown is a multiplayer game, the user may decide to join the game at some point. The hosting service 210 may or may not allow the user to join the game for a variety of reasons. For example, the user may have to pay to play the game and choose not to, the user may not have sufficient ranking to join that particular game (e.g., it would not be competitive for the other players), or the user's Internet connection may not have low enough latency to allow the user to play (e.g., there is not a latency constraint for viewing games, so a game that is being played far away (indeed, on another continent) can be viewed without latency concerns, but for a game to be played, the latency must be low enough for the user to (a) enjoy the game, and (b) be on equal footing with the other players who may have lower latency connections). If the user is permitted to play, then app/game

server 1521-1525 that had been providing the Game Finder user interface for the user will request that the hosting service control server 401 initiate (i.e., locate and start up) an app/game server 1521-1525 that is suitably configured for playing the particular game to load the game from a RAID array 1511-1512, and then the hosting service control server 401 will instruct the inbound routing 1502 to transfer the control signals from the user to the app/game game server now hosting the game and it will instruct the shared hardware compression 1530 to switch from compressing the video/audio from the app/game server that had been hosting the Game Finder application to compressing the video/audio from the app/game server now hosting the game. The vertical sync of the Game Finder app/game service and the new app/game server hosting the game are not synchronized, and as a result there is likely to be a time difference between the two syncs. Because the shared video compression hardware 1530 will begin compressing video upon an app/game server 1521-1525 completing a video frame, the first frame from the new server may be completed sooner than a full frame time of the old server, which may be before the prior compressed frame completing its transmission (e.g., consider transmit time 992 of Figure 9b: if uncompressed frame 3 963 were completed half a frame time early, it would impinge upon the transmit time 992). In such a situation the shared video compression hardware 1530 will ignore the first frame from the new server (e.g., like Frame 4 964 is ignored 974), and the client 415 will hold the last frame from the old server an extra frame time, and the shared video compression hardware 1530 will begin compressing the next frame time video from the new app/game server hosting the game. Visually, to the user, the transition from one app/game server to the other will be seamless. The hosting service control server 401 will then notify app/game game server 1521-1525 that had been hosting the Game Finder to switch to an idle state, until it is needed again.

The user then is able to play the game. And, what is exceptional is the game will play perceptually instantly (since it will have loaded onto the app/game game server 1521-1525 from a RAID array 1511-1512 at gigabit/second speed), and the game will be loaded onto a server exactly suited for the game together with an operating system exactly configured for the game with the ideal drivers, registry configuration (in the case of Windows), and with no other applications running on the server that might compete with the game's operation.

Also, as the user progresses through the game, each of the segments of the game will load into the server at gigabit/second speed (i.e., 1 gigabyte loads in 8 seconds) from the RAID array 1511-1512, and because of the vast storage capacity of the RAID array 1511-1512 (since it is a shared resource among many users, it can be very large, yet still be cost effective), geometry setup or other game segment setup can be pre-computed and stored on the RAID array 1511-1512 and loaded extremely rapidly. Moreover, because the hardware configuration and computational capabilities of each app/game server 1521-1525 is known, pixel and vertex shaders can be pre-computed.

Thus, the game will start up almost instantly, it will run in an ideal environment, and subsequent segments will load almost instantly.

But, beyond these advantages, the user will be able to view others playing the game (via the Game Finder, previously described and other means) and both decide if the game is interesting, and if so, learn tips from watching others. And, the user will be able to demo the game instantly, without having to wait for a large download and/or installation, and the user will be able to play the game instantly, perhaps on a trial basis for a smaller fee, or on a longer term basis. And, the user will be able to play the game on a Windows PC, a Macintosh, on a television set, at home, when traveling, and even on a mobile phone, with a low enough latency

wireless connection (although latency will not be an issue for just spectating). And, this can all be accomplished without ever physically owning a copy of the game.

As mentioned previously, the user can decide to not allow his gameplay to be viewable by others, to allow his game to be viewable after a delay, to allow his game to be viewable by selected users, or to allow his game to be viewable by all users. Regardless, the video/audio will be stored, in one embodiment, for 15 minutes in a delay buffer 1515, and the user will be able to "rewind" and view his prior game play, and pause, play it back slowly, fast forward, etc., just as he would be able to do had he been watching TV with a Digital Video Recorder (DVR). Although in this example, the user is playing a game, the same "DVR" capability is available if the user is using an application. This can be helpful in reviewing prior work and in other applications as detailed below. Further, if the game was designed with the capability of rewinding based on utilizing game state information, such that the camera view can be changed, etc., then this "3D DVR" capability will also be supported, but it will require the game to be designed to support it. The "DVR" capability using a delay buffer 1515 will work with any game or application, limited of course, to the video that was generated when the game or application was used, but in the case of games with 3D DVR capability, the user can control a "fly through" in 3D of a previously played segment, and have the delay buffer 1515 record the resulting video and have the game state of the game segment recorded. Thus, a particular "fly-through" will be recorded as compressed video, but since the game state will also be recorded, a different fly-through will be possible at a later date of the same segment of the game.

As described below, users on the hosting service 210 will each have a User Page, where they can post information about themselves and other data. Among of the things that users will be able to post are video segments from game

play that they have saved. For example, if the user has overcome a particularly difficult challenge in a game, the user can “rewind” to just before the spot where they had their great accomplishment in the game, and then instruct the hosting service 210 to save a video segment of some duration (e.g., 30 seconds) on the user’s User Page for other users to watch. To implement this, it is simply a matter of the app/game server 1521-1525 that the user is using to playback the video stored in a delay buffer 1515 to a RAID array 1511-1512 and then index that video segment on the user’s User Page.

If the game has the capability of 3D DVR, as described above, then the game state information required for the 3D DVR can also be recorded by the user and made available for the user’s User Page.

In the event that a game is designed to have “spectators” (i.e., users that are able to travel through the 3D world and observe the action without participating in it) in addition to active players, then the Game Finder application will enable users to join games as spectators as well as players. From an implementation point of view, there is no difference to the hosting system 210 to if a user is a spectator instead of an active player. The game will be loaded onto an app/game server 1521-1525 and the user will be controlling the game (e.g., controlling a virtual camera that views into the world). The only difference will be the game experience of the user.

MULTIPLE USER COLLABORATION

Another feature of the hosting service 210 is the ability to for multiple users to collaborate while viewing live video, even if using widely disparate devices for viewing. This is useful both when playing games and when using applications.

Many PCs and mobile phones are equipped with video cameras and have the capability to do real-time video compression, particularly when the image is

small. Also, small cameras are available that can be attached to a television, and it is not difficult to implement real-time compression either in software or using one of many hardware compression devices to compress the video. Also, many PCs and all mobile phones have microphones, and headsets are available with microphones.

Such cameras and/or microphones, combined with local video/audio compression capability (particularly employing the low latency video compression techniques described herein) will enable a user to transmit video and/or audio from the user premises 211 to the hosting service 210, together with the input device control data. When such techniques are employed, then a capability illustrated in Figure 19 is achievable: a user can have his video and audio 1900 appear on the screen within another user's game or application. This example is a multiplayer game, where teammates collaborate in a car race. A user's video/audio could be selectively viewable / hearable only by their teammates. And, since there would be effectively no latency, using the techniques described above the players would be able to talk or make motions to each other in real-time without perceptible delay.

This video/audio integration is accomplished by having the compressed video and/or audio from a user's camera/microphone arrive as inbound internet traffic 1501. Then the inbound routing 1502 routes the video and/or audio to the app/game game servers 1521-1525 that are permitted to view/hear the video and/or audio. Then, the users of the respective app/game game servers 1521-1525 that choose to use the video and/or audio decompress it and integrate as desired to appear within the game or application, such as illustrated by 1900.

The example of Figure 19 shows how such collaboration is used in a game, but such collaboration can be an immensely powerful tool for applications. Consider a situation where a large building is being designed for New York city by architects in Chicago for a real estate developer based in New York, but the decision

involves a financial investor who is traveling and happens to be in an airport in Miami, and a decision needs to be made about certain design elements of the building in terms of how it fits in with the buildings near it, to satisfy both the investor and the real estate developer. Assume the architectural firm has a high resolution monitor with a camera attached to a PC in Chicago, the real estate developer has a laptop with a camera in New York, and the investor has a mobile phone with a camera in Miami. The architectural firm can use the hosting service 210 to host a powerful architectural design application that is capable of highly realistic 3D rendering, and it can make use of a large database of the buildings in New York City, as well as a database of the building under design. The architectural design application will execute on one, or if it requires a great deal of computational power on several, of the app/game servers 1521-1525. Each of the 3 users at disparate locations will connect to the hosting service 210, and each will have a simultaneous view of the video output of the architectural design application, but it will be will appropriately sized by the shared hardware compression 1530 for the given device and network connection characteristics that each user has (e.g., the architectural firm may see a 2560x1440 60fps display through a 20Mbps commercial Internet connection, the real estate developer in New York may see a 1280x720 60fps image over a 6 Mbps DSL connection on his laptop, and the investor may see a 320x180 60fps image over a 250Kbps cellular data connection on her mobile phone. Each party will hear the voice of the other parties (the conference calling will be handled by any of many widely available conference calling software package in the app/game server(s) 1521-1525) and, through actuation of a button on a user input device, a user will be able to make video appear of themselves using their local camera. As the meeting proceeds, the architects will be able to show what the build looks like as they rotate it and fly by it next to the other building in the area, with extremely photorealistic 3D

rendering, and the same video will be visible to all parties, at the resolution of each party's display device. It won't matter that none of the local devices used by any party is incapable of handling the 3D animation with such realism, let alone downloading or even storing the vast database required to render the surrounding buildings in New York City. From the point of view of each of the users, despite the distance apart, and despite the disparate local devices they simply will have a seamless experience with an incredible degree of realism. And, when one party wants their face to be seen to better convey their emotional state, they can do so. Further, if either the real estate developer or the investor want to take control of the architectural program and use their own input device (be it a keyboard, mouse, keypad or touch screen), they can, and it will respond with no perceptual latency (assuming their network connection does not have unreasonable latency). For example, in the case of the mobile phone, if the mobile phone is connected to a WiFi network at the airport, it will have very low latency. But if it is using the cellular data networks available today in the US, it probably will suffer from a noticeable lag. Still, for most of the purposes of the meeting, where the investor is watching the architects control the building fly-by or for talking of video teleconferencing, even cellular latency should be acceptable.

Finally, at the end of the collaborative conference call, the real estate developer and the investor will have made their comments and signed off from the hosting service, the architectural firm will be able to "rewind" the video of the conference that has been recorded on a delay buffer 1515 and review the comments, facial expressions and/or actions applied to the 3D model of the building made during the meeting. If there are particular segments they want to save, those segments of video/audio can be moved from delay buffer 1515 to a RAID array 1511-1512 for archival storage and later playback.

Also, from a cost perspective, if the architects only need to use the computation power and the large database of New York City for a 15 minute conference call, they need only pay for the time that the resources are used, rather than having to own high powered workstations and having to purchase an expensive copy of a large database.

VIDEO-RICH COMMUNITY SERVICES

The hosting service 210 enables an unprecedented opportunity for establishing video-rich community services on the Internet. Figure 20 shows an exemplary User Page for a game player on the hosting service 210. As with the Game Finder application, the User Page is an application that runs on one of the app/game servers 1521-1525. All of the thumbnails and video windows on this page show constantly moving video (if the segments are short, they loop).

Using a video camera or by uploading video, the user (whose username is "KILLHAZARD") is able to post a video of himself 2000 that other users can view. The video is stored on a RAID array 1511-1512. Also, when other users come to KILLHAZARD's User Page, if KILLHAZARD is using the hosting service 210 at the time, live video 2001 of whatever he is doing (assuming he permits users viewing his User Page to watch him) will be shown. This will be accomplished by app/game server 1521-1525 hosting the User Page application requesting from the service control system 401 whether KILLHAZARD is active and if so, the app/game server 1521-1525 he is using. Then, using the same methods used by the Game Finder application, a compressed video stream in a suitable resolution and format will be sent to the app/game server 1521-1525 running the User Page application and it will be displayed. If a user selects the window with KILLHAZARD's live gameplay, and then appropriately clicks on their input device, the window will zoom up (again using the same methods as the Game Finder applications, and the live

video will fill the screen, at the resolution of the watching user's display device 422, appropriate for the characteristics of the watching user's Internet connection.

A key advantage of this over prior art approaches is the user viewing the User Page is able to see a game played live that the user does not own, and may very well not have a local computer or game console capable of playing the game. It offers a great opportunity for the user to see the user shown in the User Page "in action" playing games, and it is an opportunity to learn about a game that the viewing user might want to try or get better at.

Camera-recorded or uploaded video clips from KILLHAZARD's buddies 2002 are also shown on the User Page, and underneath each video clip is text that indicates whether the buddy is online playing a game (e.g., six_shot is playing the game "Eragon" (shown here as Game4) and MrSnuggles99 is Offline, etc.). By clicking on a menu item (not shown) the buddy video clips switch from showing recorded or uploaded videos to live video of what the buddies who are currently playing games on the hosting service 210 are doing at that moment in their games. So, it becomes a Game Finder grouping for buddies. If a buddy's game is selected and the user clicks on it, it will zoom up to full screen, and the user will be able to watch the game played full screen live.

Again, the user viewing the buddy's game does not own a copy of the game, nor the local computing/game console resources to play the game. The game viewing is effectively instantaneous.

As previously described above, when a user plays a game on the hosting service 210, the user is able to "rewind" the game and find a video segment he wants to save, and then saves the video segment to his User Page. These are called "Brag Clips™". The video segments 2003 are all Brag Clips 2003 saved by KILLHAZARD from previous games that he has played. Number 2004 shows how

many times a Brag Clip has been viewed, and when the Brag Clip is viewed, users have an opportunity to rate them, and the number of orange (shown here as black outlines) keyhole-shaped icons 2005 indicate how high the rating is. The Brag Clips 2003 loop constantly when a user views the User Page, along with the rest of the video on the page. If the user selects and clicks on one of the Brag Clips 2003, it zooms up to present the Brag Clip 2003, along with DVR controls to allow the clip to be played, paused, rewound, fast-forwarded, stepped through, etc.

The Brag Clip 2003 playback is implemented by the app/game server 1521-1525 loading the compressed video segment stored on a RAID array 1511-1512 when the user recorded the Brag Clip and decompressing it and playing it back.

Brag Clips 2003 can also be “3D DVR” video segments (i.e., a game state sequence from the game that can be replayed and allows the user to change the camera viewpoint) from games that support such capability. In this case the game state information is stored, in addition to a compressed video recording of the particular “fly through” the user made when the game segment was recorded. When the User Page is being viewed, and all of the thumbnails and video windows are constantly looping, a 3D DVR Brag Clip 2003 will constantly loop the Brag Clip 2003 that was recorded as compressed video when the user recorded the “fly through” of the game segment. But, when a user selects a 3D DVR Brag Clip 2003 and clicks on it, in addition to the DVR controls to allow the compressed video Brag Clip to be played, the user will be able to click on a button that gives them 3D DVR capability for the game segment. They will be able to control a camera “fly through” during the game segment on their own, and, if they wish (and the user who owns the user page so allows it) they will be able to record an alternative Brag Clip “fly through” in compressed video form will then be available to other viewers of the user page

(either immediately, or after the owner of the user page has a chance to review the Brag Clip).

This 3D DVR Brag Clip 2003 capability is enabled by activating the game that is about to replay the recorded game state information on another app/game server 1521-1525. Since the game can be activated almost instantaneously (as previously described) it is not difficult to activate it, with its play limited to the game state recorded by the Brag Clip segment, and then allow the user to do a “fly through” with a camera while recording the compressed video to a delay buffer 1515. Once the user has completed doing the “fly through” the game is deactivated.

From the user's point of view, activating a “fly through” with a 3D DVR Brag Clip 2003 is no more effort than controlling the DVR controls of a linear Brag Clip 2003. They may know nothing about the game or even how to play the game. They are just a virtual camera operator peering into a 3D world during a game segment recorded by another.

Users will also be able to overdub their own audio onto Brag Clips that is either recorded from microphones or uploaded. In this way, Brag Clips can be used to create custom animations, using characters and actions from games. This animation technique is commonly known as “machinima”.

As users progress through games, they will achieve differing skill levels. The games played will report the accomplishments to the service control system 401, and these skill levels will be shown on User Pages.

INTERACTIVE ANIMATED ADVERTISEMENTS

Online advertisements have transitioned from text, to still images, to video, and now to interactive segments, typically implemented using animation thin clients like Adobe Flash. The reason animation thin clients are used is that users

typically have little patience to be delayed for the privilege of having a product or service pitched to them. Also, thin clients run on very low-performance PCs and as such, the advertiser can have a high degree of confidence that the interactive ad will work properly. Unfortunately, animation thin clients such as Adobe Flash are limited in the degree of interactivity and the duration of the experience (to mitigate download time and to be operable on almost all user devices, including low-performance PCs and Macs without GPUs or high-performance CPUs).

Figure 21 illustrates an interactive advertisement where the user is to select the exterior and interior colors of a car while the car rotates around in a showroom, while real-time ray tracing shows how the car looks. Then the user chooses an avatar to drive the car, and then the user can take the car for a drive either on a race track, or through an exotic locale such as Monaco. The user can select a larger engine, or better tires, and then can see how the changed configuration affects the ability of the car to accelerate or hold the road.

Of course, the advertisement is effectively a sophisticated 3D video game. But for such an advertisement to be playable on a PC or a video game console it would require perhaps a 100MB download and, in the case of the PC, it might require the installation of special drivers, and might not run at all if the PC lacks adequate CPU or GPU computing capability. Thus, such advertisements are impractical in prior art configurations.

In the hosting service 210, such advertisements launch almost instantly, and run perfectly, no matter what the user's client 415 capabilities are. So, they launch more quickly than thin client interactive ads, are vastly richer in the experience, and are highly reliable.

STREAMING GEOMETRY DURING REAL-TIME ANIMATION

RAID array 1511-1512 and the inbound routing 1502 can provide data rates that are so fast and with latencies so low that it is possible to design video games and applications that rely upon the RAID array 1511-1512 and the inbound routing 1502 to reliably deliver geometry on-the-fly in the midst of game play or in an application during real-time animation (e.g., a fly-through with a complex database.)

With prior art systems, such as the video game system shown in Figure 1, the mass storage devices available, particularly in practical home devices, are far too slow to stream geometry in during game play except in situations where the required geometry was somewhat predictable. For example, in a driving game where there is a specified roadway, geometry for buildings that are coming into view can be reasonably well predicted and the mass storage devices can seek in advance to the location where the upcoming geometry is located.

But in a complex scene with unpredictable changes (e.g., in a battle scene with complex characters all around) if RAM on the PC or video game system is completely filled with geometry for the objects currently in view, and then the user suddenly turns their character around to view what is behind their character, if the geometry has not been pre-loaded into RAM, then there may be a delay before it can be displayed.

In the hosting service 210, the RAID arrays 1511-1512 can stream data in excess of Gigabit Ethernet speed, and with a SAN network, it is possible to achieve 10 gigabit/second speed over 10 Gigabit Ethernet or over other network technologies. 10 gigabits/second will load a gigabyte of data in less than a second. In a 60fps frame time (16.67ms), approximately 170 megabits (21MB) of data can be loaded. Rotating media, of course, even in a RAID configuration will still incur latencies greater than a frame time, but Flash-based RAID storage will eventually be as large as rotating media RAID arrays and will not incur such high latency. In one

embodiment, massive RAM write-through caching is used to provide very low latency access.

Thus, with sufficiently high network speed, and sufficiently low enough latency mass storage, geometry can be streamed into app/game servers 1521-1525 as fast as the CPUs and/or GPUs can process the 3D data. So, in the example given previously, where a user turns their character around suddenly and looks behind, the geometry for all of the characters behind can be loaded before the character completes the rotation, and thus, to the user, it will seem as if he or she is in a photorealistic world that is as real as live action.

As previously discussed, one of the last frontiers in photorealistic computer animation is the human face, and because of the sensitivity of the human eye to imperfections, the slightest error from a photoreal face can result in a negative reaction from the viewer. Figure 22 shows how a live performance captured using Contour™ Reality Capture Technology (subject of co-pending applications: "Apparatus and method for capturing the motion of a performer," Ser. No. 10/942,609, Filed September 15, 2004; "Apparatus and method for capturing the expression of a performer," Ser. No. 10/942,413 Filed September 15, 2004; "Apparatus and method for improving marker identification within a motion capture system," Ser. No. 11/066,954, Filed February 25, 2005; "Apparatus and method for performing motion capture using shutter synchronization," Ser. No. 11/077,628, Filed March 10, 2005; "Apparatus and method for performing motion capture using a random pattern on capture surfaces," Ser. No. 11/255,854, Filed October 20, 2005; "System and method for performing motion capture using phosphor application techniques," Ser. No. 11/449,131, Filed June 7, 2006; "System and method for performing motion capture by strobing a fluorescent lamp," Ser. No. 11/449,043, Filed June 7, 2006; "System and method for three dimensional capture of stop-

motion animated characters,” Ser. No. 11/449,127, Filed June 7, 2006”, each of which is assigned to the assignee of the present CIP application) results in a very smooth captured surface, then a high polygon-count tracked surface (i.e., the polygon motion follows the motion of the face precisely). Finally, when the video of the live performance is mapped on the tracked surface to produce a textured surface, a photoreal result is produced.

Although current GPU technology is able to render the number of polygons in the tracked surface and texture and light the surface in real-time, if the polygons and textures are changing every frame time (which will produce the most photoreal results) it will quickly consume all the available RAM of a modern PC or video game console.

Using the streaming geometry techniques described above, it becomes practical to continuously feed geometry into the app/game game servers 1521-1525 so that they can animate photoreal faces continuously, allowing the creation of video games with faces that are almost indistinguishable from live action faces.

INTEGRATION OF LINEAR CONTENT WITH INTERACTIVE FEATURES

Motion pictures, television programming and audio material (collectively, “linear content”) is widely available to home and office users in many forms. Linear content can be acquired on physical media, like CD, DVD and Blu-ray media. It also can be recorded by DVRs from satellite and cable TV broadcast. And, it is available as pay-per-view (PPV) content through satellite and cable TV and as video-on-demand (VOD) on cable TV.

