(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 September 2010 (30.09.2010)

PCT

(10) International Publication Number
WO 2010/111095 A1

(51) International Patent Classification:
H04N 7/12 (2006.01)

(21) International Application Number:
PCT/US2010/027712

(22) International Filing Date:
17 March 2010 (17.03.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/210,888 23 March 2009 (23.03.2009) US
12/538,096 7 August 2009 (07.08.2009) US

(71) Applicant *(for all designated States except US)*: **ONLIVE, INC.** [US/US]; 181 Lytton Avenue, Palo Alto, CA 94301 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **PERLMAN, Stephen, G.** [US/US]; 181 Lytton Avenue, Palo Alto, CA 94301 (US). **VAN DER LAAN, Roger** [NL/US]; 1310 University Drive, Menlo Park, CA 94025 (US). **COTTER, Timothy** [US/US]; 181 Matadero Drive, Sunnyvale, CA 94086 (US). **FURMAN, Scott** [US/US]; 123 Hedge Road, Menlo Park, CA 94025 (US). **MCCOOL, Robert** [US/US]; 360 Sherwood Way, Menlo Park, CA 94025 (US). **BUCKLEY, Ian** [US/US]; 73 Highland Avenue, San Carlos, CA 94070 (US).

(74) Agents: **VINCENT, Lester, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 1279 Oakmead Parkway, Sunnyvale, CA 94085-4040 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) **Title**: SYSTEM AND METHOD FOR VIDEO COMPRESSION USING FEEDBACK INCLUDING DATA RELATED TO THE SUCCESSFUL RECEIPT OF VIDEO CONTENT

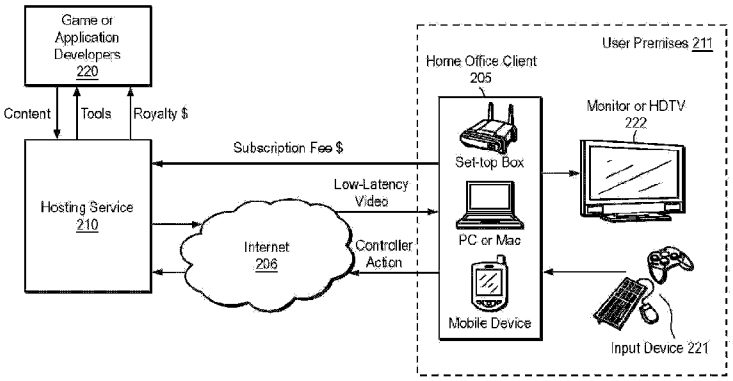


FIG. 2a

(57) **Abstract**: A computer-implemented system and method for performing video compression is described. A method according to one embodiment includes encoding a first plurality of video frames or portions thereof, wherein each encoded video frame or portion is dependent on a previously-encoded video frame or portion thereof. The method further includes transmitting the first plurality of encoded video frames or portions to a client device and receiving feedback information from the client device, the feedback information usable to determine whether data contained in the video frames or portions has not been successfully received and/or decoded; in response to detecting that a video frame has not been successfully received and/or decoded, encoding a current video frame or portion thereof to be dependent on a previously-encoded video frame or portion thereof known to have been successfully received and/or decoded; and transmitting the current video frame or portion thereof to the client device.

WO 2010/111095 A1

System and Method for Video Compression Using Feedback Including Data Related to the Successful Receipt of Video Content

RELATED APPLICATION

[01] This application claims priority to U.S. Provisional Application Serial No. 61/210,888, filed, March 23, 2009, entitled, "System And Method For Compressing Video Using Feedback", which is a continuation-in-part of co-pending U.S. Application Serial No. 12/359,150, filed January 23, 2009, entitled, "System And Method for Protecting Certain Types of Multimedia Data Transmitted Over A Communication Channel", and is a continuation of co-pending U.S. Application Serial No. 11/999,475, filed December 5, 2007, entitled, "Hosting And Broadcasting Virtual Events Using Streaming Interactive Video" which is a continuation-in-part (CIP) application of Serial No. 10/315,460 filed December 10, 2002 entitled, "APPARATUS AND METHOD FOR WIRELESS VIDEO GAMING", which is assigned to the assignee of the present CIP application.

TECHNICAL FIELD

[02] 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

[03] 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.

[04] 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.

[05] 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.

[06] 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.

[07] 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 us 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).

[08] 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.

[09] 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.

[0010] 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).

[0011] The issue of media device obsolescence and complexity is a serious problem when it comes to sophisticated interactive media, especially video games.

[0012] 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.

[0013] 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.

[0014] 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.

[0015] 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.

[0016] 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.

[0017] 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.

[0018] 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.

[0019] 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.

[0020] 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.

[0021] 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.

[0022] 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 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.

[0023] 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.

[0024] 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.

[0025] 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.

[0026] 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.

[0027] 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 they 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.

[0028] 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.

[0029] 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 usably 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.

[0030] 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.

[0031] 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.

[0032] 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.

[0033] 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.

[0034] 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.

[0035] 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.

[0036] 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.

[0037] 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.

[0038] 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.

[0039] 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).

[0040] 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.

[0041] 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.

[0042] 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.

[0043] 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.

[0044] 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.

[0045] 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).

[0046] 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.

[0047] 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.

[0048] 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).

[0049] 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. On 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.

[0050] Thus, the prior art game architecture shown in Figure 1 subjects the user to significant delays between major scene transitions of complex games.

[0051] 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.

[0052] 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).

[0053] 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.

[0054] 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.

[0055] 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.

[0056] 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.

[0057] 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.

[0058] 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.

[0059] 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.

[0060] 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.

[0061] 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 results in what appears to the viewer as a moving visual artifacts when the video is decompressed.

[0062] 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.

[0063] 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 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 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 303,935/2Mb = 145ms to transmit an I frame.

[0064] 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 = 18.2Mbps) 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.

[0065] 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 = 100ms 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.

[0066] 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.

[0067] 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).

[0068] 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 provisioned which can handle a 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

[0069] 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.

[0070] **FIG. 1** illustrates an architecture of a prior art video gaming system.

[0071] **FIGS. 2a-b** illustrate a high level system architecture according to one embodiment.

[0072] **FIG. 3** illustrates actual, rated, and required data rates for communication between a client and a server.

[0073] **FIG. 4a** illustrates a hosting service and a client employed according to one embodiment.

[0074] **FIG. 4b** illustrates exemplary latencies associated with communication between a client and hosting service.

[0075] **FIG 4c** illustrates a client device according to one embodiment.

[0076] **FIG 4d** illustrates a client device according to another embodiment.

[0077] **FIG 4e** illustrates an example block diagram of the client device in Figure 4c.

[0078] **FIG 4f** illustrates an example block diagram of the client device in Figure 4d.

[0079] **FIG. 5** illustrates an example form of video compression which may be employed according to one embodiment.

[0080] **FIG. 6a** illustrates an example form of video compression which may be employed in another embodiment.

[0081] **FIG. 6b** illustrates peaks in data rate associated with transmitting a low complexity, low action video sequence.

[0082] **FIG. 6c** illustrates peaks in data rate associated with transmitting a high complexity, high action video sequence.

[0083] **FIGS. 7a-b** illustrate example video compression techniques employed in one embodiment.

[0084] **FIG. 8** illustrates additional example video compression techniques employed in one embodiment.

[0085] **FIGS. 9a-c** illustrate frame rate processing techniques employed in one embodiment of the invention.

[0086] **FIGS. 10a-b** illustrate one embodiment which efficiently packs image tiles within packets.

[0087] **FIGS. 11a-d** illustrate embodiments which employ forward error correction techniques.

[0088] **FIG. 12** illustrates one embodiment which uses multi-core processing units for compression.

[0089] **FIGS. 13a-b** illustrate geographical positioning and communication between hosting services according to various embodiments.

[0090] **FIG. 14** illustrates exemplary latencies associated with communication between a client and a hosting service.

[0091] **FIG. 15** illustrates an example hosting service server center architecture.

[0092] **FIG. 16** illustrates an example screen shot of one embodiment of a user interface which includes a plurality of live video windows.

[0093] **FIG. 17** illustrates the user interface of Figure 16 following the selection of a particular video window.

[0094] **FIG. 18** illustrates the user interface of Figure 17 following zooming of the particular video window to full screen size.

[0095] **FIG. 19** illustrates an example collaborative user video data overlaid on the screen of a multiplayer game.

[0096] **FIG. 20** illustrates an example user page for a game player on a hosting service.

[0097] **FIG. 21** illustrates an example 3D interactive advertisement.

[0098] **FIG. 22** illustrates an example sequence of steps for producing a photoreal image having a textured surface from surface capture of a live performance.

[0099] **FIG. 23** illustrates an example user interface page that allows for selection of linear media content.

[0100] **FIG. 24** is a graph that illustrates the amount of time that elapses before the web page is live versus connection speed.

[0101]FIGS. 25a-b illustrates embodiments of the invention which employ a feedback channel from the client device to the hosting service.

[0102]FIGS. 26a-b illustrate an embodiment in which encodes tiles/frames based on the last known tile/frame to have been successfully received.

[0103] 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.

[0104] FIG. 28 illustrates one embodiment in which the state of a game or application is ported using difference data.

[0105] FIG. 29 illustrates one embodiment of the invention which employs a temporary decoder on the client device.

[0106] FIG. 30 illustrates how "I tiles" are interspersed across "R frames" according to one embodiment of the invention.

[0107] FIGS. 31a-h illustrate embodiments of the invention which generate a live stream and/or one or more HQ streams.

DESCRIPTION OF EXAMPLE EMBODIMENTS

[0108] 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.

[0109] 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.

[0110] 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.

[0111] 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.

[0112] 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 use 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.

[0113] 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)).

[0114] 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 collects a subscription fee directly from the user and pays 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.

[0115] **COMPRESSED VIDEO CHARACTERISTICS**

[0116] 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.

[0117] 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.

[0118] 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.

[0119] **HOSTING SERVICE ARCHITECTURE**

[0120] 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.

[0121] 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.

[0122] 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.

[0123] 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.

[0124] 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.

[0125] 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 compression 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.