Increasingly linear content is available through the Internet, both as downloaded and as streaming content. Today, there really is not one place to go to experience all of the features associated with linear media. For example, DVDs and other video optical media typically have interactive features not available elsewhere,

like director's commentaries, "making of" featurettes, etc. Online music sites have cover art and song information generally not available on CDs, but not all CDs are available online. And Web sites associated with television programming often have extra features, blogs and sometimes comments from the actors or creative staff.

Further, with many motion pictures or sports events, there are often video games that are released (in the case of motion pictures) often together with the linear media or (in the case of sports) may be closely tied to real-world events (e.g., the trading of players).

Hosting service 210 is well suited for the delivery of linear content in linking together the disparate forms of related content. Certainly, delivering motion pictures is no more challenging than delivering highly interactive video games, and the hosting service 210 is able to deliver linear content to a wide range of devices, in the home or office, or to mobile devices. Figure 23 shows an exemplary user interface page for hosting service 210 that shows a selection of linear content.

But, unlike most linear content delivery system, hosting service 210 is also able to deliver related interactive components (e.g., the menus and features on DVDs, the interactive overlays on HD-DVDs, and the Adobe Flash animation (as explained below) on Web sites). Thus, the client device 415 limitations no longer introduce limitations as to which features are available.

Further, the hosting system 210 is able to link together linear content with video game content dynamically, and in real-time. For example, if a user is watching a Quidditch match in a Harry Potter movie, and decides she would like to try playing Quidditch, she can just click a button and the movie will pause and immediately she will be transported to the Quidditch segment of a Harry Potter video game. After playing the Quidditch match, another click of a button, and the movie will resume instantly.

With photoreal graphics and production technology, where the photographically-captured video is indistinguishable from the live action characters, when a user makes a transition from a Quidditch game in a live action movie to a Quidditch game in a video game on a hosting service as described herein, the two scenes are virtually indistinguishable. This provides entirely new creative options for directors of both linear content and interactive (e.g., video game) content as the lines between the two worlds become indistinguishable.

Utilizing the hosting service architecture shown in Fig. 14 the control of the virtual camera in a 3D movie can be offered to the viewer. For example, in a scene that takes place within a train car, it would be possible to allow the viewer to control the virtual camera and look around the car while the story progresses. This assumes that all of the 3D objects ("assets") in the car are available as well as an adequate a level of computing power capable of rendering the scenes in real-time as well as the original movie.

And even for non-computer generated entertainment, there are very exciting interactive features that can be offered. For example, the 2005 motion picture "Pride and Prejudice" had many scenes in ornate old English mansions. For certain mansion scenes, the user may pause the video and then control the camera to take a tour of the mansion, or perhaps the surrounding area. To implement this, a camera could be carried through the mansion with a fish-eye lens as it keeps track of its position, much like prior art Apple, Inc. QuickTime VR is implemented. The various frames would then be transformed so the images are not distorted, and then stored on RAID array 1511-1512 along with the movie, and played back when the user chooses to go on a virtual tour.

With sports events, a live sports event, such as a basketball game, may be streamed through the hosting service 210 for users to watch, as they would

for regular TV. After users watched a particular play, a video game of the game (eventually with basketball players looking as photoreal as the real players) could come up with the players starting in the same position, and the users (perhaps each taking control of one player) could redo the play to see if they could do better than the players.

The hosting service 210 described herein is extremely well-suited to support this futuristic world because it is able to bring to bear computing power and mass storage resources that are impractical to install in a home or in most office settings, and also its computing resources are always up-to-date, with the latest computing hardware available, whereas in a home setting, there will always be homes with older generation PCs and video games. And, in the hosting service 210, all of this computing complexity is hidden from the user, so even though they may be using very sophisticated systems, from the user's point of view, it is as simple as changing channels on a television. Further, the users would be able to access all of the computing power and the experiences the computing power would bring from any client 415.

MULTIPLAYER GAMES

To the extent a game is a multiplayer game, then it will be able to communicate both to app/game game servers 1521-1525 through the inbound routing 1502 network and, with a network bridge to the Internet (not shown) with servers or game machines that are not running in the hosting service 210. When playing multiplayer games with computers on the general Internet, then the app/game game servers 1521-1525 will have the benefit of extremely fast access to the Internet (compared to if the game was running on a server at home), but they will be limited by the capabilities of the other computers playing the game on slower connections, and also potentially limited by the fact that the game servers on the

Internet were designed to accommodate the least common denominator, which would be home computers on relatively slow consumer Internet connections.

But when a multiplayer game is played entirely within a hosting service 210 server center, then a world of difference is achievable. Each app/game game server 1521-1525 hosting a game for a user will be interconnected with other app/game game servers 1521-1525 as well as any servers that are hosting the central control for the multiplayer game with extremely high speed, extremely low latency connectivity and vast, very fast storage arrays. For example, if Gigabit Ethernet is used for the inbound routing 1502 network, then the app/game game servers 1521-1525 will be communicating among each other and communicating to any servers hosting the central control for the multiplayer game at gigabit/second speed with potentially only 1ms of latency or less. Further, the RAID arrays 1511-1512 will be able to respond very rapidly and then transfer data at gigabit/second speeds. As an example, if a user customizes a character in terms of look and accoutrements such that the character has a large amount of geometry and behaviors that are unique to the character, with prior art systems limited to the game client running in the home on a PC or game console, if that character were to come into view of another user, the user would have to wait until a long, slow download completes so that all of the geometry and behavior data loads into their computer. Within the hosting service 210, that same download could be over Gigabit Ethernet, served from a RAID array 1511-1512 at gigabit/second speed. Even if the home user had an 8Mbps Internet connection (which is extremely fast by today's standards), Gigabit Ethernet is 100 times faster. So, what would take a minute over a fast Internet connection, would take less than a second over Gigabit Ethernet.

TOP PLAYER GROUPINGS AND TOURNAMENTS

The Hosting Service 210 is extremely well-suited for tournaments. Because no game is running in a local client, there is no opportunity for users to cheat (e.g., as they might have in a prior art tournament by modifying the copy of the game running on their local PC to give them an unfair advantage). Also, because of the ability of the output routing 1540 to multicast the UDP streams, the Hosting Service 210 is able to broadcast the major tournaments to thousands or more of people in the audience at once.

In fact, when there are certain video streams that are so popular that thousands of users are receiving the same stream (e.g., showing views of a major tournament), it may be more efficient to send the video stream to a Content Delivery Network (CDN) such as Akamai or Limelight for mass distribution to many client devices 415.

A similar level of efficiency can be gained when a CDN is used to show Game Finder pages of top player groupings.

For major tournaments, a live celebrity announcer can be used to provide commentary during certain matches. Although a large number of users will be watching a major tournament, and relatively small number will be playing in the tournament. The audio from the celebrity announcer can be routed to the app/game game servers 1521-1525 hosting the users playing in the tournament and hosting any spectator-mode copies of the game in the tournament, and the audio can be overdubbed on top of the game audio. Video of a celebrity announcer can be overlaid on the games, perhaps just on spectator views, as well.

ACCELERATION OF WEB PAGE LOADING

The World Wide Web and its primary transport protocol, Hypertext Transfer Protocol (HTTP), were conceived and defined in an era where only businesses had high speed Internet connections, and the consumers who were

online were using dialup modems or ISDN. At the time, the “gold standard” for a fast connection was a T1 line which provided 1.5Mbps data rate symmetrically (i.e., with equal data rate in both directions).

Today, the situation is completely different. The average home connection speed through DSL or cable modem connections in much of the developed world has a far higher downstream data rate than a T1 line. In fact, in some parts of the world, fiber-to-the-curb is bringing data rates as high as 50 to 100Mbps to the home.

Unfortunately, HTTP was not architected (nor has it been implemented) to effectively take advantage of these dramatic speed improvements. A web site is a collection of files on a remote server. In very simple terms, HTTP requests the first file, waits for the file to be downloaded, and then requests the second file, waits for the file to be downloaded, etc. In fact, HTTP allows for more than one “open connection”, i.e., more than one file to be requested at a time, but because of agreed-upon standards (and a desire to prevent web servers from being overloaded) only very few open connections are permitted. Moreover, because of the way Web pages are constructed, browsers often are not aware of multiple simultaneous pages that could be available to download immediately (i.e., only after parsing a page does it become apparent that a new file, like an image, needs to be downloaded). Thus, files on website are essentially loaded one-by-one. And, because of the request-and-response protocol used by HTTP, there is roughly (accessing typical web servers in the US) a 100ms latency associated with each file that is loaded.

With relatively low speed connections, this does not introduce much of a problem because the download time for the files themselves dominates the waiting

time for the web pages. But, as connection speeds grow, especially with complex web pages, problems begin to arise.

In the example shown in Figure 24, a typical commercial website is shown (this particular website was from a major athletic shoe brand). The website has 54 files on it. The files include HTML, CSS, JPEG, PHP, JavaScript and Flash files, and include video content. A total of 1.5MBytes must be loaded before the page is live (i.e., the user can click on it and begin to use it). There are a number of reasons for the large number of files. For one thing, it is a complex and sophisticated webpage, and for another, it is a webpage that is assembled dynamically based on the information about the user accessing the page (e.g., what country the user is from, what language, whether the user has made purchases before, etc.), and depending on all of these factors, different files are downloaded. Still, it is a very typical commercial web page.

Figure 24 shows the amount of time that elapses before the web page is live as the connection speed grows. With a 1.5Mbps connection speed 2401, using a conventional web server with a convention web browser, it takes 13.5 seconds until the web page is live. With a 12Mbps connection speed 2402, the load time is reduced to 6.5 seconds, or about twice as fast. But with a 96Mbps connection speed 2403, the load time is only reduced to about 5.5 seconds. The reason why is because at such a high download speed, the time to download the files themselves is minimal, but the latency per file, roughly 100ms each, still remains, resulting in $54 \text{ files} * 100\text{ms} = 5.4 \text{ seconds}$ of latency. Thus, no matter how fast the connection is to the home, this web site will always take at least 5.4 seconds until it is live. Another factor is the server-side queuing; every HTTP request is added in the back of the queue, so on a busy server this will have a significant impact because for every small item to get from the web server, the HTTP requests needs to wait for its turn.

One way to solve these issues is to discard or redefine HTTP. Or, perhaps to get the website owner to better consolidate its files into a single file (e.g., in Adobe Flash format). But, as a practical matter, this company, as well as many others has a great deal of investment in their web site architecture. Further, while some homes have 12-100Mbps connections, the majority of homes still have slower speeds, and HTTP does work well at slow speed.

One alternative is to host web browsers on app/game servers 1521-1525, and host the files for the web servers on the RAID arrays 1511-1512 (or potentially in RAM or on local storage on the app/game servers 1521-1525 hosting the web browsers. Because of the very fast interconnect through the inbound routing 1502 (or to local storage), rather than have 100ms of latency per file using HTTP, there will be de minimis latency per file using HTTP. Then, instead of having the user in her home accessing the web page through HTTP, the user can access the web page through client 415. Then, even with a 1.5Mbps connection (because this web page does not require much bandwidth for its video), the webpage will be live in less than 1 second per line 2400. Essentially, there will be no latency before the web browser running on an app/game server 1521-1525 is displaying a live page, and there will be no detectable latency before the client 415 displays the video output from the web browser. As the user mouses around and/or types on the web page, the user's input information will be sent to the web browser running on the app/game server 1521-1525, and the web browser will respond accordingly.

One disadvantage to this approach is if the compressor is constantly transmitting video data, then bandwidth is used, even if the web page becomes static. This can be remedied by configuring the compressor to only transmit data when (and if) the web page changes, and then, only transmit data to the parts of the page that change. While there are some web pages with flashing banners, etc. that

are constantly changing, such web pages tend to be annoying, and usually web pages are static unless there is a reason for something to be moving (e.g., a video clip). For such web pages, it is likely the case the less data will be transmitted using the hosting service 210 than a conventional web server because only the actual displayed images will be transmitted, no thin client executable code, and no large objects that may never be viewed, such as rollover images.

Thus, using the hosting service 210 to host legacy web pages, web page load times can be reduced to the point where opening a web page is like changing channels on a television: the web page is live effectively instantly.

FACILITATING DEBUGGING OF GAMES AND APPLICATIONS

As mentioned previously, video games and applications with real-time graphics are very complex applications and typically when they are released into the field they contain bugs. Although software developers will get feedback from users about bugs, and they may have some means to pass back machine state after crashes, it is very difficult to identify exactly what has caused a game or real-time application to crash or to perform improperly.

When a game or application runs in the hosting service 210, the video/audio output of the game or application is constantly recorded on a delay buffer 1515. Further, a watchdog process runs on each app/game server 1521-1525 which reports regularly to the hosting service control system 401 that the app/game server 1521-1525 is running smoothly. If the watchdog process fails to report in, then the server control system 401 will attempt to communicate with the app/game server 1521-1525, and if successful, will collect whatever machine state is available. Whatever information is available, along with the video/audio recorded by the delay buffer 1515 will be sent to the software developer.

Thus, when the game or application software developer gets notification of a crash from the hosting service 210, it gets a frame-by-frame record of what led up to the crash. This information can be immensely valuable in tracking down bugs and fixing them.

Note also, that when an app/game server 1521-1525 crashes, the server is restarted at the most recent restartable point, and a message is provided to the user apologizing for the technical difficulty.

RESOURCE SHARING AND COST SAVINGS

The system shown in Figures 4a and 4b provide a variety of benefits for both end users and game and application developers. For example, typically, home and office client systems (e.g., PCs or game consoles) are only in use for a small percentage of the hours in a week. According to an October 5, 2006 press release by the Nielsen Entertainment "Active Gamer Benchmark Study" (<http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/10-05-2006/0004446115&EDATE=>) active gamers spend on average 14 hours a week playing on video game consoles and about 17 hours a week on handhelds. The report also states that for all game playing activity (including console, handheld and PC game playing) Active Gamers average 13 hours a week. Taking into consideration the higher figure of console video game playing time, there are $24 \times 7 = 168$ hours in a week, that implies that in an active gamer's home, a video game console is in use only $17/168 = 10\%$ of the hours of a week. Or, 90% of the time, the video game console is idle. Given the high cost of video game consoles, and the fact that manufacturers subsidize such devices, this is a very inefficient use of an expensive resource. PCs within businesses are also typically used only a fraction of the hours of the week, especially non-portable desktop PCs often required for high-end applications such as Autodesk Maya. Although some businesses operate at all

hours and on holidays, and some PCs (e.g., portables brought home for doing work in the evening) are used at all hours and holidays, most business activities tend to center around 9AM to 5PM, in a given business' time zone, from Monday to Friday, less holidays and break times (such as lunch), and since most PC usage occurs while the user is actively engaged with the PC, it follows that desktop PC utilization tends to follow these hours of operation. If we were to assume that PCs are utilized constantly from 9AM to 5PM, 5 days a week, that would imply PCs are utilized $40/168=24\%$ of the hours of the week. High-performance desktop PCs are very expensive investments for businesses, and this reflects a very low level of utilization. Schools that are teaching on desktop computers may use computers for an even smaller fraction of the week, and although it varies depending upon the hours of teaching, most teaching occurs during the daytime hours from Monday through Friday. So, in general, PCs and video game consoles are utilized only a small fraction of the hours of the week.

Notably, because many people are working at businesses or at school during the daytime hours of Monday through Friday on non-holidays, these people generally are not playing video games during these hours, and so when they do play video games it is generally during other hours, such as evenings, weekends and on holidays.

Given the configuration of the hosting service shown in Figure 4a, the usage patterns described in the above two paragraphs result in very efficient utilization of resources. Clearly, there is a limit to the number of users who can be served by the hosting service 210 at a given time, particularly if the users are requiring real-time responsiveness for complex applications like sophisticated 3D video games. But, unlike a video game console in a home or a PC used by a business, which typically sits idle most of the time, servers 402 can be re-utilized by

different users at different times. For example, a high-performance server 402 with high performance dual CPUs and dual GPUs and a large quantity of RAM can be utilized by businesses and schools from 9AM to 5PM on non-holidays, but be utilized by gamers playing a sophisticated video game in the evenings, weekends and on holidays. Similarly, low-performance applications can be utilized by businesses and schools on a low-performance server 402 with a Celeron CPU, no GPU (or a very low-end GPU) and limited RAM during business hours and a low-performance game can utilize a low-performance server 402 during non-business hours.

Further, with the hosting service arrangement described herein, resources are shared efficiently among thousands, if not millions, of users. In general, online services only have a small percentage of their total user base using the service at a given time. If we consider the Nielsen video game usage statistics listed previously, it is easy to see why. If active gamers play console games only 17 hours of a week, and if we assume that the peak usage time for game is during the typical non-work, non-business hours of evenings (5-12AM, $7 \times 5 \text{ days} = 35$ hours/week) and weekend (8AM-12AM, $16 \times 2 = 32$ hours/week), then there are $35 + 32 = 65$ peak hours a week for 17 hours of game play. The exact peak user load on the system is difficult to estimate for many reasons: some users will play during off-peak times, there may be certain day times when there are clustering peaks of users, the peak times can be affected by the type of game played (e.g., children's games will likely be played earlier in the evening), etc. But, given that the average number of hours played by a gamer is far less than the number of hours of the day when a gamer is likely to play a game, only a fraction of the number of users of the hosting service 210 will be using it at a given time. For the sake of this analysis, we shall assume the peak load is 12.5%. Thus, only 12.5% of the computing,

compression and bandwidth resources are used at a given time, resulting in only 12.5% of the hardware cost to support a given user to play a given level of performance game due to reuse of resources.

Moreover, given that some games and applications require more computing power than others, resources may be allocated dynamically based on the game being played or the applications executed by users. So, a user selecting a low-performance game or application will be allocated a low-performance (less expensive) server 402, and a user selecting a high-performance game or applications will be allocated a high-performance (more expensive) server 402. Indeed, a given game or application may have lower-performance and higher-performance sections of the game or applications, and the user can be switched from one server 402 to another server 402 between sections of the game or application to keep the user running on the lowest-cost server 402 that meets the game or application's needs. Note that the RAID arrays 405, which will be far faster than a single disk, will be available to even low-performance servers 402, that will have the benefit of the faster disk transfer rates. So, the average cost per server 402 across all of the games being played or applications being used is much less than the cost of the most expensive server 402 that plays the highest performance game or applications, yet even the low-performance servers 402, will derive disk performance benefits from the RAID arrays 405.

Further, a server 402 in the hosting service 210 may be nothing more than a PC motherboard without a disk or peripheral interfaces other than a network interface, and in time, may be integrated down to a single chip with just a fast network interface to the SAN 403. Also, RAID Arrays 405 likely will be shared amongst far many more users than there are disks, so the disk cost per active user will be far less than one disk drive. All of this equipment will likely reside in a rack in

an environmentally-controlled server room environment. If a server 402 fails, it can be readily repaired or replaced at the hosting service 210. In contrast, a PC or game console in the home or office must be a sturdy, standalone appliance that has to be able to survive reasonable wear and tear from being banged or dropped, requires a housing, has at least one disk drive, has to survive adverse environment conditions (e.g., being crammed into an overheated AV cabinet with other gear), requires a service warranty, has to be packaged and shipped, and is sold by a retailer who will likely collect a retail margin. Further, a PC or game console must be configured to meet the peak performance of the most computationally-intensive anticipated game or application to be used at some point in the future, even though lower performance games or application (or sections of games or applications) may be played most of the time. And, if the PC or console fails, it is an expensive and time-consuming process (adversely impacting the manufacturer, user and software developer) to get it repaired.

Thus, given that the system shown in Figure 4a provides an experience to the user comparable to that of a local computing resource, for a user in the home, office or school to experience a given level of computing capability, it is much less expensive to provide that computing capability through the architecture shown in Figure 4a.

ELIMINATING THE NEED TO UPGRADE

Further, users no longer have to worry about upgrading PCs and/or consoles to play new games or handle higher performance new applications. Any game or applications on the hosting service 210, regardless of what type of server 402 is required for that game or applications, is available to the user, and all games and applications run nearly instantly (e.g., loading rapidly from the RAID Arrays 405 or local storage on a servers 402) and properly with the latest updates and bug fixes

(i.e., software developers will be able to choose an ideal server configuration for the server(s) 402 that run(s) a given game or application, and then configure the server(s) 402 with optimal drivers, and then over time, the developers will be able to provide updates, bug fixes, etc. to all copies of the game or application in the hosting service 210 at once). Indeed, after the user starts using the hosting service 210, the user is likely to find that games and applications continue to provide a better experience (e.g., through updates and/or bug fixes) and it may be the case that a user discovers a year later that a new game or application is made available on the service 210 that is utilizing computing technology (e.g., a higher-performance GPU) that did not even exist a year before, so it would have been impossible for the user to buy the technology a year before that would play the game or run the applications a year later. Since the computing resource that is playing the game or running the application is invisible to the user (i.e., from the user's perspective the user is simply selecting a game or application that begins running nearly instantly—much as if the user had changed channels on a television), the user's hardware will have been “upgraded” without the user even being aware of the upgrade.

ELIMINATING THE NEED FOR BACKUPS

Another major problem for users in businesses, schools and homes are backups. Information stored in a local PC or video game console (e.g., in the case of a console, a user's game achievements and ranking) can be lost if a disk fails, or if there is an inadvertent erasure. There are many applications available that provide manual or automatic backups for PCs, and game console state can be uploaded to an online server for backup, but local backups are typically copied to another local disk (or other non-volatile storage device) which has to be stored somewhere safe and organized, and backups to online services are often limited because of the slow upstream speed available through typical low-cost Internet

connections. With the hosting service 210 of Figure 4a, the data that is stored in RAID arrays 405 can be configured using prior art RAID configuration techniques well-known to those skilled in the art such that if a disk fails, no data will be lost, and a technician at the server center housing the failed disk will be notified, and then will replace the disk, which then will be automatically updated so that the RAID array is once again failure tolerant. Further, since all of the disk drives are near one another and with fast local networks between them through the SAN 403 it is not difficult in a server center to arrange for all of the disk systems to be backed up on a regular basis to secondary storage, which can be either stored at the server center or relocated offsite. From the point of view of the users of hosting service 210, their data is simply secure all the time, and they never have to think about backups.

ACCESS TO DEMOS

Users frequently want to try out games or applications before buying them. As described previously, there are prior art means by which to demo (the verb form of "demo" means to try out a demonstration version, which is also called a "demo", but as a noun) games and applications, but each of them suffers from limitations and/or inconveniences. Using the hosting service 210, it is easy and convenient for users to try out demos. Indeed, all the user does is select the demo through a user interface (such as one described below) and try out the demo. The demo will load almost instantly onto a server 402 appropriate for the demo, and it will just run like any other game or application. Whether the demo requires a very high performance server 402, or a low performance server 402, and no matter what type of home or office client 415 the user is using, from the point of view of the user, the demo will just work. The software publisher of either the game or application demo will be able to control exactly what demo the user is permitted to try out and for how long, and of course, the demo can include user interface elements that offer the user

an opportunity to gain access to a full version of the game or application demonstrated.