[0126] 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. It 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.

[0127] 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.

[0128] 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.

[0129] 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 is 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.

[0130] 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.

[0131] 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.

[0132] For audio, the client 475 has left/right analog stereo RCA jacks, and for digital audio output it has a TOSLINK output.

[0133] In addition to a Bluetooth wireless interface to input devices 479, it also has USB jacks to interface to input devices.

[0134] 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.

[0135] 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.

[0136] 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.

[0137] Bluetooth interface 484 communicates to input devices 469 wirelessly through its antenna, internal to client 465.

[0138] 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.

[0139] 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.

[0140] 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).

[0141] 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

[0142] 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.

[0143] CPU 483 communicates with and configures the additional devices.

[0144] 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.

[0145] 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.

[0146] 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.

[0147] 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.

[0148] **ROUND-TRIP LATENCY ANALYSIS**

[0149] 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.

[0150] 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).

[0151] 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.

[0152] 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.

[0153] 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.

[0154] 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.

[0155] 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.

[0156] 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.

[0157] 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 experience by a user of the system shown in Figure 4a. They are: 1+1+25+22+1+16+6+8 = 80ms. 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.

[0158] 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.

[0159] **LOW-LATENCY VIDEO COMPRESSION**

[0160] 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.

[0161] 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).

[0162] 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

640x480x24bits/pixel = 640*480*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 KB * 8 bits * 60 frames/sec = 42.2Mbps, 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.

[0163] 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.

[0164] 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.

[0165] 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 Kb + 16 Kb * 59 = 1.1Mbps. 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.

[0166] 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.

[0167] 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 $1/60^{th}$ second duration each, or 500ms of latency. Clearly that is far too long.

[0168] 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 160Kb/2.2Mb=71ms 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.

[0169] 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.

[0170] 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).

[0171] 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 increasingly latency).

[0172] 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.

[0173] 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.

[0174] 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.

[0175] 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.

[0176] 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 $1/16^{th}$ 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" $1/16^{th}$ 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., I_{15}). 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.

[0177] 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, I_0 and P_0 from R frames 711 to 770 are used to decompress and render tile 0 of the video image. Similarly, I_1 and P_1 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.

[0178] 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 10Kb + 15 * 1Kb = 25Kb. So, each 60-frame sequence would be 25Kb * 60 = 1.5Mbps. 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.

[0179] 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.

[0180] 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 10Kb + 15 * 0.75Kb = 21.25Kb, or at 60 frames per second, the data rate would be 21.25Kb * 60 = 1.3Mbps, 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.

[0181] 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.

[0182] 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 15^{th} frame time (not shown in Figure 7b) and during the 16^{th} 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).

[0183] 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.

[0184] 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.

[0185] 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.

[0186] 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.

[0187] 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.

[0188] 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).

[0189] 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.

[0190] 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.

[0191] 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.

[0192] 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 $1/60^{th}$ 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 ($1/60^{th}$ 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.

[0193] 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).

[0194] 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.

[0195] 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.

[0196] 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.

[0197] 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.

[0198] 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., 100Mbs Ethernet or 802.11g wireless) or on a private network (e.g., 100Mbps connection between a data center an 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.

[0199] **DATA RATE ADJUSTMENT**

[0200] 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.

[0201] 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.

[0202] **DROPPED OR DELAYED PACKETS**

[0203] 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.

[0204] 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.

[0205] 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).

[0206] 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.

[0207] 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.

[0208] 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 $1/60^{th}$ 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.

[0209] 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.

[0210] 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).

[0211] 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.

[0212] 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.

[0213] 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.

[0214] 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.

[0215] 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.

[0216] 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.

[0217] 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.

[0218] **VIDEO AND AUDIO COMPRESSOR/DECOMPRESSOR IMPLEMENTATION**

[0219] 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.

[0220] 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.

[0221] 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.

[0222] 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.

[0223] 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.

[0224] 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.

[0225] 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.

[0226] **HOSTING SERVICE SERVER CENTER DISTRIBUTION**

[0227] 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 = 80ms. 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.

[0228] 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.

[0229] 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.

[0230] 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.

[0231] 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.

[0232] 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.

[0233] 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.

[0234] 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 it 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.

[0235] 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.

[0236] **HOSTING SERVICE FEATURES**

[0237] 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.

[0238] 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.

[0239] 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.

[0240] 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.

[0241] 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.

[0242] 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.

[0243] 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.

[0244] 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.

[0245] 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.

[0246] 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.

[0247] **LIVE, INSTANTLY-VIEWABLE, INSTANTLY-PLAYABLE GAMES**

[0248] 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.

[0249] 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.

[0250] 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.

[0251] 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.

[0252] 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.

[0253] 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).

[0254] 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.

[0255] 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.

[0256] 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.

[0257] Thus, the game will start up almost instantly, it will run in an ideal environment, and subsequent segments will load almost instantly.

[0258] 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.

[0259] 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.

[0260] 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.

[0261] 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.

[0262] 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.

[0263] **MULTIPLE USER COLLABORATION**

[0264] 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.

[0265] 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.

[0266] 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.

[0267] 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.

[0268] 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 develop 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.

[0269] 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.