Since demos are likely to be offered below cost or free of charge, some users may try to use demos repeated (particularly game demos, which may be fun to play repeatedly). The hosting service 210 can employ various techniques to limit demo use for a given user. The most straightforward approach is to establish a user ID for each user and limit the number of times a given user ID is allowed to play a demo. A user, however, may set up multiple user IDs, especially if they are free. One technique for addressing this problem is to limit the number of times a given client 415 is allowed to play a demo. If the client is a standalone device, then the device will have a serial number, and the hosting service 210 can limit the number of times a demo can be accessed by a client with that serial number. If the client 415 is running as software on a PC or other device, then a serial number can be assigned by the hosting service 210 and stored on the PC and used to limit demo usage, but given that PCs can be reprogrammed by users, and the serial number erased or changed, another option is for the hosting service 210 to keep a record of the PC network adapter Media Access Control (MAC) address (and/or other machine specific identifiers such as hard-drive serial numbers, etc.) and limit demo usage to it. Given that the MAC addresses of network adapters can be changed, however, this is not a foolproof method. Another approach is to limit the number of times a demo can be played to a given IP address. Although IP addresses may be periodically reassigned by cable modem and DSL providers, it does not happen in practice very frequently, and if it can be determined (e.g., by contacting the ISP) that the IP is in a block of IP addresses for residential DSL or cable modem accesses, then a small number of demo uses can typically be established for a given home. Also, there may be multiple devices at a home behind a NAT router sharing the same IP address, but

typically in a residential setting, there will be a limited number of such devices. If the IP address is in a block serving businesses, then a larger number of demos can be established for a business. But, in the end, a combination of all of the previously mentioned approaches is the best way to limit the number of demos on PCs. Although there may be no foolproof way that a determined and technically adept user can be limited in the number of demos played repeatedly, creating a large number of barriers can create a sufficient deterrent such that it's not worth the trouble for most PC users to abuse the demo system, and rather they will use the demos as they were intended to be used: to try out new games and applications.

BENEFITS TO SCHOOLS, BUSINESSES AND OTHER INSTITUTIONS

Significant benefits accrue particularly to businesses, schools and other institutions that utilize the system shown in Figure 4a. Businesses and schools have substantial costs associated with installing, maintaining and upgrading PCs, particularly when it comes to PCs for running high-performance applications, such as Maya. As stated previously, PCs are generally utilized only a fraction of the hours of the week, and as in the home, the cost of PC with a given level of performance capability is far higher in an office or school environment than in a server center environment.

In the case of larger businesses or schools (e.g., large universities), it may be practical for the IT departments of such entities to set up server centers and maintain computers that are remotely accessed via LAN-grade connections. A number of solutions exist for remote access of computers over a LAN or through a private high bandwidth connection between offices. For example, with Microsoft's Windows Terminal Server, or through virtual network computing applications like VNC, from RealVNC, Ltd., or through thin client means from Sun Microsystems, users can gain remote access to PCs or servers, with a range of quality in graphics

response time and user experience. Further, such self-managed server centers are typically dedicated for a single business or school and as such, are unable to take advantage of the overlap of usage that is possible when disparate applications (e.g., entertainment and business applications) utilize the same computing resources at different times of the week. So, many businesses and schools lack the scale, resources or expertise to set up a server center on their own that has a LAN-speed network connection to each user. Indeed, a large percentage of schools and businesses have the same Internet connections (e.g., DSL, cable modems) as homes.

Yet such organizations may still have the need for very high-performance computing, either on a regular basis or on a periodic basis. For example, a small architectural firm may have only a small number of architects, with relatively modest computing needs when doing design work, but it may require very high-performance 3D computing periodically (e.g., when creating a 3D fly-through of a new architectural design for a client). The system shown in Figure 4a is extremely well suited for such organizations. The organizations need nothing more than the same sort of network connection that are offered to homes (e.g., DSL, cable modems) and are typically very inexpensive. They can either utilize inexpensive PCs as the client 415 or dispense with PCs altogether and utilize inexpensive dedicated devices which simply implement the control signal logic 413 and low-latency video decompression 412. These features are particularly attractive for schools that may have problems with theft of PCs or damage to the delicate components within PCs.

Such an arrangement solves a number of problems for such organizations (and many of these advantages are also shared by home users doing general-purpose computing). For one, the operating cost (which ultimately must be

passed back in some form to the users in order to have a viable business) can be much lower because (a) the computing resources are shared with other applications that have different peak usage times during the week, (b) the organizations can gain access to (and incur the cost of) high performance computing resources only when needed, (c) the organizations do not have to provide resources for backing up or otherwise maintaining the high performance computing resources.

ELIMINATION OF PIRACY

In addition, games, applications, interactive movies, etc, can no longer be pirated as they are today. Because each game is stored and executed at the hosting service 210, users are not provided with access to the underlying program code, so there is nothing to pirate. Even if a user were to copy the source code, the user would not be able to execute the code on a standard game console or home computer. This opens up markets in places of the world such as China, where standard video gaming is not made available. The re-sale of used games is also not possible because there are no copies of a games distributed to users.

For game developers, there are fewer market discontinuities as is the case today when new generations of game consoles or PCs are introduced to the market. The hosting service 210 can be gradually updated with more advanced computing technology over time as gaming requirements change, in contrast to the current situation where a completely new generation of console or PC technology forces users and developers to upgrade and the game developer is dependent on the timely delivery of the hardware platform to the user (e.g. in the case of the PlayStation 3, its introduction was delayed by more than a year, and developers had to wait until it was available and significant numbers of units were purchased).

STREAMING INTERACTIVE VIDEO

The above descriptions provide a wide range of applications enabled by the novel underlying concept of general Internet-based, low-latency streaming interactive video (which implicitly includes audio together with the video as well, as used herein). Prior art systems that have provided streaming video through the Internet only have enabled applications which can be implemented with high latency interactions. For example, basic playback controls for linear video (e.g. pause, rewind, fast forward) work adequately with high latency, and it is possible to select among linear video feeds. And, as stated previously, the nature of some video games allow them to be played with high latency. But the high latency (or low compression ratio) of prior art approaches for streaming video have severely limited the potential applications of streaming video or narrowed their deployments to specialized network environments, and even in such environments, prior art techniques introduce substantial burdens on the networks. The technology described herein opens the door for the wide range of applications possible with low-latency streaming interactive video through the Internet, particularly those enabled through consumer-grade Internet connections.

Indeed, with client devices as small as client 465 of Figure 4c sufficient to provide an enhanced user experience with an effectively arbitrary amount of computing power, arbitrary amount of fast storage, and extremely fast networking amongst powerful servers, it enables a new era of computing. Further, because the bandwidth requirements do not grow as the computing power of the system grows (i.e., because the bandwidth requirements are only tied to display resolution, quality and frame rate), once broadband Internet connectivity is ubiquitous (e.g., through widespread low-latency wireless coverage), reliable, and of sufficiently high bandwidth to meet the needs of the display devices 422 of all users, the question will be whether thick clients (such as PCs or mobile phones running Windows, Linux,

OSX, etc.,) or even thin clients (such as Adobe Flash or Java) are necessary for typical consumer and business applications.

The advent of streaming interactive video results in a rethinking of assumptions about the structure of computing architectures. An example of this is the hosting service 210 server center embodiment shown in Figure 15. The video path for delay buffer and/or group video 1550 is a feedback loop where the multicasted streaming interactive video output of the app/game servers 1521-1525 is fed back into the app/game servers 1521-1525 either in real-time via path 1552 or after a selectable delay via path 1551. This enables a wide range of practical applications (e.g. such as those illustrated in Figures 16, 17 and 20) that would be either impossible or infeasible through prior art server or local computing architectures. But, as a more general architectural feature, what feedback loop 1550 provides is recursion at the streaming interactive video level, since video can be looped back indefinitely as the application requires it. This enables a wide range of application possibilities never available before.

Another key architectural feature is that the video streams are unidirectional UDP streams. This enables effectively an arbitrary degree of multicasting of streaming interactive video (in contrast, two-way streams, such as TCP/IP streams, would create increasingly more traffic logjams on the networks from the back-and-forth communications as the number of users increased). Multicasting is an important capability within the server center because it allows the system to be responsive to the growing needs of Internet users (and indeed of the world's population) to communicate on a one-to-many, or even a many-to-many basis. Again, the examples discussed herein, such as Figure 16 which illustrates the use of both streaming interactive video recursion and multicasting are just the tip of a very large iceberg of possibilities.

NON-TRANSIT PEERING

In one embodiment, the hosting service 210 has one or more peering connections to one or more Internet Service Providers (ISP) who also provide Internet service to users, and in this way the hosting service 210 may be able to communicate with the user through a non-transit route that stays within that ISP's network. For example, if the hosting service 210 WAN Interface 441 directly connected to Comcast Cable Communications, Inc.'s network, and the user premises 211 was provisioned with broadband service with a Comcast cable modem, a route between the hosting service 210 and client 415 may be established entirely within Comcast's network. The potential advantages of this would include lower cost for the communications (since the IP transit costs between two or more ISP networks might be avoided), a potentially more reliable connection (in case there were congestion or other transit disruptions between ISP networks), and lower latency (in case there were congestion, inefficient routes or other delays between ISP networks).

In this embodiment, when the client 415 initially contacts the hosting service 210 at the beginning of a session, the hosting service 210 receives the IP address of the user premises 211. It then uses available IP address tables, e.g., from ARIN (American Registry for Internet Numbers), to see if the IP address is one allocated to a particular ISP connected to the hosting service 210 that can route to the user premises 211 without IP transit to through another ISP. For example, if the IP address was between 76.21.0.0 and 76.21.127.255, then the IP address is assigned to Comcast Cable Communications, Inc. In this example, if the hosting service 210 maintains connections to Comcast, AT&T and Cox ISPs, then it selects Comcast as the ISP most likely to provide an optimal route to the particular user.

VIDEO COMPRESSION USING FEEDBACK

In one embodiment, feedback is provided from the client device to the hosting service to indicate successful (or unsuccessful) tile and/or frame delivery. The feedback information provided from the client is then used to adjust the video compression operations at the hosting service.

For example, Figures 25a-b illustrate one embodiment of the invention in which a feedback channel 2501 is established between the client device 205 and the hosting service 210. The feedback channel 2501 is used by the client device 205 to send packetized acknowledgements of successfully received tiles/frames and/or indications of unsuccessfully received tiles/frames.

In one embodiment, after successfully receiving each tile/frame, the client transmits an acknowledgement message to the hosting service 210. In this embodiment, the hosting service 210 detects a packet loss if it does not receive an acknowledgement after a specified period of time and/or if it receives an acknowledgement that the client device 205 has received a subsequent tile/frame than one that had been sent. Alternatively, or in addition, the client device 205 may detect the packet loss and transmit an indication of the packet loss to the hosting service 210 along with an indication of the tiles/frames affected by the packet loss. In this embodiment, continuous acknowledgement of successfully delivered tiles/frames is not required.

Regardless of how a packet loss is detected, in the embodiment illustrated in Figures 25a-b, after generating an initial set of I-tiles for an image (not shown in Figure 25a), the encoder subsequently generates only P-tiles until a packet loss is detected. Note that in Figure 25a, each frame, such as 2510 is illustrated as 4 vertical tiles. The frame may be tiled in a different configuration, such as a 2x2, 2x4, 4x4, etc., or the frame may be encoded in its entirety with no tiles (i.e. as 1 large tile). The foregoing examples of frame tiling configurations are provided for the purpose of

illustration of this embodiment of the invention. The underlying principles of the invention are not limited to any particular frame tiling configuration.

Transmitting only P-tiles reduces the bandwidth requirements of the channel for all of the reasons set forth above (i.e., P-tiles are generally smaller than I-tiles). When a packet loss is detected via the feedback channel 2501, new I-tiles are generated by the encoder 2500, as illustrated in Figure 25b, to re-initialize the state of the decoder 2502 on the client device 205. As illustrated, in one embodiment, the I-tiles are spread across multiple encoded frames to limit the bandwidth consumed by each individual encoded frame. For example, in Figure 25, in which each frame includes 4 tiles, a single I-tile is transmitted at a different position within 4 successive encoded frames.

The encoder 2500 may combine the techniques described with respect to this embodiment with other encoding techniques described herein. For example, in addition to generating I-tiles in response to a detected packet loss the encoder 2500 may generate I-tiles in other circumstances in which I-tiles may be beneficial to properly render the sequence of images (such as in response to sudden scene transitions).

Figure 26a illustrates another embodiment of the invention which relies on a feedback channel 2601 between the client device 205 and the hosting service 210. Rather than generating new I-tiles/frames in response to a detected packet loss, the encoder 2600 of this embodiment adjusts the dependencies of the P-tiles/frames. As an initial matter, it should be noted that the specific details set forth in this example are not required for complying with the underlying principles of the invention. For example, while this example will be described using P-tiles/frames, the underlying principles of the invention are not limited to any particular encoding format.

In Figure 26a, the encoder 2600 encodes a plurality of uncompressed tiles/frames 2605 into a plurality of P-tiles/frames 2606 and transmits the P-tiles/frames over a communication channel (e.g., the Internet) to a client device 205. A decoder 2602 on the client device 205 decodes the P-tiles/frames 2606 to generate a plurality of decompressed tiles/frames 2607. The past state(s) 2611 of the encoder 2600 is stored within a memory device 2610 on the hosting service 210 and the past state(s) 2621 of the decoder 2602 is stored within a memory device 2620 on the client device 205. The “state” of a decoder is a well known term of art in video coding systems such as MPEG-2 and MPEG-4. In one embodiment, the past “state” stored within the memories comprises the combined data from prior P-tiles/frames. The memories 2611 and 2621 may be integrated within the encoder 2600 and decoder 2602, respectively, rather than being detached from the encoder 2600 and decoder 2602, as shown in Figure 26a. Moreover, various types of memory may be used including, by way of example and not limitation, random access memory.

In one embodiment, when no packet loss occurs, the encoder 2600 encodes each P-tile/frame to be dependent on the previous P-tile/frame. Thus, as indicated by the notation used in Figure 26a, P-tile/frame 4 is dependent on P-tile/frame 3 (identified using the notation 4_3); P-tile/frame 5 is dependent on P-tile/frame 4 (identified using the notation 5_4); and P-tile/frame 6 is dependent on P-tile/frame 5 (identified using the notation 6_5). In this example, P-tile/frame 4_3 has been lost during transmission between the encoder 2600 and the decoder 2602. The loss may be communicated to the encoder 2600 in various ways including, but not limited to, those described above. For example, each time the decoder 2606 successfully receives and/or decodes a tile/frame, this information may be communicated from the decoder 2602 to the encoder 2600. If the encoder 2600

does not receive an indication that a particular tile/frame has been received and/or decoded after a period of time, then the encoder 2600 will assume that the tile/frame has not been successfully received. Alternatively, or in addition, the decoder 2602 may notify the encoder 2600 when a particular tile/frame is not successfully received.

In one embodiment, regardless of how the lost tile/frame is detected, once it is, the encoder 2600 encodes the next tile/frame using the last tile/frame known to have been successfully received by the decoder 2602. In the example shown in Figure 26a, tiles/frames 5 and 6 are not considered “successfully received” because they cannot be properly decoded by the decoder 2602 due to the loss of tile/frame 4 (i.e., the decoding of tile/frame 5 depends on tile/frame 4 and the decoding of tile/frame 6 depends on tile/frame 5). Thus, in the example shown in Figure 26a, the encoder 2600 encodes tile/frame 7 to be dependent on tile/frame 3 (the last successfully received tile/frame) rather than tile/frame 6 which the decoder 2602 cannot properly decode. Although not illustrated in Figure 26a, tile/frame 8 will subsequently be encoded to be dependent on tile/frame 7 and tile/frame 9 will be encoded to be dependent on tile/frame 8, assuming that no additional packet losses are detected.

As mentioned above, both the encoder 2600 and the decoder 2602 maintain past encoder and decoder states, 2611 and 2621, within memories 2610 and 2620, respectively. Thus, when encoding tile/frame 7, the encoder 2600 retrieves the prior encoder state associated with tile/frame 3 from memory 2610. Similarly, the memory 2620 associated with decoder 2602 stores at least the last known good decoder state (the state associated with P-tile/frame 3 in the example). Consequently, the decoder 2602 retrieves the past state information associated with tile/frame 3 so that tile/frame 7 can be decoded.

As a result of the techniques described above, real-time, low latency, interactive video can be encoded and streamed using relatively small bandwidth because no I-tiles/frames are ever required (except to initialize the decoder and encoder at the start of the stream). Moreover, while the video image produced by the decoder may temporarily include undesirable distortion resulting from lost tile/frame 4 and tiles/frames 5 and 6 (which cannot be properly decoded due to the loss of tile/frame 4), this distortion will be visible for a very short duration. Moreover, if tiles are used (rather than full video frames), the distortion will be limited to a particular region of the rendered video image.

A method according to one embodiment of the invention is illustrated in Figure 26b. At 2650, a tile/frame is generated based on a previously-generated tile/frame. At 2651, a lost tile/frame is detected. In one embodiment, the lost tile/frame is detected based on information communicated from the encoder to the decoder, as described above. At 2652, the next tile/frame is generated based on a tile/frame which is known to have been successfully received and/or decoded at the decoder. In one embodiment, the encoder generates the next tile/frame by loading the state associated with the successfully received and/or decoded tile/frame from memory. Similarly, when the decoder receives the new tile/frame, it decodes the tile/frame by loading the state associated with the successfully received and/or decoded tile/frame from memory.

In one embodiment the next tile/frame is generated based upon the last tile/frame successfully received and/or decoded at the encoder. In another embodiment, the next tile/frame generated is an I tile/frame. In yet another embodiment, the choice of whether to generate the next tile/frame based on a previously successfully received tile/frame, or as an I frame, is based on the how many tile/frames were lost and/or the latency of the channel. In the situation where

a relatively small number (e.g., 1 or 2) tile/frames are lost and the round-trip latency is relatively low (e.g. 1 or 2 frame times), then it may be optimal to generate a P tile/frame since the difference between the last successfully received tile/frame and the newly generated one may be relatively small. If several tile/frames are lost or the round-trip latency is high, then it may be optimal to generate an I tile/frame since the difference between the last successfully received tile/frame and the newly generated one may be large. In one embodiment, a tile/frame loss threshold and/or a latency threshold value is set to determine whether to transmit an I tile/frame or a P tile/frame. If the number of lost tiles/frames is below the tile/frame loss threshold and/or if the round trip latency is below the latency threshold value, then a new I tile/frame is generated; otherwise, a new P tile/frame is generated.

In one embodiment, the encoder always attempts to generate a P tile/frame relative to the last successfully received tile/frame, and if in the encoding process the encoder determines that the P tile/frame will likely be larger than an I tile/frame (e.g. if it has compressed $1/8^{\text{th}}$ of the tile/frame and the compressed size is larger than $1/8^{\text{th}}$ of the size of the average I tile/frame previously compressed), then the encoder will abandon compressing the P tile/frame and will instead compress an I tile/frame.

If lost packets occur infrequently, the systems described above using feedback to report a dropped tile/frame typically results in a very slight disruption in the video stream to the user because a tile/frame that was disrupted by a lost packet is replaced in roughly the time of one round trip between the client device 205 and hosting service 210 assuming the encoder 2600 compresses the tile/frame in a short amount of time. And, because the new tile/frame that is compressed is based upon a later frame in the uncompressed video stream, the video stream does not fall behind the uncompressed video stream. But, if a packet containing the new tile/frame also

is lost, then this results in a delay of at least two round trips to yet again request and send another new tile/frame, which in many practical situations will result in a noticeable disruption to the video stream. As a consequence, it is very important that the newly-encoded tile/frame sent after dropped tile/frame is successfully sent from the hosting service 210 to the client device 205.

In one embodiment, forward-error correction (FEC) coding techniques, such as those previously described and illustrated in Figures 11a, 11b, 11c and 11d, are used to mitigate the probability of losing the newly-encoded tile/frame. If FEC coding is already being used when transmitting tiles/frames, then a stronger FEC code is used for the newly-encoded tile/frame.

One potential cause of dropped packets is a sudden loss in channel bandwidth, for example, if some other user of the broadband connection at the user premises 211 starts using a large amount of bandwidth. If a newly-generated tile/frame also is lost due to dropped packets (even if FEC is used), then in one embodiment when hosting service 210 is notified by client 415 that a second newly encoded tile/frame is dropped, video compressor 404 reduces the data rate when it encodes a subsequent newly encoded tile/frame. Different embodiments reduce the data rate using different techniques. For example, in one embodiment, this data rate reduction is accomplished by lowering the quality of the encoded tile/frame by increasing the compression ratio. In another embodiment, the data rate is reduced by lowering the frame rate of the video (e.g. from 60fps to 30fps) and accordingly slowing the rate of data transmission. In one embodiment, both techniques for reducing the data rate are used (e.g., both reducing the frame rate and increasing the compression ratio). If this lower rate of data transmission is successful at mitigating the dropped packets, then in accordance the channel data rate detection and adjustment methods previously described, the hosting service 210 will continue

encoding at a lower data rate, and then gradually adjust the data rate upward or downward as the channel will allow. The continuous receipt of feedback data related to dropped packets and/or latency allow the hosting service 210 to dynamically adjust the data rate based on current channel conditions.

STATE MANAGEMENT IN AN ONLINE GAMING SYSTEM

One embodiment of the invention employs techniques to efficiently store and port the current state of an active game between servers. While the embodiments described herein are related to online gaming, the underlying principles of the invention may be used for various other types of applications (e.g., design applications, word processors, communication software such as email or instant messaging, etc). Figure 27a illustrates an example system architecture for implementing this embodiment and Figure 27b illustrates an example method. While the method and system architecture will be described concurrently, the method illustrated in Figure 27b is not limited to any particular system architecture.