[0270] 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.

[0271] **VIDEO-RICH COMMUNITY SERVICES**

[0272] 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).

[0273] 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.

[0274] 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.

[0275] 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.

[0276] 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.

[0277] 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.

[0278] 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.

[0279] 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 the review the Brag Clip).

[0280] 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.

[0281] 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.

[0282] 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".

[0283] 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.

[0284] **INTERACTIVE ANIMATED ADVERTISEMENTS**

[0285] 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).

[0286] 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.

[0287] 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.

[0288] 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.

[0289] **STREAMING GEOMETRY DURING REAL-TIME ANIMATION**

[0290] 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.)

[0291] 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 reasonable well predicted and the mass storage devices can seek in advance to the location where the upcoming geometry is located.

[0292] 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.

[0293] 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.

[0294] Thus, with sufficiently high network speed, and sufficiently low enough latency mass storage, geometry can be streamed into app/game 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.

[0295] 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.

[0296] 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.

[0297] 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.

[0298] **INTEGRATION OF LINEAR CONTENT WITH INTERACTIVE FEATURES**

[0299] 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.

[0300] 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.

[0301] 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).

[0302] 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.

[0303] 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.

[0304] 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.

[0305] 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.

[0306] 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.

[0307] 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.

[0308] 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.

[0309] 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 a 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.

[0310] **MULTIPLAYER GAMES**

[0311] To the extent a game is a multiplayer game, then it will be able 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.

[0312] 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.

[0313] **TOP PLAYER GROUPINGS AND TOURNAMENTS**

[0314] 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 is 210 is able to broadcast the major tournaments to thousands or more of people in the audience at once.

[0315] 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.

[0316] A similar level of efficiency can be gained when a CDN is used to show Game Finder pages of top player groupings.

[0317] 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.

[0318] **ACCELERATION OF WEB PAGE LOADING**

[0319] 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).

[0320] 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.

[0321] 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.

[0322] 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.

[0323] 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.

[0324] 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 files * 100ms = 5.4 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.

[0325] 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.

[0326] 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.

[0327] 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.

[0328] 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.

[0329] **FACILITATING DEBUGGING OF GAMES AND APPLICATIONS**

[0330] 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.

[0331] 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.

[0332] 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.

[0333] 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.

[0334] **RESOURCE SHARING AND COST SAVINGS**

[0335] 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*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.

[0336] 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.

[0337] 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 a 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.

[0338] 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*5 days=35 hours/week) and weekend (8AM-12AM, 16*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.

[0339] 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.

[0340] 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.

[0341] 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.

[0342] **ELIMINATING THE NEED TO UPGRADE**

[0343] 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.

[0344] **ELIMINATING THE NEED FOR BACKUPS**

[0345] 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.

[0346] **ACCESS TO DEMOS**

[0347] 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.

[0348] 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.

[0349] **BENEFITS TO SCHOOLS, BUSINESSES AND OTHER INSTITUTIONS**

[0350] 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 a 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.

[0351] 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.

[0352] 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.

[0353] 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.

[0354] **ELIMINATION OF PIRACY**

[0355] 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.

[0356] 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).

[0357] **STREAMING INTERACTIVE VIDEO**

[0358] 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.

[0359] 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.

[0360] 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.

[0361] 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.

[0362] NON-TRANSIT PEERING

[0363]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).

[0364]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.

[0365] **VIDEO COMPRESSION USING FEEDBACK**

[0366] 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.

[0367] 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.

[0368] 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.

[0369] 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.

[0370]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.

[0371] 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).

[0372] **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.

[0373] 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.

[0374] 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.

[0375] 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.

[0376] 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.

[0377] 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.

[0378] 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.

[0379]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 a 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.

[0380]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 $1/8^{th}$ of the tile/frame and the compressed size is larger than $1/8^{th}$ 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.

[0381]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 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.

[0382]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.

[0383]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.

[0384] STATE MANAGEMENT IN AN ONLINE GAMING SYSTEM

[0385] 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 exemplary system architecture for implementing this embodiment and **Figure 27b** illustrates an exemplary 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.

[0386] 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.

[0387] 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).

[0388] 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.

[0389] 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)).

[0390] Other platform-dependent data related to the client/user may include (but is not limited to) the following:

[0391] 1. The user's screen resolution. When the user resumes the game, the user may be using a different device with a different resolution.

[0392] 2. The user's controller configuration. When game resumes, the user may have switched from a game controller to a keyboard/mouse.

[0393] 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).

[0394] 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.

[0395] 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.

[0396] **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.

[0397] 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.

[0398] **TECHNIQUES FOR MAINTAINING A CLIENT DECODER**

[0399] 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.

[0400] 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).

[0401] 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.

[0402] 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.

[0403] 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).

[0404] 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.

[0405] 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).

[0406] 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).

[0407] 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).

[0408] 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.

[0409] 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.

[0410] 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.

[0411] **COMPRESSION USING STANDARDS-BASED TILE SIZES**

[0412] 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.

[0413] 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).

[0414] 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.

[0415] **TECHNIQUES FOR STREAM REWIND AND PLAYBACK OPERATIONS**

[0416] 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 resampling 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.

[0417] 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).

[0418] 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).

[0419] 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.

[0420] 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. 1/60th of a second for a 60 frame/sec stream), then 12 (frames)/60 (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 fames in 6 frames, which is just 1/10th 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 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.

[0421]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.

[0422]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.

[0423]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.

[0424]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.

[0425] 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.

[0426] 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).

[0427] 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.

[0428]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.

[0429]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 an 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.

[0430]

[0431] As a result of the techniques described above, the user receives the benefits of both Live and HQ video stream during online gameplay or other online interaction, without suffering from any of the limitations of either a Live stream or an HQ stream.

[0432]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 could be used, providing multiple simultaneous HQ encodings at a variety of resolutions.

[0433] 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.

[0434]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).

[0435]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.

[0436]An advantage of the configuration of **Figure 31c** is that App/Game Servers 3121-3125 that require smaller that full-size video windows will not need to process and decompress a full-size window. Also, App/Game Services 3121-3125that 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.

[0437]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.

[0438]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 stream is 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.

[0439]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 cary dozens of 10Mbps video streams, with little apparent reduction in quality.

[0440]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.

[0441]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.

[0442]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 is 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, and Non-scaled Video 3196 also is 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**.

[0443]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 undelayed. 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**.

[0444]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.

[0445]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 gameplayers 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.

[0446]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 are any number of other restrictions that would 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).

[0447]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.

[0448]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.

[0449]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.

[0450] 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.

[0451] 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.

[0452] 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, have been left out of some or all of the figures to avoid obscuring the pertinent aspects.

[0453] 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, propagation media or other type of machine-readable media suitable for storing electronic instructions. For example, the present invention may be downloaded as a computer program which may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[0454] 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, propagation media or other type of media/machine-readable medium suitable for storing electronic instructions. For example, elements of the disclosed subject matter may be downloaded as a computer program product, wherein the program may be transferred from a remote computer or electronic device to a requesting process by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[0455] 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 a restrictive sense.

CLAIMS

What is claimed:

1. A computer-implemented method for performing video compression comprising:

encoding a first plurality of video frames or portions thereof, wherein each encoded video frame or portion thereof is dependent on a previously-encoded video frame or portion thereof, respectively;

transmitting the first plurality of encoded video frames or portions to a client device;

receiving feedback information from the client device, the feedback information usable to determine whether data contained in the video frames or portions has not been successfully received and/or decoded;

in response to detecting that a video frame or portion thereof has not been successfully received and/or decoded, encoding a current video frame or portion thereof to be dependent on a previously-encoded video frame or portion thereof known to have been successfully received and/or decoded on the client device; and

transmitting the current video frame or portion thereof to the client device.

2. The method as in claim 1 wherein the previously-encoded video frame or portion thereof known to have been successfully received and/or decoded comprises the last known video frame or portion thereof known to have been successfully received and/or decoded.

3. The method as in claim 1 wherein the video frames or portions thereof comprise P-frames or P-tiles, respectively.

4. The method as in claim 1 wherein the feedback information comprises an indication that the frames or portions thereof have been successfully received and/or decoded at the client device.

5. The method as in claim 1 wherein the feedback information comprises an indication that the frames or portions thereof have not been successfully received and/or decoded at the client device.

6. The method as in claim 1 wherein encoding a current video frame or portion thereof to be dependent on a previously-encoded video frame or portion thereof known to have been successfully received and/or decoded on the client device further comprises:

retrieving a prior state from a memory, the prior state comprising a state of an encoder after encoding the previously-encoded video frame or portion.

7. The method as in claim 6 further comprising:

decoding the current video frame or portion thereof, wherein decoding comprises retrieving a prior state from a memory, the prior state comprising a state of a decoder after decoding the previously-encoded video frame or portion.

8. The method as in claim 1 further comprising:

decoding the encoded video frames or portion thereof at the client device; and

displaying images associated with each of the video frames or portions thereof on a display on the client device.

9. A system comprising a memory for storing program code and a processor for processing the program code to perform the operations of:

encoding a first plurality of video frames or portions thereof, wherein each encoded video frame or portion thereof is dependent on a previously-encoded video frame or portion thereof, respectively;

transmitting the first plurality of encoded video frames or portions to a client device;

receiving feedback information from the client device, the feedback information usable to determine whether data contained in the video frames or portions has not been successfully received and/or decoded;

in response to detecting that a video frame or portion thereof has not been successfully received and/or decoded, encoding a current video frame or portion thereof to be dependent on a previously-encoded video frame or portion thereof known to have been successfully received and/or decoded on the client device; and

transmitting the current video frame or portion thereof to the client device.

10. The system as in claim 9 wherein the previously-encoded video frame or portion thereof known to have been successfully received and/or decoded comprises the last known video frame or portion thereof known to have been successfully received and/or decoded.

11. The system as in claim 9 wherein the video frames or portions thereof comprise P-frames or P-tiles, respectively.

12. The system as in claim 9 wherein the feedback information comprises an indication that the frames or portions thereof have been successfully received and/or decoded at the client device.

13. The system as in claim 9 wherein the feedback information comprises an indication that the frames or portions thereof have not been successfully received and/or decoded at the client device.