At 2751 of Figure 27b, a user initiates a new online game on a hosting service 210a from a client device 205. In response, at 2752, a “clean” image of the game 2702a is loaded from storage (e.g., a hard drive, whether connected directly to a server executing the game, or connected to a server through a network) to memory (e.g., RAM) on the hosting service 210a. The “clean” image comprises the runtime program code and data for the game prior to the initiation of any game play (e.g., as when the game is executed for the first time). The user then plays the game at 2753, causing the “clean” image to change to a non-clean image (e.g., an executing game represented by “State A” in Figure 27a). At 2754, the game is paused or terminated, either by the user or the hosting service 210a. At 2755, state management logic 2700a on the hosting service 210a determines the differences between the “clean” image of the game and the current game state (“State A”).

Various known techniques may be used to calculate the difference between two binary images including, for example, those used in the well known “diff” utility available on the UNIX operating system. Of course, the underlying principles of the invention are not limited to any particular techniques for difference calculation.

Regardless of how the differences are calculated, once they are, the difference data is stored locally within a storage device 2705a and/or transmitted to a different hosting service 210b. If transmitted to a different hosting service 210b, the difference data may be stored on a storage device (not shown) at the new hosting service 210b. In either case, the difference data is associated with the user’s account on the hosting services so that it may be identified the next time the user logs in to the hosting services and initiates the game. In one embodiment, rather than being transmitted immediately, the difference data is not transmitted to a new hosting service until the next time the user attempts to play the game (and a different hosting service is identified as the best choice for hosting the game).

Returning to the method shown in Figure 27b, at 2757, the user reinitiates the game from a client device, which may be the same client device 205 from which the user initially played the game or a different client device (not shown). In response, at 2758, state management logic 2700b on the hosting service 210b retrieves the “clean” image of the game from a storage device and the difference data. At 2759, the state management logic 2700b combines the clean image and difference data to reconstruct the state that the game was in on the original hosting service 210a (“State A”). Various known techniques may be used to recreate the state of a binary image using the difference data including, for example, those used in the well known “patch” utility available on the UNIX operating system. The difference calculation techniques used in well known backup programs such as PC

Backup may also be used. The underlying principles of the invention are not limited to any particular techniques for using difference data to recreate a binary image.

In addition, at 2760, platform-dependent data 2710 is incorporated into the final game image 2701b. The platform-dependent data 2710 may include any data which is unique to the destination server platform. By way of example, and not limitation, the platform-dependent data 2710 may include the Medium Access Control (MAC) address of the new platform, the TCP/IP address, the time of day, hardware serial numbers (e.g., for the hard drive and CPU), network server addresses (e.g., DHCP/Wins servers), and software serial number(s) / activation code(s) (including Operating System serial number(s) / activation code(s)).

Other platform-dependent data related to the client/user may include (but is not limited to) the following:

1. The user's screen resolution. When the user resumes the game, the user may be using a different device with a different resolution.

2. The user's controller configuration. When game resumes, the user may have switched from a game controller to a keyboard/mouse.

3. User entitlements, such as whether a discount rate has expired (e.g., if the user was playing the game during a promotional period and is now playing during a normal period at higher cost) or whether the user or device has certain age restrictions (e.g., the parents of the user may have changed the settings for a child so the child is not allowed to see mature material, or if the device playing the game (e.g., a computer at a public library) has certain restrictions on whether mature material can be displayed).

4. The user's ranking. The user may have been allowed to play a multiplayer game in a certain league, but because some other users had exceeded the user's ranking, the user may have been downgraded to a lesser league.

The foregoing examples of platform-dependent data 2710 are provided for the purpose of illustration of this embodiment of the invention. The underlying principles of the invention are not limited to any particular set of platform-dependent data.

Figure 28 graphically illustrates how the state management logic 2700a at the first hosting service extracts difference data 2800 from the executing game 2701a. The state management logic 2700b at the second hosting service then combines the clean image 2702b with the difference data 2800 and platform-dependent data 2710 to regenerate the state of the executing game 2701b. As shown generally in Figure 28, the size of the difference data is significantly smaller than the size of the entire game image 2701a and, consequently, a significant amount of storage space and bandwidth is conserved by storing/transmitting only difference data. Although not shown in Figure 28, the platform-dependent data 2700 may overwrite some of the difference data when it is incorporated into the final game image 2701b.

While an online video gaming implementation is described above, the underlying principles of the invention are not limited to video games. For example, the foregoing state management techniques may be implemented within the context of any type of online-hosted application.

TECHNIQUES FOR MAINTAINING A CLIENT DECODER

In one embodiment of the invention, the hosting service 210 transmits a new decoder to the client device 205 each time the user requests connect to hosting service 210. Consequently, in this embodiment, the decoder used by the client device is always up-to-date and uniquely tailored to the hardware/software implemented on the client device.

As illustrated in Figure 29, in this embodiment, the application which is permanently installed on the client device 205 does not include a decoder. Rather, it is a client downloader application 2903 which manages the download and installation of a temporary decoder 2900 each time the client device 205 connects to the hosting service 210. The downloader application 2903 may be implemented in hardware, software, firmware, or any combination thereof. In response to a user request for a new online session, the downloader application 2903 transmits information related to the client device 205 over a network (e.g., the Internet). The information may include identification data identifying the client device and/or the client device's hardware/software configuration (e.g., processor, operating system, etc).

Based on this information, a downloader application 2901 on the hosting service 210 selects an appropriate temporary decoder 2900 to be used on the client device 205. The downloader application 2901 on the hosting service then transmits the temporary decoder 2900 and the downloader application 2903 on the client device verifies and/or installs the decoder on the client device 205. The encoder 2902 then encodes the audio/video content using any of the techniques described herein and transmits the content 2910 to the decoder 2900. Once the new decoder 2900 is installed, it decodes the content for the current online session (i.e., using one or more of the audio/video decompression techniques described herein). In one embodiment, when the session is terminated, the decoder 2900 is removed (e.g., uninstalled) from the client device 205.

In one embodiment the downloader application 2903 characterizes the channel as the temporary decoder 2900 is being downloaded by making channel assessments such as the data rate achievable on the channel (e.g. by determining how long it takes for data to download), the packet loss rate on the channel, and the

latency of the channel. The downloader application 2903 generates channel characterization data describing the channel assessments. This channel characterization data is then transmitted from the client device 205 to the hosting service downloader 2901, which uses the channel characterization data to determine how best to utilize the channel to transmit media to the client device 205.

The client device 205 typically will send back messages to the hosting service 205 during the downloading of the temporary decoder 2900. These messages can include acknowledgement messages indicating whether packets were received without errors or with errors. In addition, the messages provide feedback to the downloader 2901 as to the data rate (calculated based on the rate at which packets are received), the packet error rate (based on the percentage of packets reported received with errors), and the round-trip latency of the channel (based on the amount of time that it takes before the downloader 2901 receives feedback about a given packet that has been transmitted).

By way of example, if the data rate is determined to be 2 Mbps, then the downloader may choose a smaller video window resolution for the encoder 2902 (e.g. 640x480 at 60fps) than if the data rate is determined to be 5 Mbps (e.g. 1280x720 at 60fps). Different forward error correction (FEC) or packet structures may be chosen, depending on the packet loss rate.

If the packet loss is very low, then the compressed audio and video may be transmitted without any error correction. If the packet loss is medium, then the compressed audio and video may be transmitted with error correction coding techniques (e.g., such as those previously described and illustrated in Figures 11a, 11b, 11c and 11d). If the packet loss is very high, it may be determined that an audiovisual stream of adequate quality cannot be transmitted, and the client device 205 may either notify the user that the hosting service is not available through the

communications channel (i.e. the "link"), or it may try to establish a different route to the hosting service that has a lower packet loss (as described below).

If the latency is low, then the compressed audio and video can be transmitted with low latency and a session can be established. If the latency is too high (e.g. higher than 80ms) then, for games which require low latency, the client device 205 may either notify the user that the hosting service is not available through the link, that a link is available but the response time to user input will be sluggish or "laggy," or that the user can try to establish a different route to the hosting service that has a lower latency (as described below).

The Client Device 205 may try to connect to the Hosting Service 210 through another route through the network (e.g. the Internet) to see if impairments are reduced (e.g. the packet loss is lower, the latency is lower, or even if the data rate is higher). For example, the Hosting Service 210 may connect to the Internet from multiple locations geographically (e.g., a hosting center in Los Angeles and one in Denver), and perhaps there is high packet loss due to congestion in Los Angeles, but there is not congestion in Denver. Also, the Hosting Service 210 may connect to the Internet through multiple Internet service providers (e.g. AT&T and Comcast).

Because of congestion or other issues between the client device 205 and one of the service providers (e.g. AT&T), packet loss and/or high latency and/or constrained data rate may result. However, if the Client Device 205 connects to the hosting service 210 through another service provider (e.g., Comcast), it may be able to connect without congestion problems and/or lower packet loss and/or lower latency and/or higher data rate. Thus, if the client device 205 experiences packet loss above a specified threshold (e.g., a specified number of dropped packets over a specified duration), latency above a specified threshold and/or a data rate below a specified threshold while downloading the temporary decoder 2900, in one

embodiment, it attempts to reconnect to the hosting service 210 through an alternate route (typically by connecting to a different IP address or different domain name) to determine if a better connection can be obtained.

If the connection is still experiencing unacceptable impairments after alternative connection options are exhausted, then it could be that the client device 205's local connection to the Internet is suffering from impairments, or that it is too far away from the hosting service 210 to achieve adequate latency. In such a case the client device 205 may notify the user that the Hosting Service is not available through the link or that it is only available with impairments, and/or the only certain types of low-latency games/applications are available.

Since this assessment and potential improvement of the link characteristics between the Hosting Service 210 and the Client Device 205 occurs while the temporary decoder is being downloaded, it reduces the amount of time that the client device 205 would need to spend separately downloading the temporary decoder 2900 and assessing the link characteristics. Nonetheless, in another embodiment, the assessment and potential improvement of the link characteristics is performed by the client device 205 separately from downloading the temporary decoder 2900 (e.g., by using dummy test data rather than the decoder program code). There are number of reasons why this may be a preferable implementation. For example, in some embodiments, the client device 205 is implemented partially or entirely in hardware. Thus, for these embodiments, there is no software decoder per se necessary to download.

COMPRESSION USING STANDARDS-BASED TILE SIZES

As mentioned above, when tile-based compression is used, the underlying principles of the invention are not limited to any particular tile size, shape, or orientation. For example, in a DCT-based compression system such as MPEG-2

and MPEG-4, tiles may be the size of macroblocks (components used in video compression which typically represent a block of 16 by 16 pixels). This embodiment provides a very fine level of granularity for working with tiles.

Moreover, regardless of tile size, various types of tiling patterns may be used. For example, Figure 30 illustrates an embodiment in which multiple I-tiles are used in each R frame 3001-3004. A rotating pattern is used in which I-tiles are dispersed throughout each R frame so that a full I-frame is generated every four R frames. Dispersing the I-tiles in this manner will reduce the effects of a packet loss (limiting the loss to a small region of the display).

The tiles may also be sized to an integral native structure of the underlying compression algorithm. For example, if the H.264 compression algorithm is used, in one embodiment, tiles are set to be the size of H.264 "slices." This allows the techniques described herein to be easily integrated the context of various different standard compression algorithms such as H.264 and MPEG-4. Once the tile size is set to a native compression structure, the same techniques as those described above may be implemented.

TECHNIQUES FOR STREAM REWIND AND PLAYBACK OPERATIONS

As previously described in connection with Figure 15, the uncompressed video/audio stream 1529 generated by an app/game server 1521-1525 may be compressed by shared hardware compression 1530 at multiple resolutions simultaneously resulting in multiple compressed video/audio streams 1539. For example, a video/audio stream generated by app/game server 1521 may be compressed at 1280x720x60fps by the shared hardware compression 1530 and transmitted to a user via outbound routing 1540 as outbound Internet traffic 1599. That same video/audio stream may be simultaneously scaled down to thumbnail size (e.g. 200x113) by the shared hardware compression 1530 via path 1552 (or

through delay buffer 1515) to app/game server 1522 to be displayed as one thumbnail 1600 of a collection of thumbnails in Figure 16. When thumbnail 1600 is zoomed through intermediate size 1700 in Figure 17 to size 1800 (1280x720x60fps) in Figure 18, then rather than decompressing the thumbnail stream, app/game server 1522 can decompress a copy of the 1280x720x60fps stream being sent to the user of app/game server 1521, and scale the higher resolution video as it is zoomed from thumbnail size to 1280x720 size. This approach has the advantage of reutilizing the 1280x720 compressed stream twice. But it has several disadvantages: (a) the compressed video stream sent to the user may vary in image quality if the data throughput of the user's Internet connection varies resulting in a varying image quality viewed by the "spectating" user of app/game server 1522, even if that user's Internet connection does not vary, (b) app/game server 1522 will have to use processing resources to decompress the entire 1280x720 image and then scale that image (and likely apply a re-sampling filter) to display much smaller sizes (e.g. 640x360) during the zoom, (c) if frames are dropped due to limited Internet connection bandwidth and/or lost/corrupted packets, and the spectating user "rewinds" and "pauses" the video recorded in the delay buffer 1515, the spectating user will find the dropped frames are missing in the delay buffer (this will be particularly apparent if the user "steps" frame-by-frame), and (d) if the spectating user rewinds to find a particular frame in the video recorded in the delay buffer, then the app/game server 1522 will have to find an I frame or I tiles prior to the sought frame in the video stream recorded in the delay buffer, and then decompress all of the P frames/tiles until the desired frame is reached. This same limitations would not only apply to users "spectating" the video/audio stream live, but users (including the user that generated the video/audio stream) viewing an archived (e.g. "Brag Clip") copy of the video/audio stream.

An alternative embodiment of the invention addresses these issues by compressing the video stream in more than one size and/or structure. One stream (the "Live" stream) is compressed optimally to stream to the end user, as described herein, based on the characteristics of the network connection (e.g. data bandwidth, packet reliability) and the user's local client capabilities (e.g., decompression capability, display resolution). Other streams (referred to herein as "HQ" streams) are compressed at high quality, at one or more resolutions, and in a structure amenable to video playback, and such HQ streams are routed and stored within the server center 210. For example, in one embodiment, the HQ compressed streams are stored on a RAID disk array 1515 and are used to provide functions such as pause, rewind, and other playback functions (e.g., "Brag Clips" which may be distributed to other users for viewing).

As illustrated in Figure 31a, one embodiment of the invention comprises an encoder 3100 capable of compressing a video stream in at least two formats: one which periodically includes I-Tiles or I-Frames 3110 and one which does not include I-Tiles or I-Frames 3111, unless necessary due to a disruption of the stream or because an I-Tile or I-Frame is determined to likely be smaller than an I-Tile or I-Frame (as described above). For example, the "Live" stream 3111 transmitted to the user while playing a video game may be compressed using only P-Frames (unless I-Tiles or I-Frames are necessary or smaller as described above). In addition, the encoder 3100 of this embodiment concurrently compresses the Live video stream 3111 in a second format which, in one embodiment, periodically includes I-Tiles or I-Frames (or similar type of image format).

While the embodiments described above employ I-Tiles, I-Frames, P-Tiles and P-Frames, the underlying principles of the invention are not limited to any particular compression algorithm. For example, any type of image format in which

frames are dependent on previous or subsequent frames may be used in place of P-Tiles or P-Frames. Similarly, any type of image format which is not dependent on previous or subsequent frames may be substituted in place of the I-Tiles or I-Frames described above.

As mentioned above, the HQ Stream 3110 includes periodic I-Frames (e.g., in one embodiment, every 12 frames or so). This is significant because if the user ever wants to quickly rewind the stored video stream to a particular point, I-Tiles or I-Frames are required. With a compressed stream of only P-Frames (i.e. without the first frame of the sequence being an I-Frame), it would be necessary for the decoder go back to the first frame of the sequence (which might be hours long) and decompress P frames up to the point to which the user wants to rewind. With an I-Frame every 12 frames stored in the HQ stream 3110, the user can decide to rewind to a particular spot and the nearest preceding I-Frame of the HQ stream is no more than 12 frames prior to the desired frame. Even if the decoder maximum decode rate is real-time (e.g. $1/60^{\text{th}}$ of a second for a 60 frame/sec stream), then $12 \text{ (frames)} / 60 \text{ (frames/sec)} = 1/5$ second away from an I-Frame. And, in many cases, decoders can operate much faster than real-time so, for example, at 2x real-time a decoder could decode 12 frames in 6 frames, which is just $1/10^{\text{th}}$ of a second delay for a "rewind". Needless to say, that even a fast decoder (e.g. 10x real-time) would have an unacceptable delay if the nearest preceding I-Frame were a large number of frames previous to a rewind point (e.g. it would take $1 \text{ hour} / 10 = 6$ minutes to do a "rewind"). In another embodiment, periodic I-Tiles are used, and in this case when the user seeks to rewind the decoder will find the nearest preceding I-Tile prior to the rewind point, and then commence decoding of that tile from that point until all tiles are decoded through to the rewind point. Although periodic I-Tiles or I-Frames result in less efficient compression than eliminating I-Frames entirely, the hosting service

210 typically has more than enough locally available bandwidth and storage capacity to manage the HQ stream.

In another embodiment the encoder 3100 encodes the HQ stream with periodic I-Tile or I-Frames, followed by P-Tiles or P-Frames, as previously described, but also preceded by B-Tiles or B-Frames. B-Frames, as described previously are frames that precede an I-Frame and are based on frame differences from the I-Frame working backwards in time. B-Tiles are the tile counterpart, preceding an I-Tile and based on frame differences working backwards from the I-Tile. In this embodiment, if the desired rewind point is a B-Frame (or contains B-Tiles), then the decoder will find the nearest *succeeding* I-Frame or I-Tile and decode backwards in time until the desired rewind point is decoded, and then as video playback proceeds from that point forward, the decoder will decode B-Frames, I-Frames and P-Frames (or their tile counterparts) in successive frames going forward. An advantage of employing B-Frames or B-Tiles in addition to I and P types is that, often higher quality at a given compression ratio can be achieved.

In yet another embodiment, the encoder 3100 encodes the HQ stream as all I-Frames. An advantage of this approach is that every rewind point is an I-Frame, and as a result, no other frames need to be decoded in order to reach the rewind point. A disadvantage is the compressed data rate will be very high compared to I, P or I, P, B stream encoding.

Other video stream playback actions (e.g. fast or slow rewind, fast or slow forward, etc.), typically are much more practically accomplished with periodic I-Frames or I-Tiles (alone or combined with P and/or B counterparts), since in each case the stream is played back in a different frame order than frame-by-frame forward in time, and as a result, the decoder needs to find and decode a particular, often arbitrary, frame in the sequence. For example, in the case of very fast-forward

(e.g. 100x speed), each successive frame displayed is 100 frames after the prior frame. Even with a decoder that runs at 10x real-time and decodes 10 frames in 1 frame time, it would still be 10x too slow to achieve 100x fast-forward. Whereas, with periodic I-Frames or I-Tiles as described above, the decoder is able to seek the nearest applicable I-Frame or I-Tiles to the frame it needs to display next and only decode the intervening frames or tiles to the point of the target frame.

In another embodiment I-Frames are encoded in the HQ stream at a consistent periodicity (e.g. always each 8 frames) and the speed multipliers made available to the user for fast forward and rewind that are faster than the I-Frame rate are exact multiples of the I-Frame periodicity. For example, if the I-Frame periodicity is 8 frames, then the fast-forward or rewind speeds made available to the user might be 1X, 2X, 3X, 4X, 8X, 16X, 64X and 128X and 256X. For speeds faster than the I-Frame periodicity, the decoder will first jump ahead to the closest I-Frame that is the number of frames ahead at the speed (e.g., if the currently displayed frame is 3 frames prior to an I-Frame, then at 128X, the decoder would jump to a frame $128+3$ frames ahead), and then for each successive frame the decoder would jump the exact number of frames as the chosen speed (e.g. at the chosen speed of 128X, the decoder would jump 128 frames) which would land exactly on an I-Frame each time. Thus, given that all speeds faster than the I-Frame periodicity are exact multiples of the I-Frame periodicity, the decoder will never need to decode any preceding or following frames to seek the desired frame, and only will have to decode one I-Frame per displayed frame. For speeds slower than the I-Frame periodicity (e.g. 1X, 2X, 3X, 4X), or for speeds faster that are non-multiples of the I-Frame periodicity, for each frame displayed, the decoder seeks whichever frames require the least additional newly decoded frames to display the desired frame, be it an undecoded I-Frame or an already-decoded frame still available in decoded form (in RAM or other

fast storage), and then decode intervening frames, as necessary, until the desired frame is decoded and displayed. For example, at 4X fast forward, in an I,P encoded sequence with 8X I-Frame periodicity, if the current frame is a P-frame that is 1 frame following an I-frame, then the desired frame to be displayed is 4 frames later, which would be the 5th P-Frame following the preceding I-frame. If the currently displayed frame (which had just been decoded) is used as a starting point, the decoder will need to decode 4 more P-frames to display the desired frame. if the preceding I-Frame is used, the decoder will need to decode 6 frames (the I-Frame and the succeeding 5 P-Frames) in order to display the desired frame. (Clearly, in this case, it is advantageous to use the currently displayed frame to minimize the additional frames to decode.). Then, the next frame to be decoded, 4 frames ahead, would be the 1st P-Frame following an I-Frame. In this case, if the currently decoded frame were used as a starting point, the decoder would need to decode 4 more frames (2 P-Frames, an I-Frame and a P-Frame). But, if the next I-Frame were used instead, the decoder would only need to decode the I-Frame and the successive P-Frame. (Clearly, in this case, it is advantageous to use the next I-Frame as a starting point to minimize the additional frames to decode.) Thus, in this example, the decoder would alternate between using the currently decoded frame as a starting point and using a subsequent I-Frame as a starting point. As a general principal, regardless of the HQ video stream playback mode (fast-forward, rewind or step) and speed, the decoder would start with whichever frame, be it an I-Frame or a previously decoded frame, requires the least number of newly decoded frames to display the desired frame for each successive frame displayed for that playback mode and speed.

As illustrated in Figure 31b, one embodiment of the hosting service 210 includes stream replay logic 3112 for managing user requests to replay the HQ

stream 3110. The stream replay logic 3112 receives client requests containing video playback commands (e.g., pause, rewind, playback from a specified point, etc), interprets the commands, and decodes the HQ stream 3110 from the specified point (either starting with either an I-Frame or previously decoded frame, as appropriate, and then proceeding forward or backward to the specified point). In one embodiment, a decoded HQ stream is provided to an encoder 3100 (potentially the self-same encoder 3100, if capable of encoding more than one stream at once, or a separate encoder 3100) so that it may be recompressed (using the techniques described herein) and transmitted to the client device 205. The decoder 3102 on the client device then decodes and renders the stream as described above.