14. The system as in claim 9 wherein encoding a current video frame or portion thereof to be dependent on a previously-encoded video frame or portion thereof known to have been successfully received and/or decoded on the client device further comprises:

retrieving a prior state from a memory, the prior state comprising a state of an encoder after encoding the previously-encoded video frame or portion.

15. The system as in claim 14 comprising additional program code to cause the processor to perform the operations of:

decoding the current video frame or portion thereof, wherein decoding comprises retrieving a prior state from a memory, the prior state comprising a state of a decoder after decoding the previously-encoded video frame or portion.

16. The system as in claim 9 comprising additional program code to cause the processor to perform the operations of:

decoding the encoded video frames or portion thereof at the client device; and

displaying images associated with each of the video frames or portions thereof on a display on the client device.

17. A machine-readable medium having program code stored thereon which, when executed by a machine, causes the machine to perform the operations of:

encoding a first plurality of video frames or portions thereof, wherein each encoded video frame or portion thereof is dependent on a previously-encoded video frame or portion thereof, respectively;

transmitting the first plurality of encoded video frames or portions to a client device;

receiving feedback information from the client device, the feedback information usable to determine whether data contained in the video frames or portions has not been successfully received and/or decoded;

in response to detecting that a video frame or portion thereof has not been successfully received and/or decoded, encoding a current video frame or portion thereof to be dependent on a previously-encoded video frame or portion thereof known to have been successfully received and/or decoded on the client device; and

transmitting the current video frame or portion thereof to the client device.

18. The machine-readable medium as in claim 17 wherein the previously-encoded video frame or portion thereof known to have been successfully received and/or decoded comprises the last known video frame or portion thereof known to have been successfully received and/or decoded.

19. The machine-readable medium as in claim 17 wherein the video frames or portions thereof comprise P-frames or P-tiles, respectively.

20. The machine-readable medium as in claim 17 wherein the feedback information comprises an indication that the frames or portions thereof have been successfully received and/or decoded at the client device.

21. The machine-readable medium as in claim 17 wherein the feedback information comprises an indication that the frames or portions thereof have not been successfully received and/or decoded at the client device.

22. The machine-readable medium as in claim 17 wherein encoding a current video frame or portion thereof to be dependent on a previously-encoded video frame or portion thereof known to have been successfully received and/or decoded on the client device further comprises:

retrieving a prior state from a memory, the prior state comprising a state of an encoder after encoding the previously-encoded video frame or portion.

23. The machine-readable medium as in claim 22 comprising additional program code to cause the machine to perform the operations of:

decoding the current video frame or portion thereof, wherein decoding comprises retrieving a prior state from a memory, the prior state comprising a state of a decoder after decoding the previously-encoded video frame or portion.

24. The machine-readable medium as in claim 17 comprising additional program code to cause the processor to perform the operations of:

decoding the encoded video frames or portion thereof at the client device; and

displaying images associated with each of the video frames or portions thereof on a display on the client device.

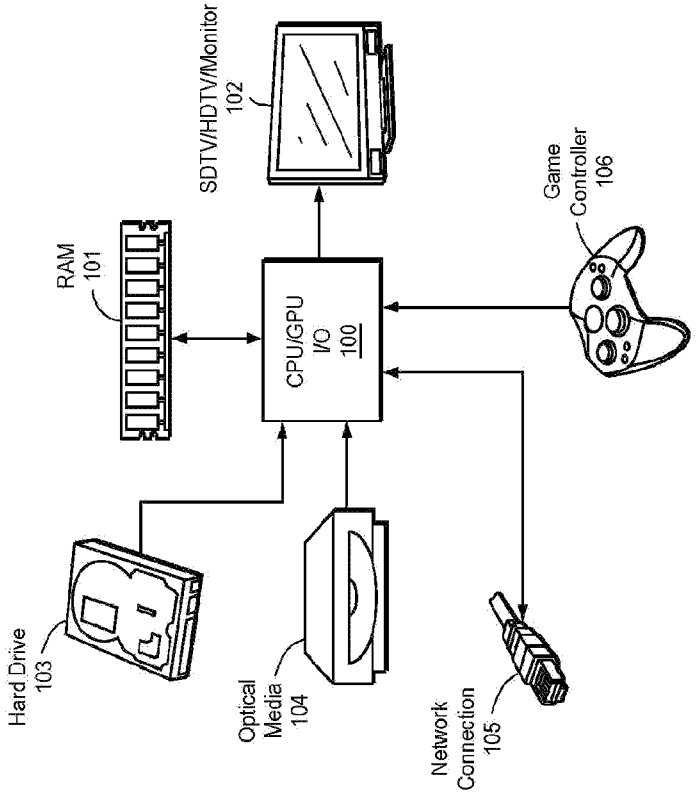


FIG. 1
(Prior Art)