In one embodiment, the stream replay logic 3112 does not decode the HQ stream and then cause the encoder 3100 to re-encode the stream. Rather, it simply streams the HQ stream 3110 directly to the client device 205 from the specified point. The decoder 3102 on the client device 205 then decodes the HQ stream. Because the playback functions described herein do not typically have the same low-latency requirements as playing a real-time video game (e.g. if the player is simply reviewing prior gameplay; not actively playing), the added latency typically inherent in the usually higher-quality HQ stream may result in an acceptable end user experience (e.g., with higher latency but higher-quality video).

By way of example, and not limitation, if the user is playing a video game, the encoder 3100 is providing a Live stream of essentially all P-frames optimized for the user's connection and local client (e.g., approximately 1.4Mbps at a 640 x 360 resolution). At the same time, the encoder 3100 is also compressing the video stream as an HQ stream 3110 within the hosting service 310 and storing the HQ stream on a local Digital Video Decoder RAID array at, for example, 1280 x 720 at 10 Mbps with I-frames every 12 frames. If the user hits a "Pause" button, then

game will be paused on the client's last decoded frame and the screen will freeze. Then if the user hits a "Rewind" button, the stream replay logic 3112 will read the HQ stream 3110 from the DVR RAID starting from the closest I-frame or available already-decoded frame, as described above. The stream replay logic 3112 will decompress the intervening P or B frames, as necessary, re-sequence the frames as necessary so that the playback sequence is backwards at the desired rewind speed, and then resize (using prior art image scaling techniques well-known in the art) the desired decoded intended to be displayed from 1280 x 720 to 640 x 360, and the Live stream encoder 3100 will re-compress the re-sequenced stream at 640 x360 resolution and transmit it to the user. If the user pauses again, and then single-steps through the video to watch a sequence closely, the HQ stream 3110 on the DVR RAID will have every frame available for single stepping (even though the original Live stream may have dropped frames for any of the many reasons described herein). Further, the quality of the video playback will be quite high at every point in the HQ stream, whereas there may be points in the Live stream where, for example, the bandwidth had been impaired, resulting in a temporary reduction in compressed image quality. While impaired image quality for a brief period of time, or in a moving image, may be acceptable for the user, if the user stops at a particular frame (or single-steps slowly) and studies frames closely, impaired quality may not be acceptable. The user is also provided with the ability to fast forward, or jump to a particular spot, by specifying a point within the HQ stream (e.g., 2 minutes prior). All of these operations would be impractical in their full generality and at high quality with a Live video stream that was P-frame-only or rarely (or unpredictably) had I-Frames.

In one embodiment, the user is provided with a video window (not shown) such as a Apple QuickTime or Adobe Flash video window with a "scrubber"

(i.e., a left-right slider control) that allows the user to sweep forward and backward through the video stream, as far back as the HQ stream has stored the video. Although it appears to the user as if he or she is “scrubbing” through the Live stream, in fact he or she is scrubbing through the stored HQ stream 3110, which is then resized and recompressed as a Live stream. In addition, as previously mentioned, if the HQ stream is watched by anyone else at the same time, or the user at a different time, it can be watched a higher (or lower) resolution than the Live stream’s resolution while the HQ stream is simultaneously encoded, and the quality will be as high as the quality of the *viewer’s* Live stream, potentially up to the quality of the HQ stream.

Thus, by simultaneously encoding both the Live stream (as described herein in an appropriate manner for its low-latency, bandwidth and packet error-tolerance requirements) and an HQ stream with its high-quality, stream playback action requirements, the user is thereby provided with desired configuration of both scenarios. And, in fact, it is effectively transparent to the user that there are two different streams being encoded differently. From the user’s perspective, the experience is highly responsive with low-latency, despite running on a highly variable and relatively low bandwidth Internet connection, yet the Digital Video Recording (DVR) functionality is very high quality, with flexible actions and flexible speeds.

As a result of the techniques described above, the user receives the benefits of both Live and HQ video streams during online game play, or other online interaction, without suffering from any of the limitations of either a Live stream or an HQ stream.

Figure 31c illustrates one embodiment of a system architecture for performing the above operations. As illustrated, in this embodiment, the encoder

3100 encodes a series of "Live" streams 3121L, 3122L, and 3125L and a corresponding series of "HQ" streams 3121H1-H3, 3122H1-H3, and 3125H1-H3, respectively. Each HQ stream H1 is encoded at full resolution, while each encoder H2, and H3 scales to the video stream to a smaller size prior to encoding. For example, if the video stream were 1280x720 resolution, H1 would encode at 1280x720 resolution, while H2 could scale to 640x360 and encode at that resolution and H3 could scale to 320x180 and encode at that resolution. Any number of simultaneous H_n scaler/encoders, where n is an integer greater than 1, could be used, providing multiple simultaneous HQ encodings at a variety of resolutions.

Each of the Live streams operate in response to channel feedback signals 3161, 3162, and 3165 received via an inbound Internet connection 3101, as described above (see, e.g., the discussion of feedback signals 2501 and 2601 in Figures 25-26). The Live streams are transmitted out over the Internet (or other network) via outbound routing logic 3140. The Live compressors 3121L-3125L include logic for adapting the compressed video streams (including scaling, dropping frames, etc.) based on channel feedback.

The HQ streams are routed by inbound routing logic 3141 and 1502 to internal delay buffers (e.g., RAID array 3115) or other data storage devices via signal path 3151 and/or are fed back via signal path 3152 into app/game servers and encoder 3100 for additional processing. As described above, the HQ streams 3121H n -3125 H n are subsequently streamed to end users upon request (see, e.g., Figure 31b and associated text).

In one embodiment, the encoder 3100 is implemented with the shared hardware compression logic 1530 shown in Figure 15. In another embodiment, some or all of the encoders and scalers are individual subsystems. The underlying

principles of the invention are not limited to any particular sharing of scaling or compression resources or hardware/software configuration.

An advantage of the configuration of Figure 31c is that App/Game Servers 3121-3125 that require smaller than full-size video windows will not need to process and decompress a full-size window. Also, App/Game Services 3121-3125 that require in-between window sizes can receive a compressed stream that is near the desired window size, and then scale up or down to the desired window size. Also, if multiple App/Game Servers 3121-3125 request the same size video stream from another App/Game Server 3121-3125, Inbound Routing 3141 can implement IP multicast techniques, such as those well-known in the art, and broadcast the requested stream to multiple App/Game Servers 3121-3125 at once, without requiring an independent stream to each App/Game Server making a request. If an App/Game server receiving a broadcast changes the size of a video window, it can switch over to the broadcast of a different video size. Thus, an arbitrarily large number of users can simultaneously view a App/Game Server video stream, each with the flexibility of scaling their video windows and always getting the benefit of a video stream scaled closely to the desired window size.

One disadvantage with the approach shown in Figure 31c is that in many practical implementations of the Hosting Service 210, there is never a time when all of the compressed HQ streams, let alone all of the sizes of all of the compressed HQ streams, are viewed at once. When encoder 3100 is implemented as a shared resource (e.g. a scaler/compressor, either implemented in software or hardware), this wastefulness is mitigated. But, there may be practical issues in connecting a large number of uncompressed streams to a common shared resource, due to the bandwidth involved. For example, each 1080p60 stream is almost 3 Gbps,

which is far in excess of even Gigabit Ethernet. The following alternate embodiments address this issue.

Figure 31d shows an alternative embodiment of the Hosting Service 210 in which each App/Game Server 3121-3125 has two compressors allocated to it: (1) a Live stream compressor 3121L-3125L, that adapts the compressed video stream based on Channel Feedback 3161-3165, and (2) an HQ stream compressor that outputs a full-resolution HQ stream, as described above. Notably, the Live compressor is dynamic and adaptive, utilizing two-way communications with the client 205, while the HQ stream is non-adaptive and one-way. Other differences between the streams are the Live stream quality may vary dramatically, depending on the channel conditions and the nature of the video material. Some frames may be of poor quality, and there may be dropped frames. Also, the Live stream may be almost entirely P-frames or P-tiles, with I-frames or I-tiles appearing infrequently. The HQ stream typically will be much higher data rate than the Live Stream, and it will provide consistent high-quality, without dropping any frames. The HQ stream may be all I-frames, or may have frequent and/or regular I-frames or I-tiles. The HQ stream may also include B-frames or B-tiles.

In one embodiment, Shared video scaling and recompression 3142 (detailed below) selects only certain HQ video streams 3121H1-3125H1 to be scaled and recompressed at one or more different resolutions, before sent to Inbound Routing 3141 for routing as previously described. The other HQ video streams are either passed through at their full size to Inbound Routing 3141 for routing as previously described, or not passed through at all. In one embodiment, the decision on which HQ streams are scaled and recompressed and/or which HQ streams are passed through at all is determined based on whether there is a App/Game Server 3121-3125 that is requesting that particular HQ stream at the particular resolution (or

a resolution close to the scaled or full resolution). Through this means, the only HQ streams that are scaled and recompressed (or potentially passed through at all) are HQ streams that are actually needed. In many applications of Hosting Service 210, this results in a dramatic reduction of scaling and compression resources. Also, given that every HQ stream is at least compressed at its full resolution by compressors 3121H1-3125H1, the bandwidth needed to be routed to and within Shared video scaling and recompression 3142 is dramatically reduced than if it would be accepted uncompressed video. For example, a 3GBps uncompressed 1080p60 stream could be compressed to 10Mbps and still retain very high quality. Thus, with Gigabit Ethernet connectivity, rather than be unable to carry even one uncompressed 3GBps video stream, it would be possible to carry dozens of 10Mbps video streams, with little apparent reduction in quality.

Figure 31f shows details of Shared video scaling and recompression 3142, along with a larger number of HQ video compressors HQ 3121H1-3131H1. Internal routing 3192, per requests for particular video streams scaled to particular sizes from the App/Game Servers 3121-3125, selects typically a subset of compressed HQ streams from HQ video compressors HQ 3121H1-3131H1. A stream within this selected subset of streams is routed either through a Decompressor 3161-3164 if the stream requested is to be scaled, or routed on Non-scaled Video path 3196 if the stream requested is at full resolution. The streams to be scaled are decompressed to uncompressed video by Decompressors 3161-3164, then each scaled to the requested size by Scalers 3171-3174, then each compressed by Compressor 3181-3184. Note that if a particular HQ stream is requested at more than one resolution, then Internal Routing 3192 multicasts that stream (using IP multicasting technology that is well-known by practitioners in the art) to one or more Decompressors 3161-3164 and (if one a requested size if full

resolution) to Outbound Routing 3193. All of the requested streams, whether scaled (from Compressors 3181-3184) or not (from Internal Routing 3192), are then sent to Outbound Routing 3193. Routing 3193 then sends each requested stream to the App/Game Server 3121-3125 that requested it. In one embodiment, if more than one App/Game server requests the same stream at the same resolution, then Outbound Routing 3193 multicasts the stream to all of the App/Game servers 3121-3125 that are making the request.

In the presently preferred embodiment of the Shared video scaling and recompression 3142, the routing is implemented using Gigabit Ethernet switches, and the decompression, scaling, and compression is implemented by discrete specialized semiconductor devices implementing each function. The same functionality could be implemented with a higher level of integration in hardware or by very fast processors.

Figure 31e shows another embodiment of Hosting Service 210, where the function of Delay Buffer 3115, previously described, is implemented in a Shared video delay buffer, scaling and decompression subsystem 3143. The details of subsystem 3143 are shown in Figure 31g. The operation of subsystem 3143 is similar to that of subsystem 3142 shown in Figure 31f, except 3191 first selects which HQ video streams are to be routed, per requests from App/Game Servers 3121-3125, and then, the HQ streams that are requested to be delayed are routed through Delay Buffer 3194, implemented as a RAID Array in the presently preferred embodiment (but could be implemented in any storage medium of sufficient bandwidth and capacity), and streams that are not requested to be delayed are routed through Non-delayed Video path 3195. The output of both the Delay Buffer 3194 and Non-delayed Video 3195 is then routed by Internal Routing 3192 based on whether requested streams are to be scaled or not scaled. Scaled streams are

routed through Decompressors 3161-3164, Scalers 3171-3174 and Compressors 3181-3184 to Outbound Routing 3193. Non-scaled Video 3196 is also sent to Outbound Routing 3193, and then Outbound Routing 3193 then sends the video in unicast or multicast mode to App/Game Servers in the same manner as previously described in subsystem 3142 of Figure 31f.

Another embodiment of video delay buffer, scaling and decompression subsystem 3143 is shown in Figure 31h. In this embodiment, an individual Delay Buffer HQ 3121D-HQ 3131D is provided for each HQ stream. Given the rapidly declining cost of RAM and Flash ROM, which can be used to delay an individual compressed video stream, this may end up being less expensive and/or more flexible than having a shared Delay Buffer 3194. Or, in yet another embodiment, a single Delay Buffer 3197 (shown in dotted line) can provide delay for all of the HQ streams individually in a high-performance collective resource (e.g. very fast RAM, Flash or disk). In either scenario, each Delay Buffer HQ 3121D-3131D is able to variably delay a stream from the HQ video source, or pass the stream through without delay. In another embodiment, each delay buffer is able to provide multiple streams with different delay amounts. All delays or non-delays are requested by App/Game Services 3121-3125. In all of these cases Delayed and Non-Delayed Video streams 3198 are sent to Internal Routing 3192, and proceeds through the rest of the subsystem 3143 as previously described relative to Figures 31g.

In the preceding embodiments relative to the various Figures 31*n* note that the Live stream utilizes a two-way connection and is tailored for an particular user, with minimal latency. The HQ streams utilize one-way connections and are both unicast and multicast. Note that while the multicast function is illustrated in these Figures as a single unit, such as could be implemented in a Gigabit Ethernet switch, in a large scale system, the multicast function would likely be implemented

through a tree of multiple switches. Indeed, in the case of a video stream from a top-ranked video game player, it may well be the case that the player's HQ stream is watched by millions of users simultaneously. In such a case, there would likely be a large number of individual switches in successive stages broadcasting the multicasted HQ stream.

For both diagnostic purposes, and so as to provide feedback to the user (e.g. to let the user know how popular his gameplay performance is), in one embodiment, the hosting service 210 would keep track of how many simultaneous viewers there are of each App/Game Server 3121-3125's video stream. This can be accomplished by keeping a running count of the number of active requests by App/Game servers for a particular video stream. Thus, a gamer who has 100,000 simultaneous viewers will know that his or her gameplay is very popular, and it will create incentive for game players to do a better performance and attract viewers. When there is very large very viewership of video streams (e.g. of a championship video game match), it may be desirable for commentators to speak during the video game match such that some or all users watching the multicast can hear their commentary.

Applications and Games running on the App/Game servers will be provided with an Application Program Interface (API) in which the App and/or Game can submit requests for particular video streams with particular characteristics (e.g. resolution and amount of delay). Also, these APIs, submitted to an operating environment running on the App/Game Server, or to a Hosting Service Control System 401 of Figure 4a may reject such requests for a variety of reasons. For example, the video stream requested may have certain licensing rights restrictions (e.g. such that it can only be viewed by a single viewer, not broadcast to others), there may be subscription restrictions (e.g. the viewer may have to pay for the right

to view the stream), there may be age restrictions (e.g. the viewer may have to be 18 to view the stream), there may be privacy restrictions (e.g. the person using the App or playing the game may limit viewing to just a selected number or class of viewers (e.g. his or her "friends"), or may not allow viewing at all), and there may be restrictions requiring the material is delayed (e.g. if the user is playing a stealth game where his or her position might be revealed). There is any number of other restrictions that might limit viewing of the stream. In any of these cases, the request by the App/Game server would be rejected with a reason for the rejection, and in one embodiment, with alternatives by which the request would be accepted (e.g. stating what fee must be paid for a subscription).

HQ video streams that are stored in Delay Buffers in any of the preceding embodiments may be exported to other destinations outside of the Hosting Service 210. For example, a particularly interesting video stream can be requested by an App/Game server (typically by the request of a user), to be exported to YouTube. In such a case, the video stream would be transmitted through the Internet in a format agreed-upon with YouTube, together with appropriate descriptive information (e.g. the name of the user playing, the game, the time, the score, etc.). This could be implemented by multicasting in a separate stream the commentary audio to all of the Game/App Servers 3121-3125 requesting such commentary. The Game/App Servers would merge the audio of the commentary, using audio mixing techniques well-known by practitioners in the art, into the audio stream sent to the user premises 211. There could well be multiple commentators (e.g. with different viewpoints, or in different languages), and users could select among them.

In a similar manner, separate audio streams could be mixed in or serve as replacement for the audio track of particular video streams (or individual streams)

in the Hosting Service 210, either mixing or replacing audio from video streaming in real-time or from a Delay Buffer. Such audio could be commentary or narration, or it could provide voices for characters in the video stream. This would enable Machinima (user-generation animations from video game video streams) to be readily created by users.

The video streams described throughout this document are shown as captured from the video output of App/Game servers, and then being streamed and/or delay and being reused or distributed in a variety of ways. The same Delay Buffers can be used to hold video material that has come from non-App/Game server sources and provide the same degree of flexibility for playback and distribution, with appropriate restrictions. Such sources include live feeds from television stations (either over-the-air, or non-over-the-air, such as CNN, and either for-pay, such as HBO, or free). Such sources also include pre-recorded movies or television shows, home movies, advertisements and also live video teleconference feeds. Live feeds would be handled like the live output of a Game/App Server. Pre-recorded material would be handled like the output of a Delay Buffer.

VIDEO WINDOW ZOOMING AND TRANSLATION

Figures 16, 17 and 18 shows a user interface through the stages of zooming a thumbnail video window 1600 to medium size 1700, and finally to full size video window 1800. The zoomed video windows 1600, 1700 and 1800 are generated by app/game server 1521-1525 of Figure 15, as described previously. If the user is viewing video windows 1600, 1700 and 1800 on a typical desktop monitor or TV screen, once a window has been zoomed to full size, the image is often large enough for the user to see the content of the window in sufficient detail for the user's application. For example, the content shown in video window 1800 is a video game, and if the display device is a 32" TV set, or a 23" computer monitor, then if the user

is within typically viewing distance for the display device, the user will be able to make out sufficient details in the video so as to be able to play the game. As an example, the user will be able to see there is a car 1802 up ahead (and perhaps avoid it, run into it, follow it, etc. depending on the game) and the user will be able to discern details of map 1803 and speedometer 1804 with sufficient clarity so as to be able to play the game.

If the user's display device is very small, however, then it may be the case that the video window 1800 is so small that the user is unable to discern sufficient details in the content for the user's application (or due to limitations of vision). For example, if the display device is a small cellular phone, such as an Apple® iPhone, or a small media player, such as an Apple iPod Touch, the very small screen size and/or low screen resolution may make it difficult or impossible to make out car 1802, or read details of map 1803 or speedometer 1804. If so, it may be difficult or impossible to play the game.

In one embodiment, server 1521-1525 is responsive to user input to zoom video window 1800 to be larger than the full size of the screen and/or the server 1521-1525 is responsive to user input to translate horizontally and/or vertically and/or diagonally the zoomed-up screen. This allows the user to zoom in on particular areas of the screen so as to view them in more detail. For example, if the user zooms in to the center of the screen, car 1802 can become big enough to be discernable. If the user zooms into the screen and translates the image so that the lower left is visible, then map 1803 can become discernable. If the user zooms into the lower right of the screen, then speedometer 1804 can become discernable. Depending on the nature of the game or application, the user may want to pause the video either before, during or after zooming/translating by performing a user interface action that indicates to server 1521-1525 that the user desires to pause.

For example, in a driving game, the user may not pause while zoomed into the center of the screen and focused on the road, but may pause before viewing the map 1803 or speedometer 1804, so as not to crash the car while the road ahead is no longer visible on the screen. In the case of a productivity application or a web browser application, it may not be necessary to pause the video for zooming or translation.

There are various prior art user interface techniques that are used for scaling and/or translating prior art windows, such a “pinching”, “spreading” and/or “swiping” a touch screen or track pad with the fingers, e.g. on an Apple iPhone, iPod Touch or Macintosh. As another example, the scroll wheel on a mouse can be used to specify how much to zoom, while clicking the mouse and moving it can be used to specify translation. These techniques and others can be applied to embodiments of video window 1800 of the present invention so that the user can specify zoom and/or translation. The specified scaling and/or translation user input is sent as Control Signals 406 in Figure 4a to server 1521-1525, and the requested scaling and/or translation action is performed. In one embodiment, to the extent the scaled and/or translated video window would extend beyond the edge of the display device, it is cropped by server 1521-1525 to the edges of the display device (or the edges of a window containing the zoomed portion of the image), so that only the visible pixels are compressed by Shared Hardware Compression 1530 and sent to Home or Office client device 415 in Figure 4a, so as to reduce transmission bandwidth. In another embodiment, some or all of the zoomed and/or translated video window that would extend beyond the edges of the display device (or the edges of a window containing the zoomed portion of the image) is generated by server 1521-1525 and is compressed by Shared Hardware Compression 1530 and sent to Home or Office client device 415, and then is cropped to the edges of the display device or the

edges of a window containing the zoomed portion of the image). Although the embodiment of the preceding sentence results in higher transmission bandwidth than minimally necessary, the scaling and/or translating operation is carried out locally by Home or Office client device 415, which may reduce the latency of the response time of the zooming and/or translation operation, particularly if the connection between Home or Office client device 415 and Hosting Service 210 is incurring noticeable latency.

In one embodiment, the scaled and/or translated video window is a Web browser window. Prior art devices with small screens, for example, cellular phones or media players, such as the Apple iPhone or iPod Touch, have Web browsers which have the facility to zoom and/or translate the window of the displayed web site (e.g. by “pinching”, “spreading” and/or “swiping”) so as view the content more closely to discern more detail or view it with low detail in its entirety.

In a practical context, there are significant disadvantages using such prior art Web browsers, often resulting in slow Web page loading or inefficient usage of network resources. If the user is zoomed into a Web page, typically the browser still needs to download most, if not all, of the Web page content because it is typically necessary for the Web browser to process and display all or most of the elements of the Web page before it can determine what the resulting image will be in the zoomed-in portion of the Web page. As one simple example, if the Web browser is zoomed into a portion of a large jpeg image, the jpeg compression algorithm typically will require the entire image to be decompressed before a portion of the image can be displayed. As another example, HTML pages often implicitly or explicitly position certain elements of a Web page relative to the position of other elements, requiring elements that may be off the edge of a zoomed-in page to be

downloaded and parsed before the position of the elements of the zoomed-in portion is known.