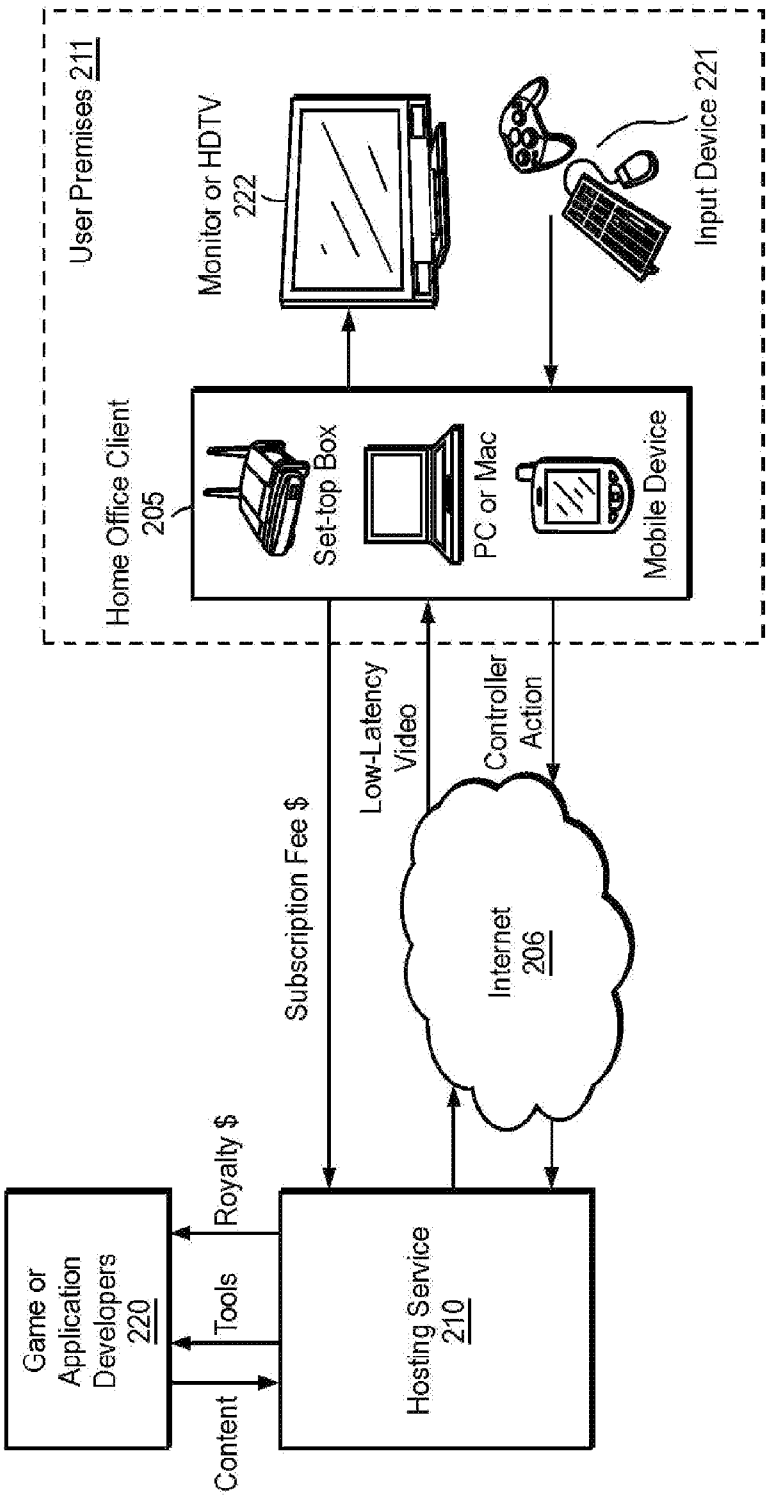

User Premises 211
Monitor or HDTV 222
Input Device 221
Home Office Client 205
Set-top Box
PC or Mac
Mobile Device
Subscription Fee $
Low-Latency Video
Controller Action
Internet 206
Royalty $
Tools
Content
Game or Application Developers 220
Hosting Service 210