Further, as described above and illustrated in Figure 24, the load time of websites asymptotically approaches the sum of the load time incurred in loading each file that makes up the website to the extent the loading of the files cannot be overlapped. And, also, many elements, such as jpeg images for rollovers, have to be loaded, but may never be displayed.

Figure 24 illustrates the load time at various connection speeds of a 54-file Web site where each file has a latency overhead of 100ms, which is a common latency for HTTP files using a wireline Internet connection. If a cellular network is used, for example, the AT&T 3G cellular network, the HTTP file latency overhead is typically much higher – perhaps as high as 400ms or more. So, while the example website of Figure 24 asymptotically approaches 5.4 seconds of load time, with 400ms HTTP file latency, the same website would asymptotically approach 21.6 seconds of load time, and given that 3G cellular networks may well operate at lower connection speeds than wireline connections, the total load time may be 30 seconds or longer. And, only then will the Web browser be able to display a Web page, even if only a zoomed-in portion is being viewed.

This is not only an inconvenience to the user, but it is a very inefficient use of wireless data throughput, which often is shared among many users, and often less (frequently far less) wireless data throughput is available than there is demand for.

As previously described, in one embodiment a Web browser is hosted in server 1521-1525, sourcing Web page files stored locally in Hosting Service 210, or sourcing files by utilizing a connection to the Internet available in Hosting Service 210. The Web browser window is then transmitted to Home or Office Client device

415 for display. The Web browser is controlled by the user's actions via Control Signals 406. If the Home or Office Client device 415 is a cellular phone operating through a cellular network, the latency will typically be much higher between the cell phone and the Internet than between the Hosting Service 210 and the Internet. So, whether the Web page files are stored locally in the Hosting Service 210 or located on remote Web servers, it is highly likely that the user will experience much lower delay in viewing a Web page generated by a Web Browser hosted in server 1521-1525 and transmitted to the cellular phone as video, than the delay the user would experience if the same Web page were generated by a Web Browser running locally on the cellular phone, with Web page files transmitted conventionally through the cellular network. Further, as previously described, hosting the Web Browser in server 1521-1525 would likely result in much less data being transmitted for a given Web page to the Home or Office Client device 415. Also, if the cellular network connection speed degrades, various techniques previously described can be used to dynamically reduce the transmitted data throughput so as to not exceed the capacity of the cellular channel. Additionally, this approach eliminates the need for keeping a local Web browser up-to-date (e.g. with the latest version of Adobe Flash), since the Web browser running in server 1521-1525 can be kept up-to-date. Indeed, the Web browser running in server 1521-1525 may be able to perform operations that are beyond the computational capabilities of Home or Office Client device 415.

In one embodiment, the video window generated on the Home or Office or Client device 415 can be zoomed and/or translated as previously described. In this way, the user can either zoom and/or translate to view detail in the Web page, or if zooming out, get an overall view of the Web page. Indeed, it may well be the case that the Web Page image is zoomed down in size to only take up a portion of the screen, such as the windows 1600 and 1700, and indeed, may be viewed

simultaneously with other windows, whether they are other Web pages, or whether they are video windows of video games, videos, applications, etc.

Some prior art Web browsers, such as Apple Safari, can display windows of multiple reduced-sized websites at once, for example, to show which websites the user has visited most frequently. Because the multiple websites are displayed simultaneously, it is typically impractical to display all of them dynamically (e.g. showing what is presently displayed live on the websites) because, among other issues, the sum of the bandwidth demands from all of the websites may be excessive. For example, if 8 websites are showing video, the full video bandwidth from all websites must be received, decompressed and then scaled down to the size of reduced-sized websites. This not only results in a non-real-time experience for the user, but it is wasteful of bandwidth since the entire websites need to be downloaded, and then they are scaled down, with much of the detail lost.

In one embodiment, one or more Web browsers are hosted on server 1521-1525 and the video is scaled before being sent to Office or Client device 415, so as to appear as multiple scaled-down video windows, such as those shown in Figure 16. In contrast to prior art Web browsers that can display multiple Web sites at once as reduced-size windows, in this embodiment, the Web sites can all be displayed live at once, even if some or all of them incorporated high bandwidth elements, like video. And, these multiple live website windows can be simultaneously displayed with other live video windows, such as video games, videos, applications, etc.

In one embodiment, the number of windows available for viewing by the user is larger than the number of video windows displayed within the bounds of the display device. For example, in Figure 16, a 6x3 array of 18 video windows is visible, but there may well be far more video windows available for viewing by the

user (e.g. a 20x20 array), of which only the 6x3 subset is visible within the bounds of the display device at a given time. In one embodiment, the user is able to effectuate a translation (e.g. horizontally, vertically and/or diagonally) of the array of video windows by "swiping" a finger across a touch screen or a track pad, thus creating the illusion that the finger touch is causing the array of video windows to translate its position and reveal other video windows. This can be implemented by the finger swipe control information (e.g. position of the swipe, velocity of the swipe, etc.) being sent as Control Signals 406, and the app/game servers 1521-1525 implementing an animation effect showing the motion of the array of video windows. In one embodiment, the animation effect is a translation showing the video wall moving horizontally, vertically or at an angle in response to the finger swipe (or other user input, such as controller action or keyboard/mouse presses, body motion, etc.), but with the video windows all remaining the same size as they move. In another embodiment, the animation effect is a non-rectilinear motion, where the video wall moves with a complex motion in response to a finger swipe (or other user input actions), and some or all of the video windows change size during the animation effect. One such complex motion is a perspective 3D motion, creating the illusion that the video wall is located in 3D space.

In one embodiment, the various functional modules illustrated herein and the associated steps may be performed by specific hardware components that contain hardwired logic for performing the steps, such as an application-specific integrated circuit ("ASIC") or by any combination of programmed computer components and custom hardware components.

In one embodiment, the modules may be implemented on a programmable digital signal processor ("DSP") such as a Texas Instruments'

TMS320x architecture (e.g., a TMS320C6000, TMS320C5000,..., etc.). Various different DSPs may be used while still complying with these underlying principles.

Embodiments may include various steps as set forth above. The steps may be embodied in machine-executable instructions which cause a general-purpose or special-purpose processor to perform certain steps. Various elements, which are not relevant to these underlying principles such as computer memory, hard drive, input devices, etc., have been left out of some or all of the figures to avoid obscuring the pertinent aspects.

Elements of the disclosed subject matter may also be provided as a machine-readable medium for storing the machine-executable instructions. The machine-readable medium may include, but is not limited to, flash memory, optical disks, CD-ROMs, DVD ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, or other type of machine-readable media suitable for storing electronic instructions.

It should also be understood that elements of the disclosed subject matter may also be provided as a computer program product which may include a machine-readable medium having stored thereon instructions which may be used to program a computer (e.g., a processor or other electronic device) to perform a sequence of operations. Alternatively, the operations may be performed by a combination of hardware and software. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnet or optical cards, or other type of machine-readable medium suitable for storing electronic instructions.

Additionally, although the disclosed subject matter has been described in conjunction with specific embodiments, numerous modifications and alterations are well within the scope of the present disclosure. Accordingly, the

specification and drawings are to be regarded in an illustrative rather than restrictive sense.

CLAIMS

What is claimed is:

1. A method comprising:
 rendering an application frame buffer in a hosting service that is streaming interactive video; and
 displaying only a portion of the application frame buffer on a local device.
2. The method of claim 1 wherein the hosting service only streams the portion displayed on the local device.
3. The method of claim 1 wherein the hosting service streams the application frame buffer in its entirety, the local device cropping the application frame buffer, with the portion being displayed in a video window on the local device.
4. The method of claim 3 wherein the local device sends a first control input to the hosting service regarding the cropping of the video window.
5. The method of claim 4 wherein the local device sends a second control input to the hosting service regarding positioning of the video window.
6. The method of claim 5 wherein the local device sends a third control input to the hosting service regarding scaling of the video window.

7. The method of claim 6 wherein the first, second, and third control inputs are generated responsive commands of a touch screen user interface.
8. The method of claim 7 wherein the commands include a dragging command.
9. The method of claim 7 wherein the commands include a swiping command.
10. The method of claim 7 wherein the commands include a pinching command.
11. The method of claim 7 wherein the commands include a widening command.
12. The method of claim 1 wherein the application comprises a web browser.
13. The method of claim 1 wherein the hosting service and the local device are connected to the Internet, the hosting service having a higher bandwidth connection to the Internet as compared to the local device.
14. The method of claim 1 wherein the hosting service and the local device are connected to the Internet, the hosting service having a lower latency connection to the Internet as compared to the local device.

ABSTRACT OF THE DISCLOSURE

A method for remote-hosted video effects includes rendering an application frame buffer in a hosting service that is streaming interactive video. Only a portion of the application frame buffer is subsequently displayed on a local device.

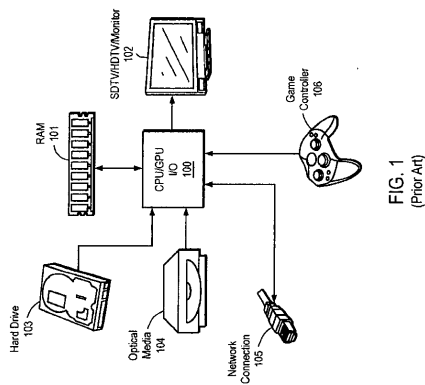


FIG. 1
(Prior Art)

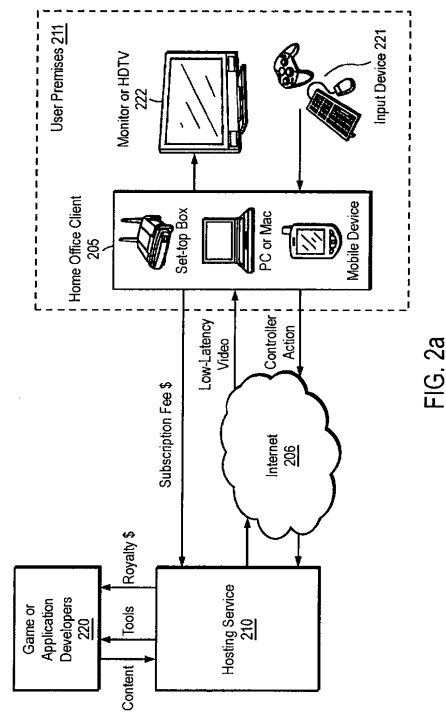


FIG. 2a

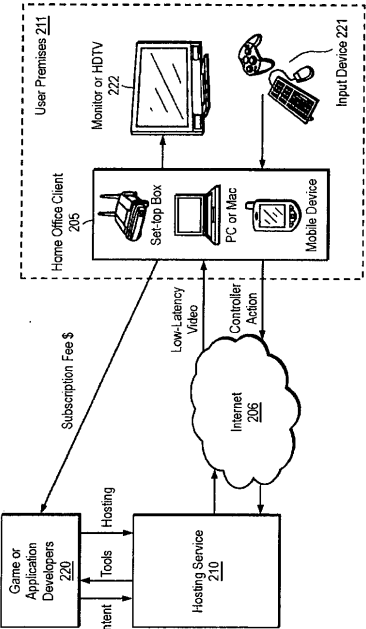


FIG. 2b

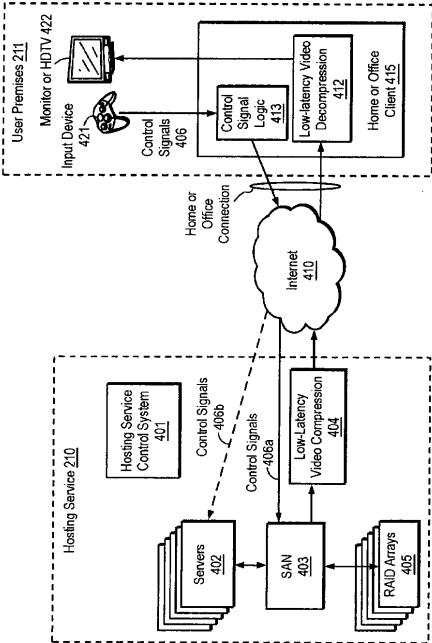


FIG. 4a

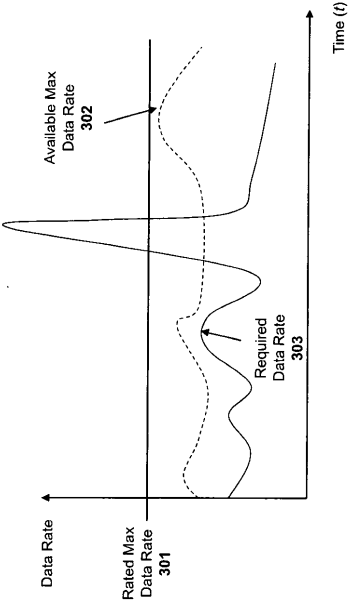


Fig. 3

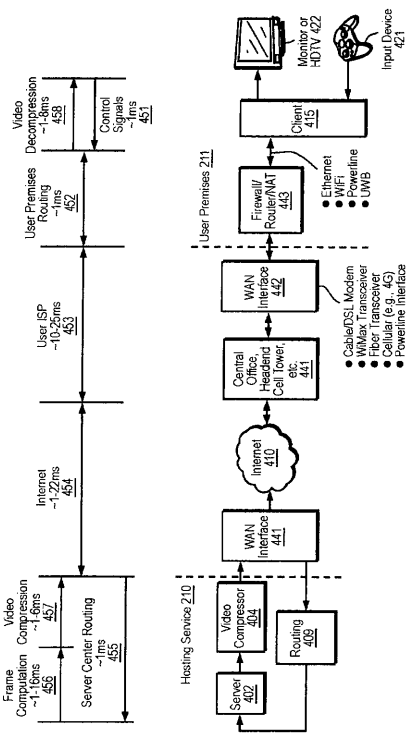
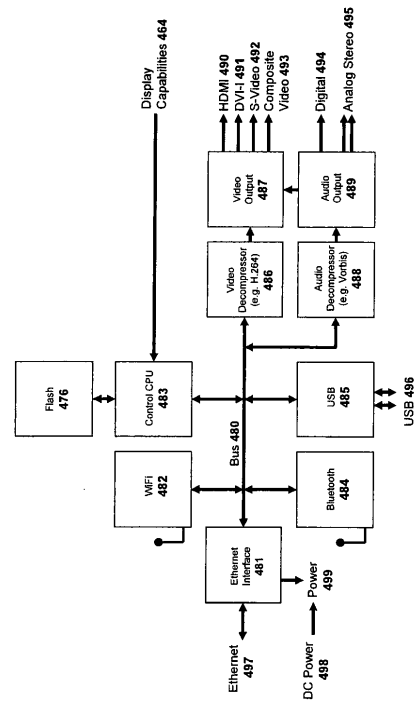
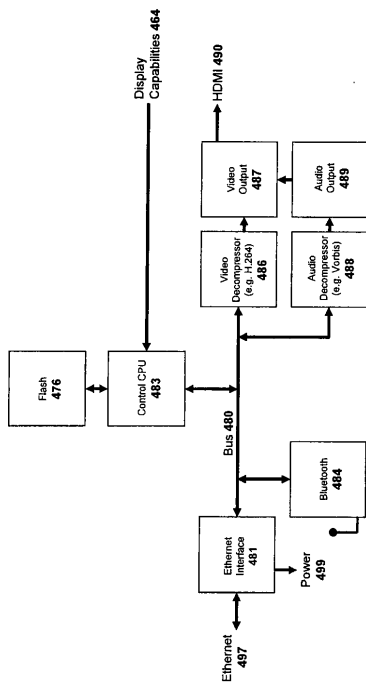
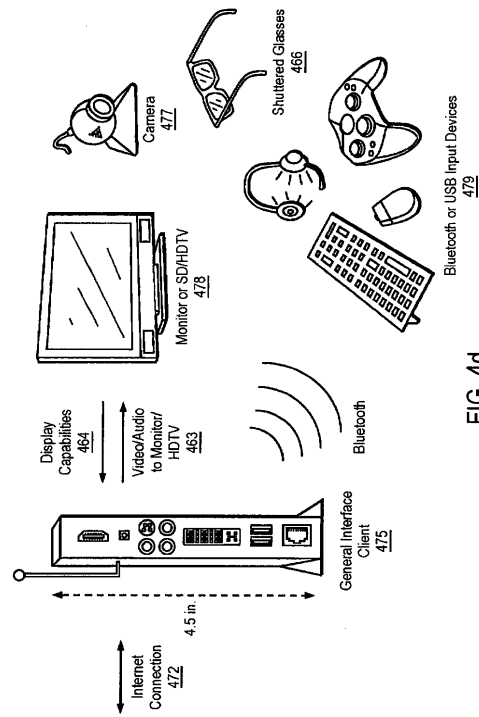
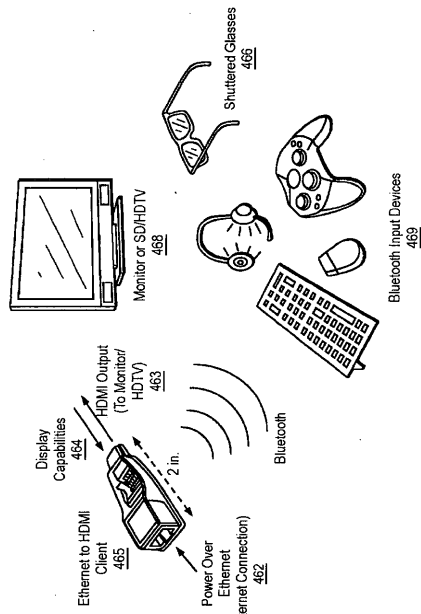


FIG. 4b



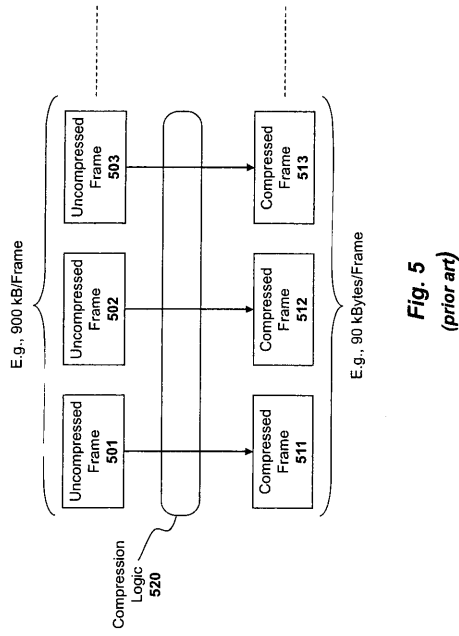
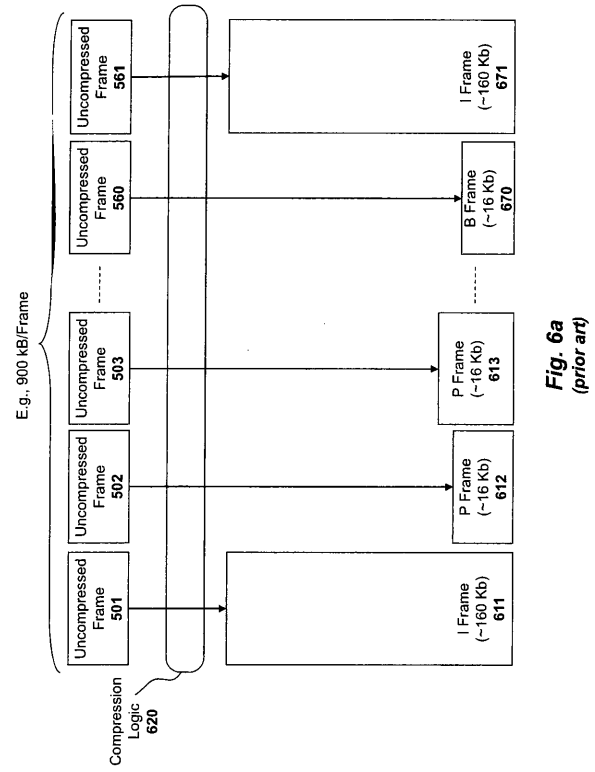
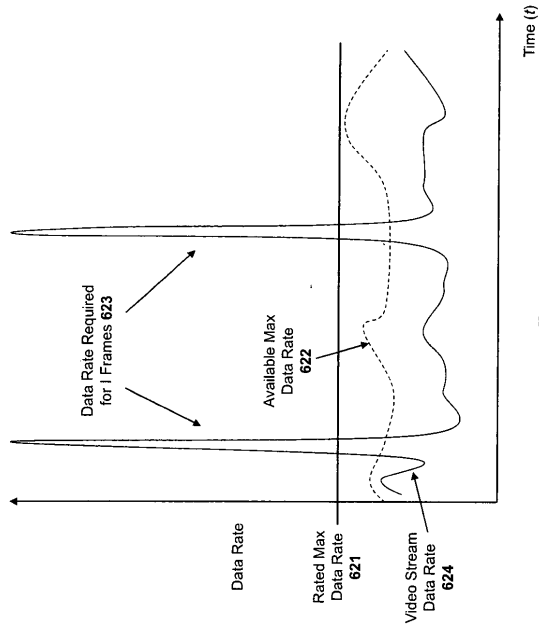
Fig. 5
(prior art)Fig. 6a
(prior art)

Fig. 6b

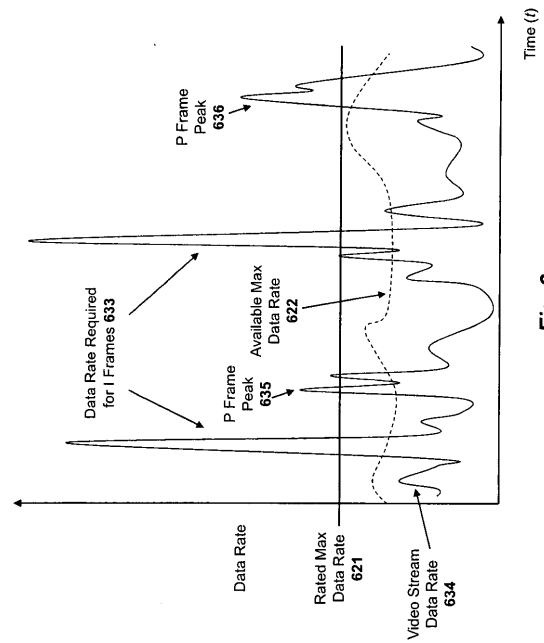


Fig. 6c

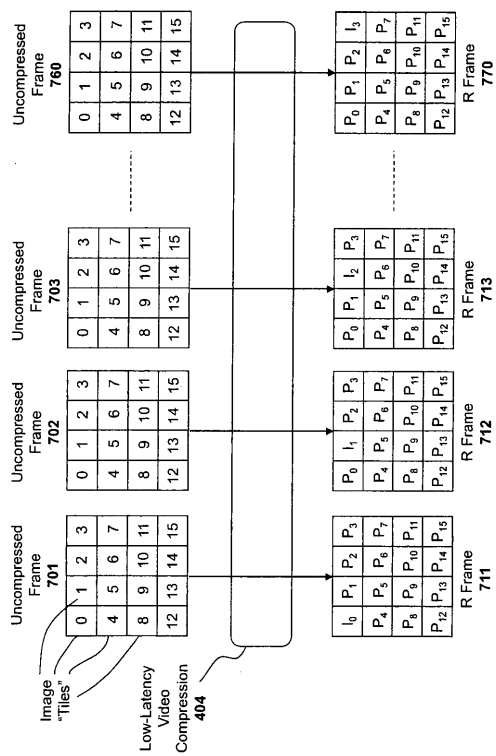


Fig. 7a

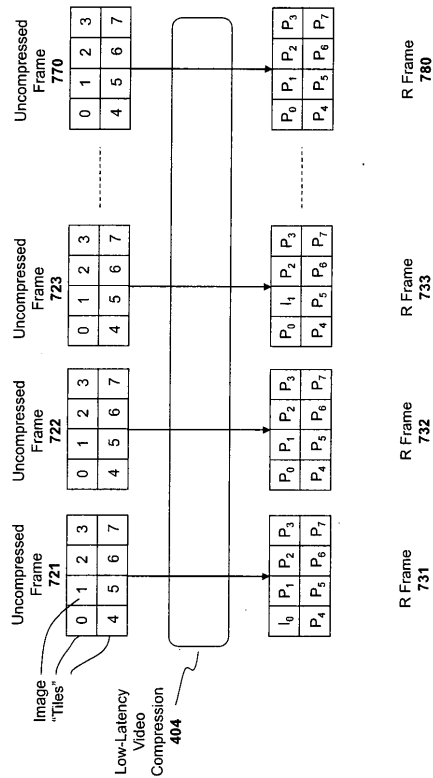


Fig. 7b

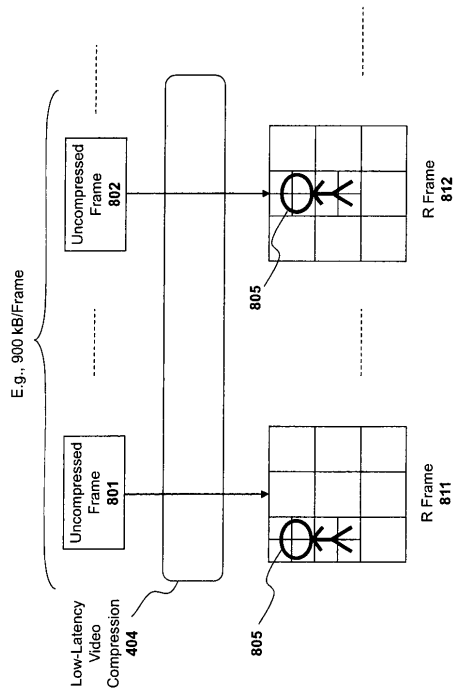


Fig. 8

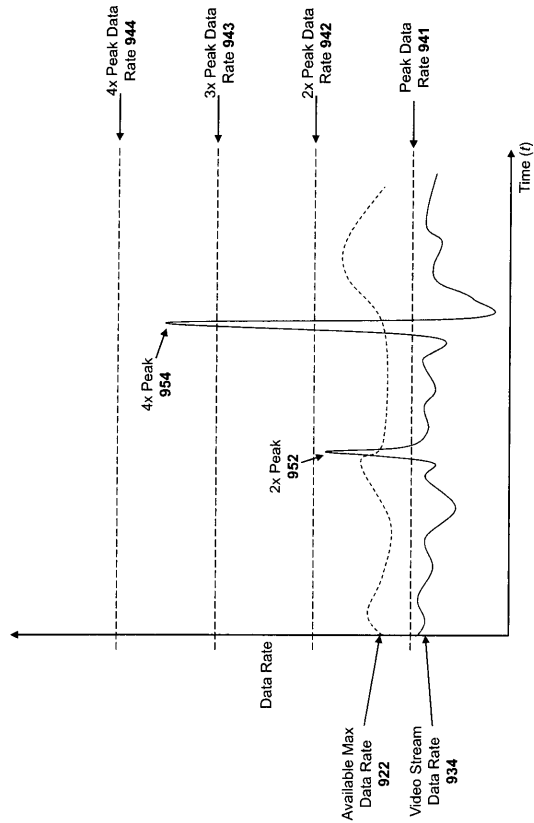


Fig. 9a

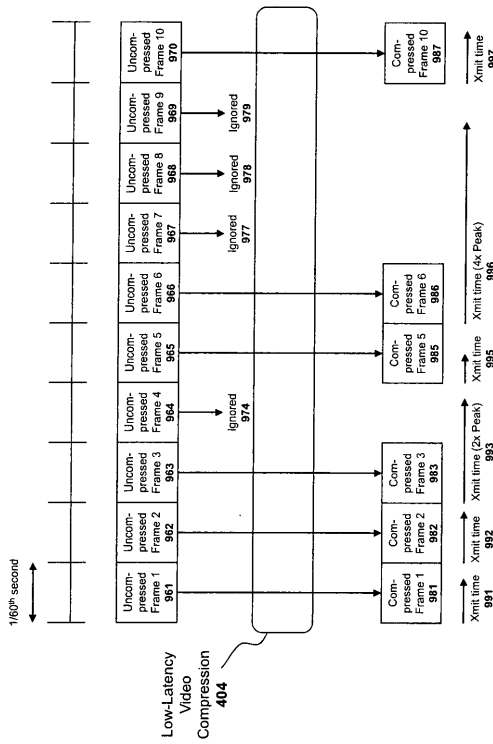
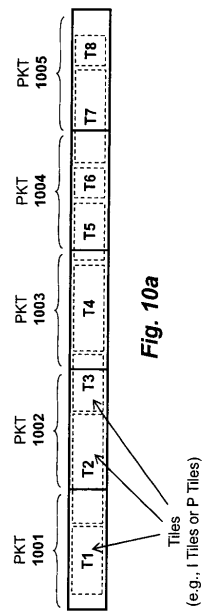
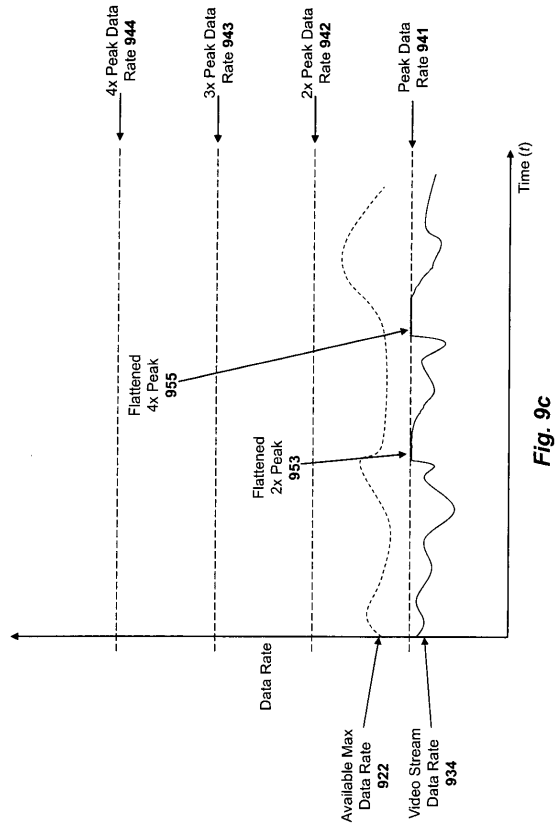
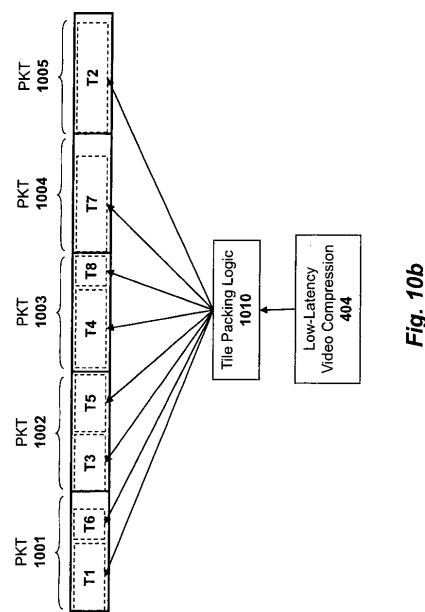
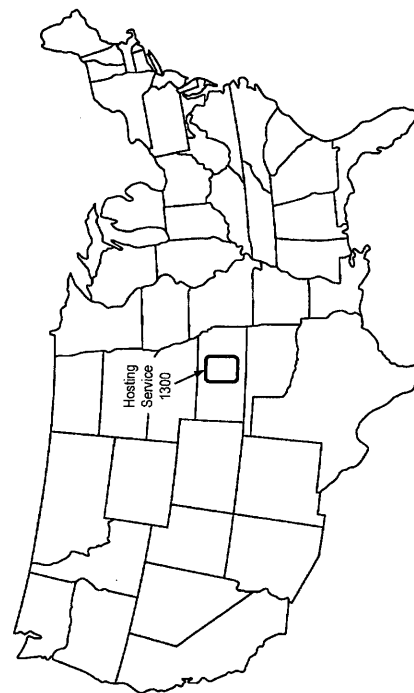
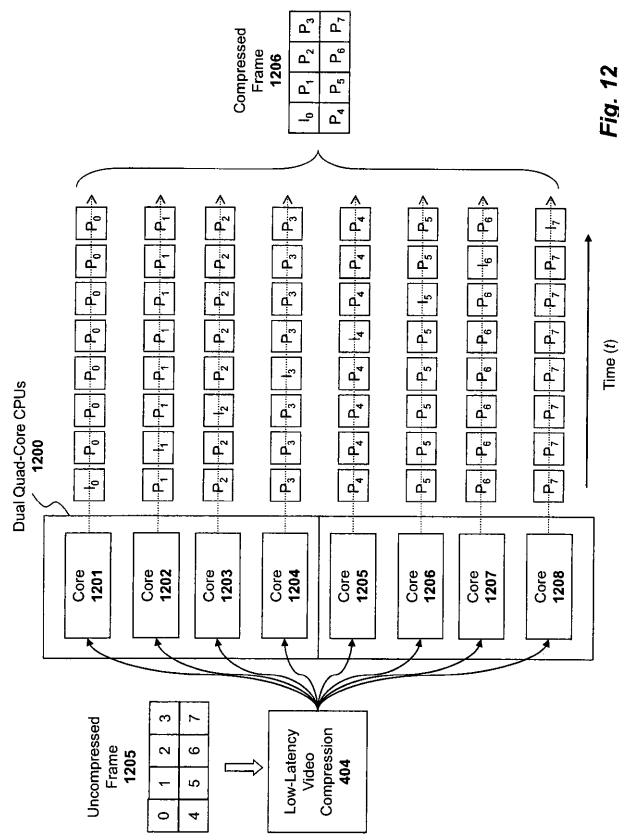
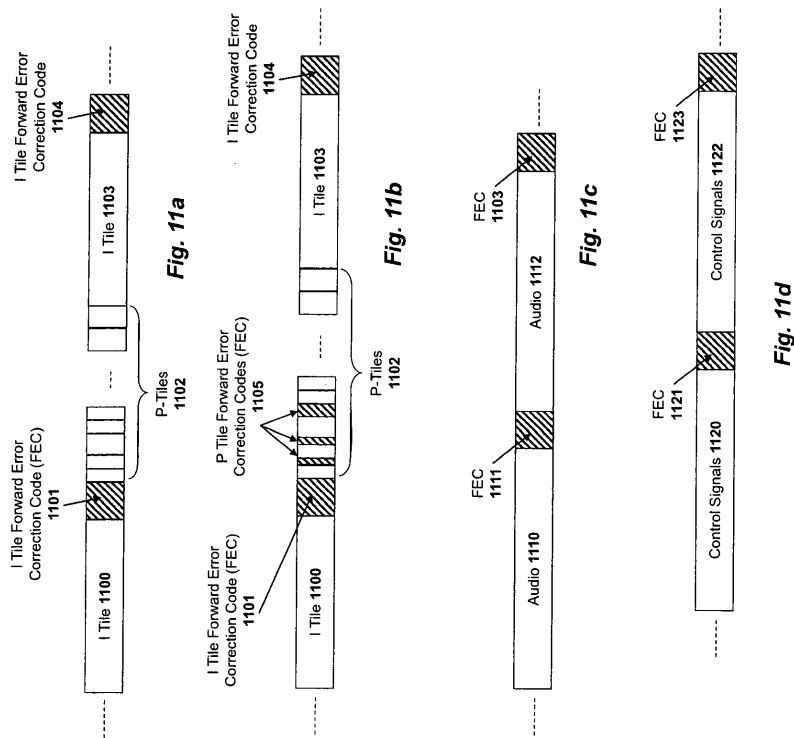


Fig. 9b





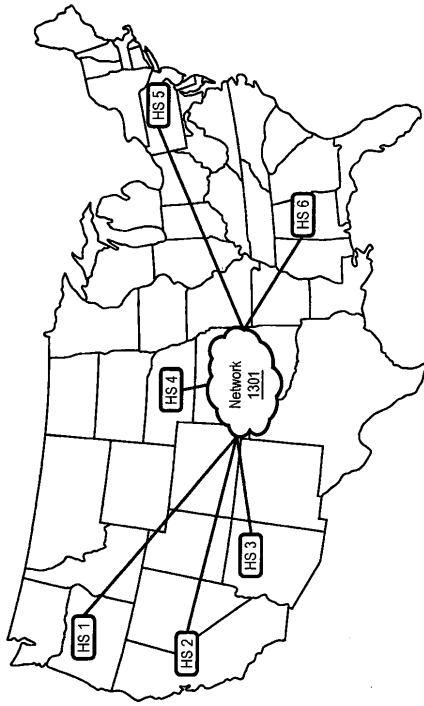


FIG. 13b

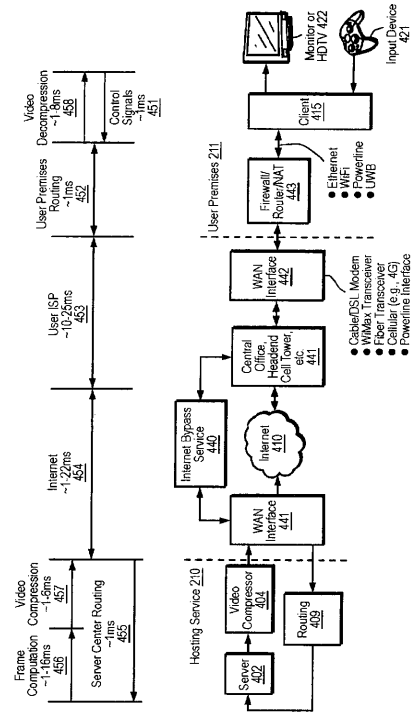


FIG. 14

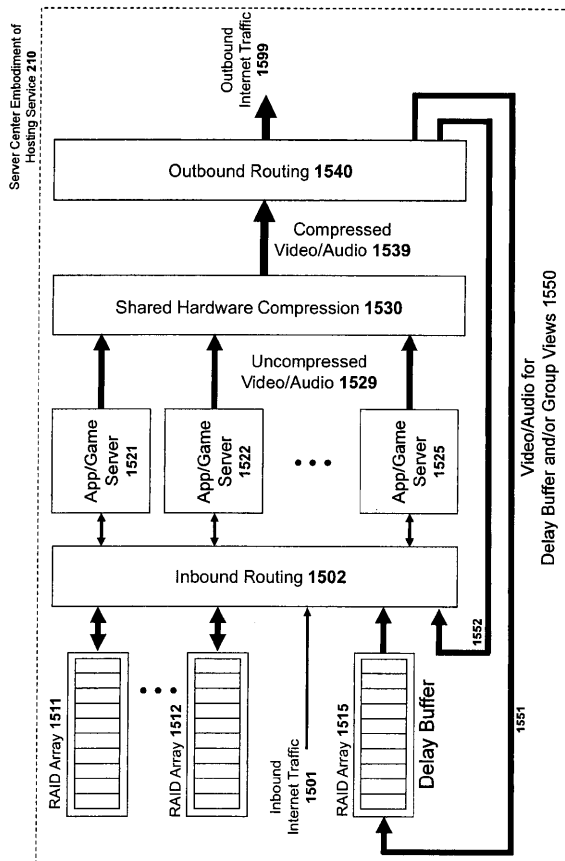


Fig. 15

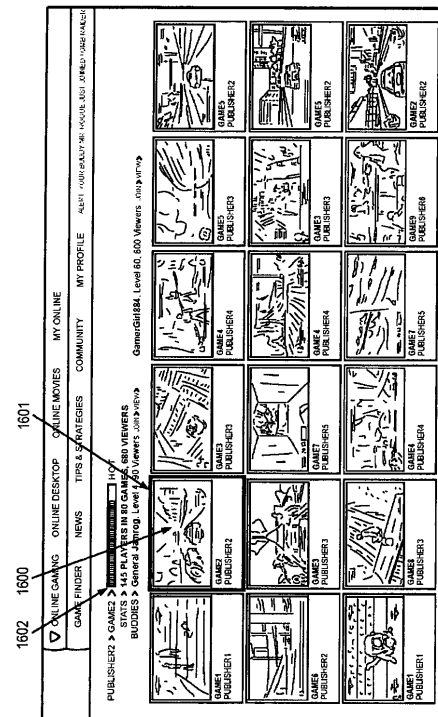


FIG. 16

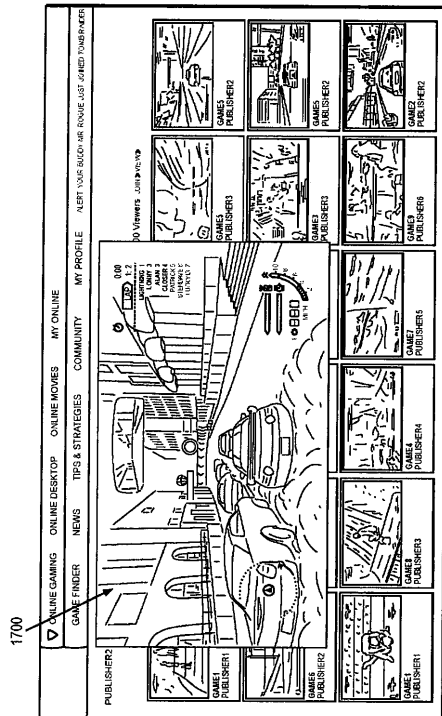


FIG. 17

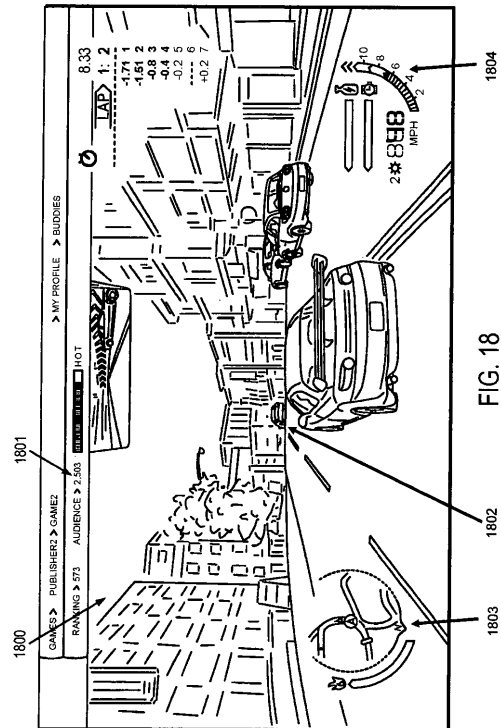


FIG. 18

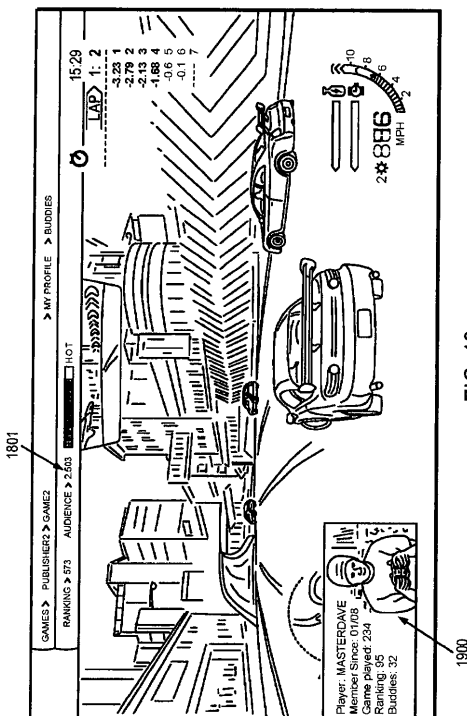


FIG. 19

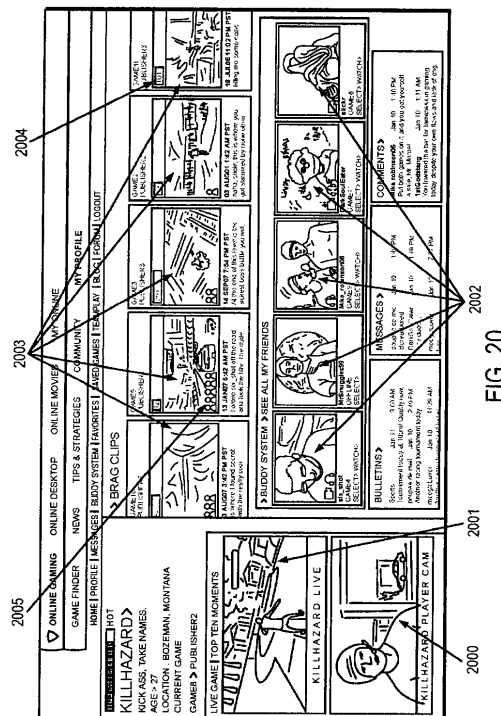
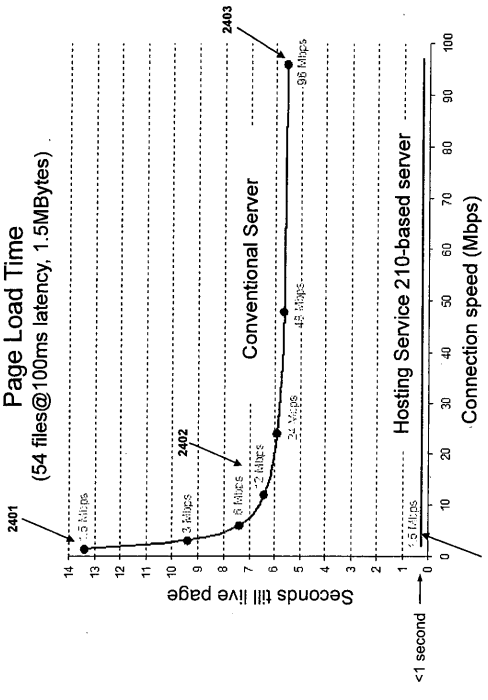
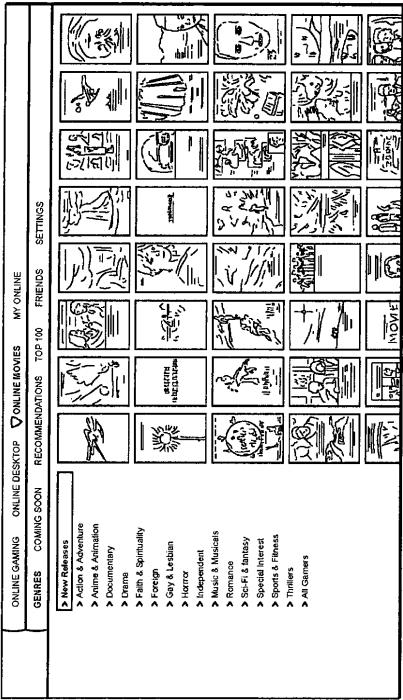
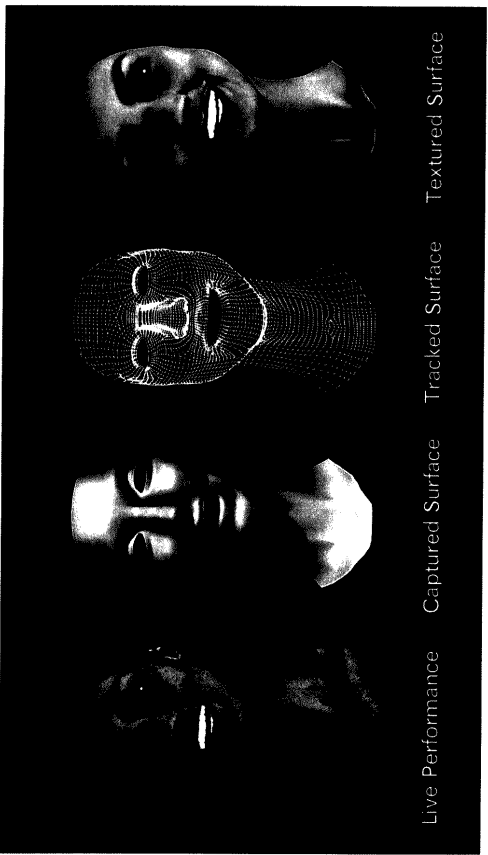
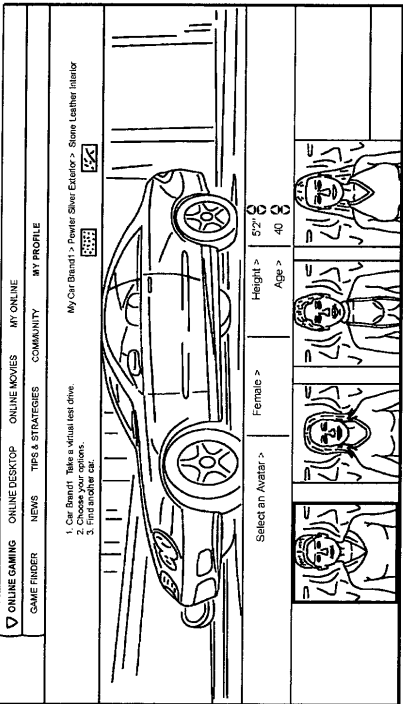
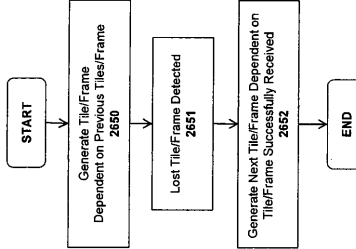
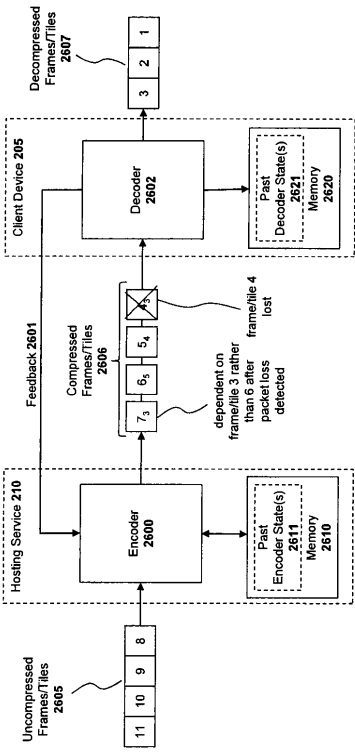
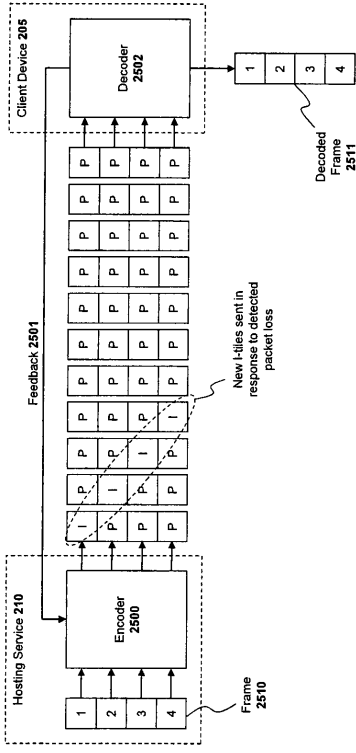
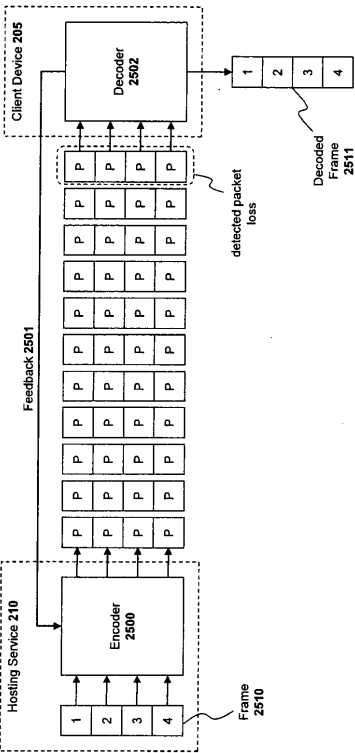


FIG. 20





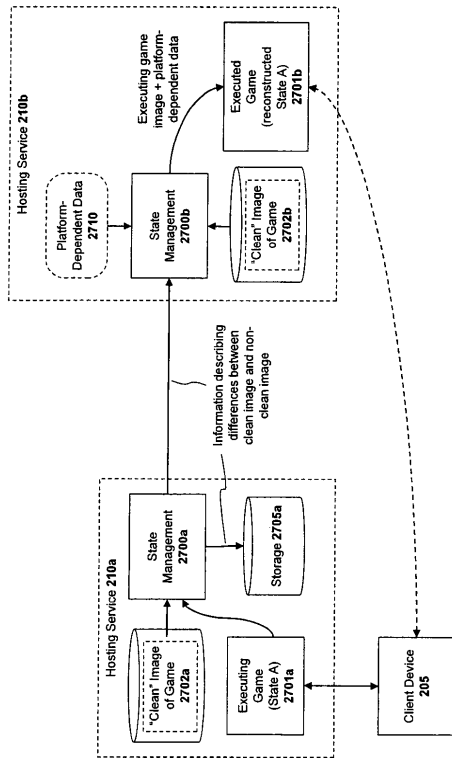


Fig. 27a

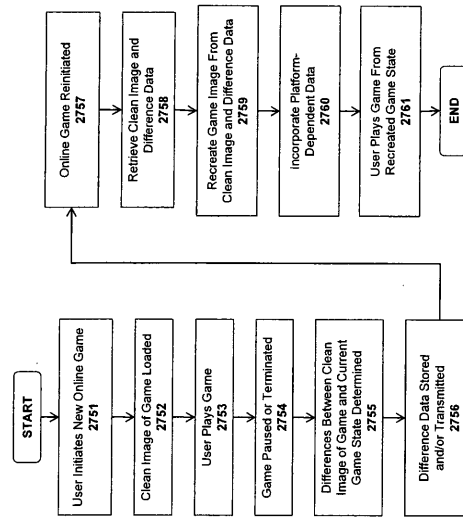


Fig. 27b

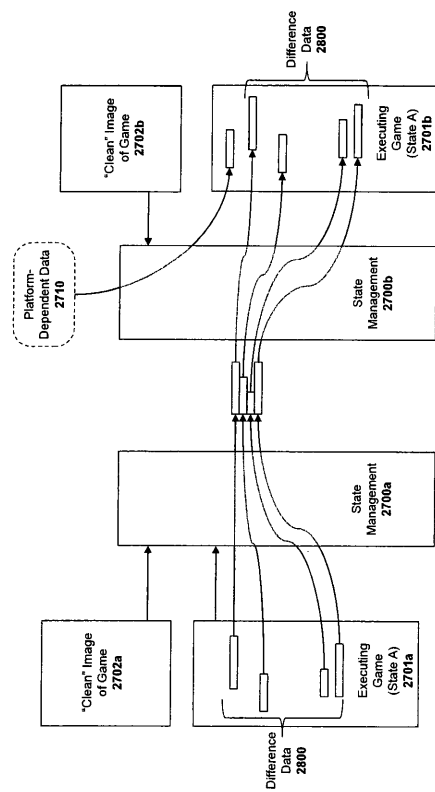


Fig. 28

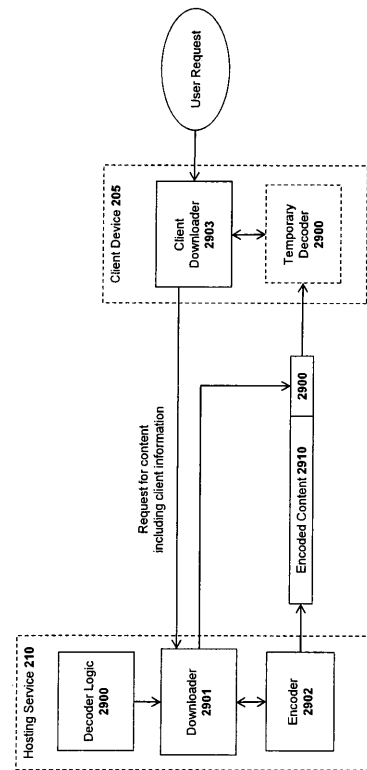


Fig. 29

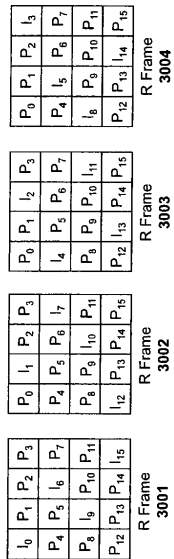


Fig. 30

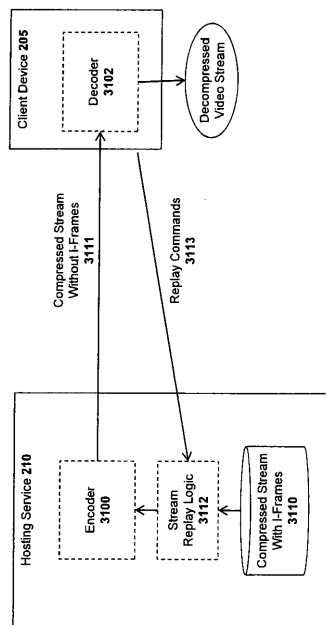


Fig. 31b

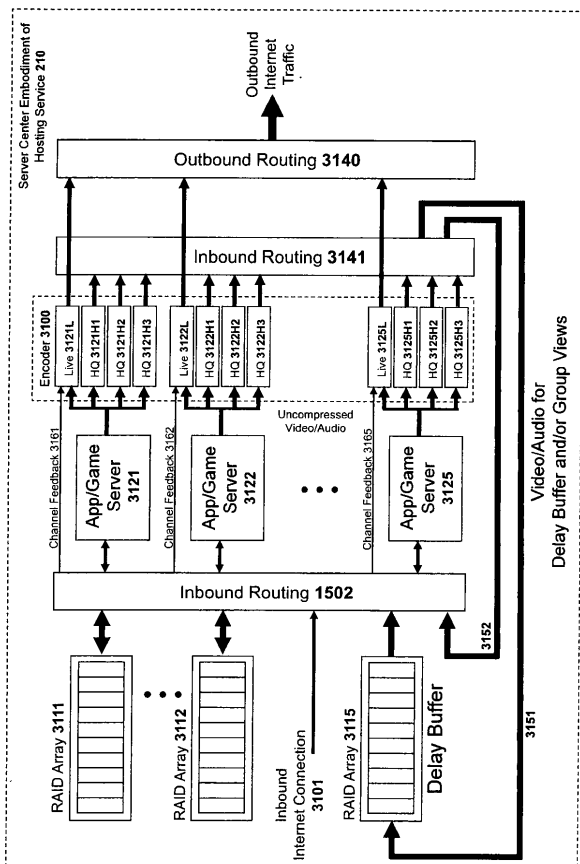


Fig. 31c

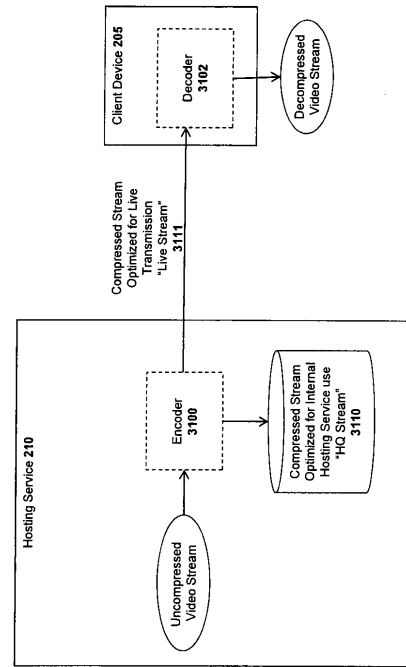


Fig. 31a

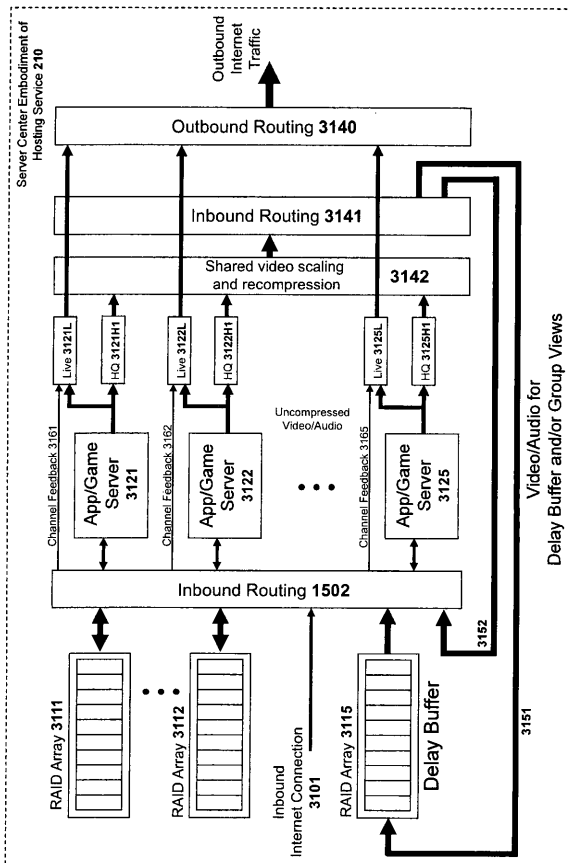


Fig. 31d

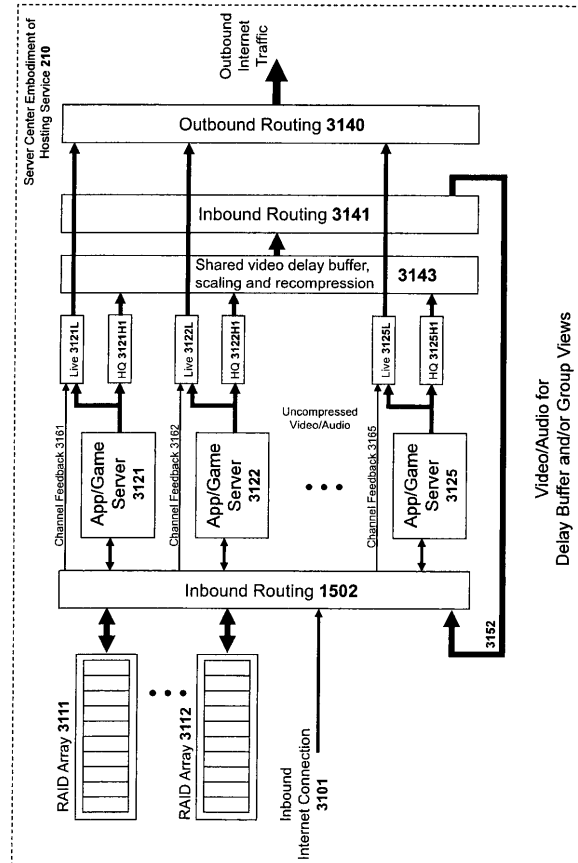


Fig. 31e

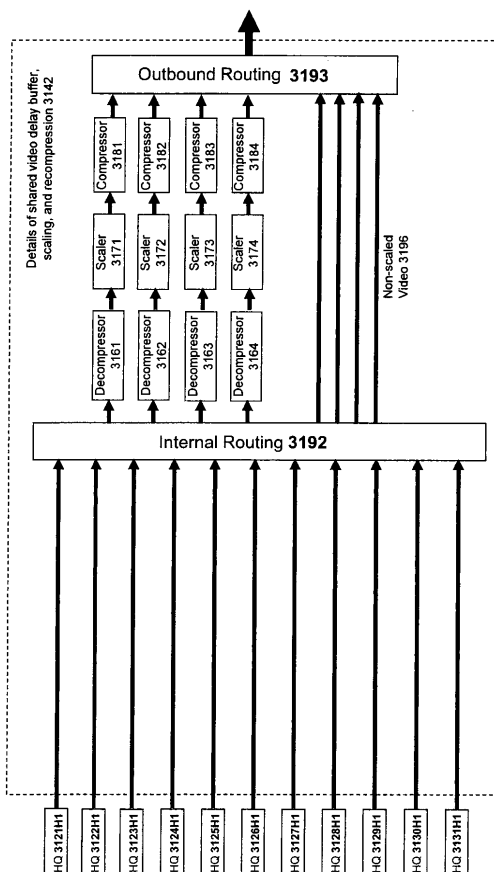


Fig. 31f

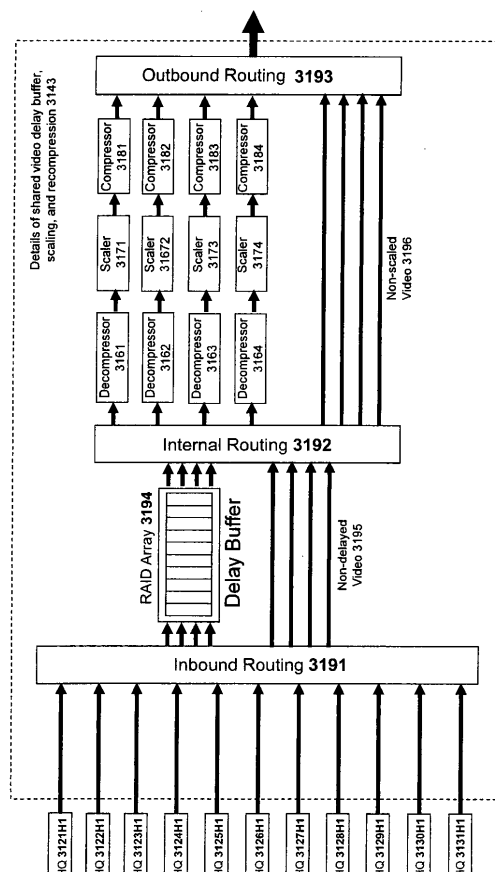


Fig. 31g

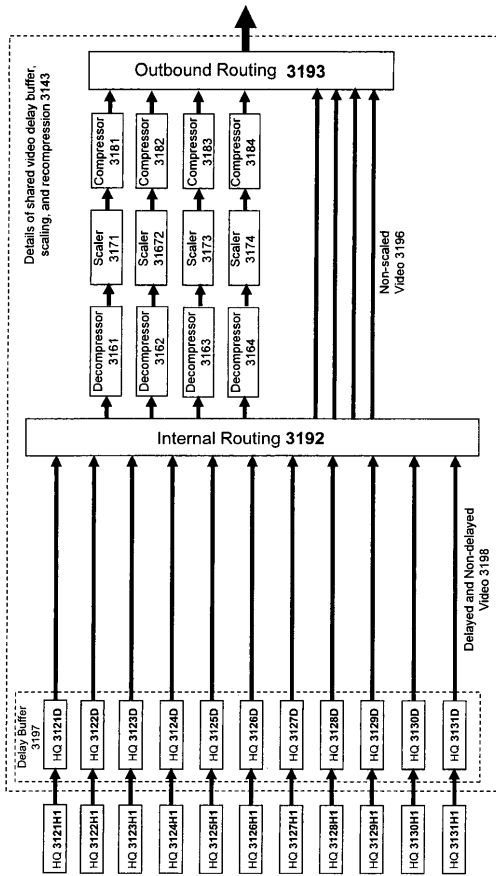


Fig. 31h