FIG. 2a

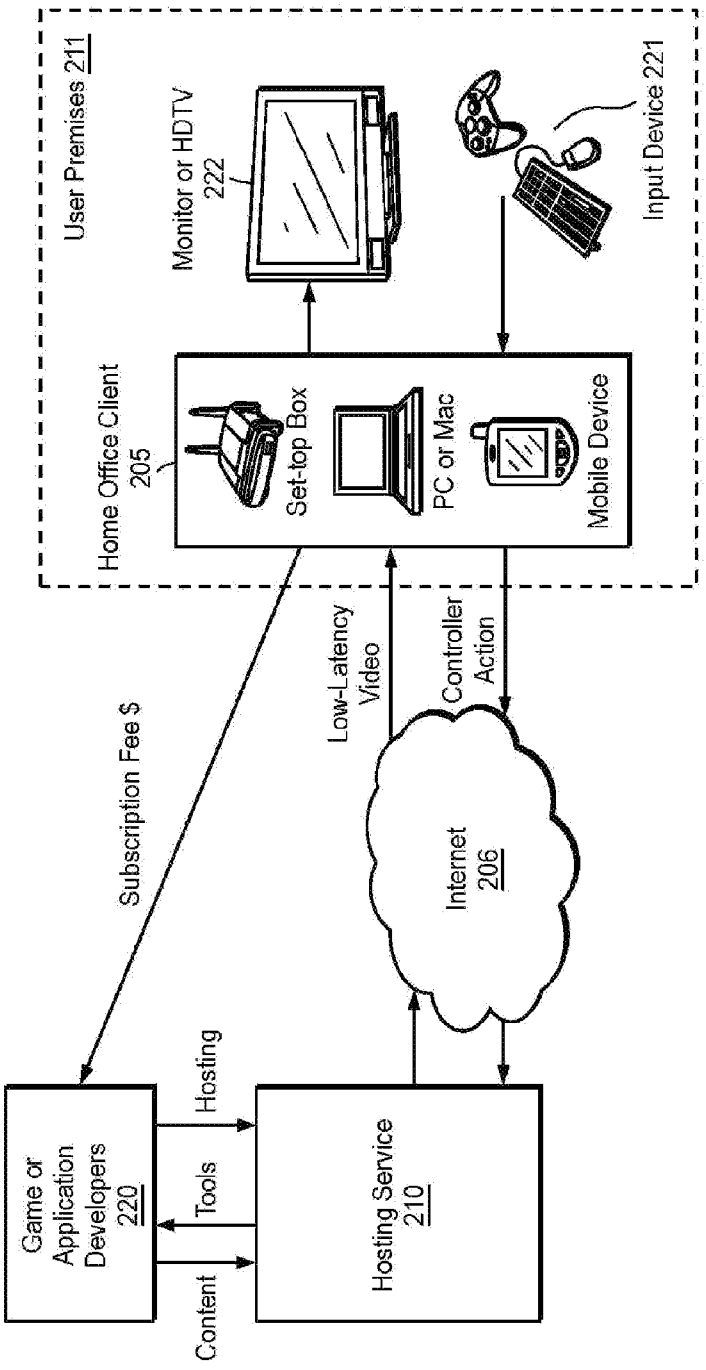

User Premises 211
Monitor or HDTV 222
Input Device 221
Home Office Client 205
Set-top Box
PC or Mac
Mobile Device
Subscription Fee $
Low-Latency Video
Controller Action
Internet 206
Hosting
Tools
Content
Game or Application Developers 220
Hosting Service 210

FIG. 2b

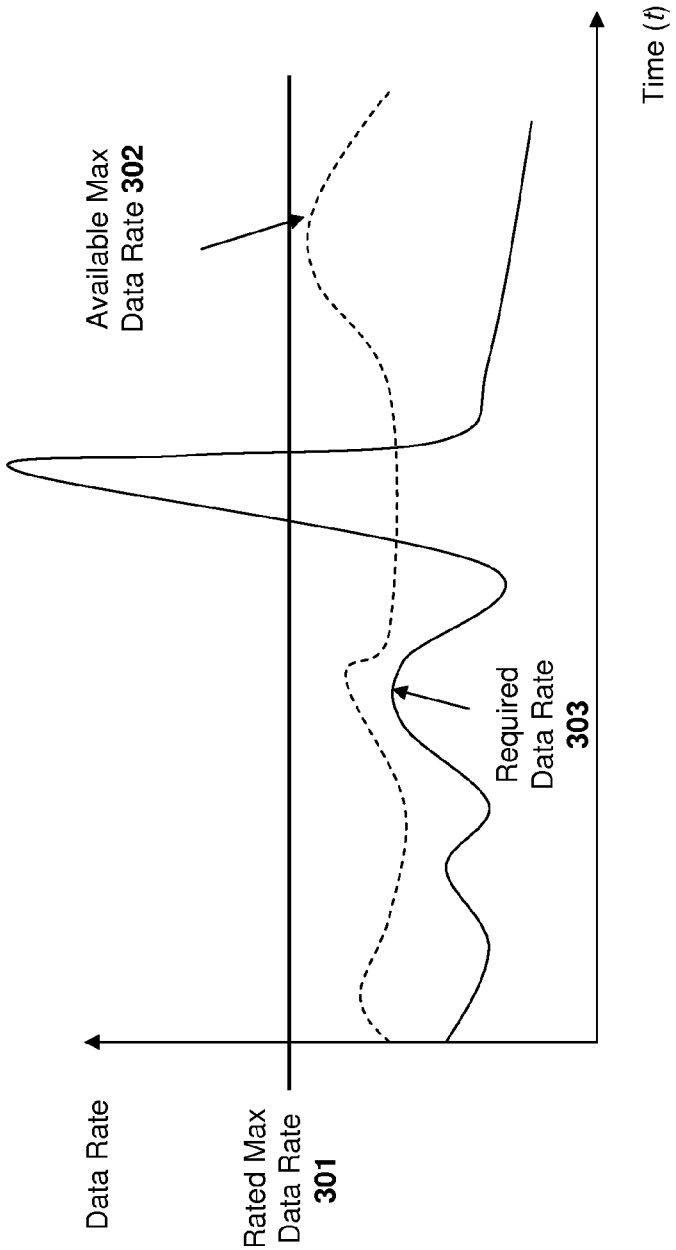

Data Rate
Rated Max
Data Rate
301
Available Max
Data Rate 302
Required
Data Rate
303
Time (t)

Fig. 3

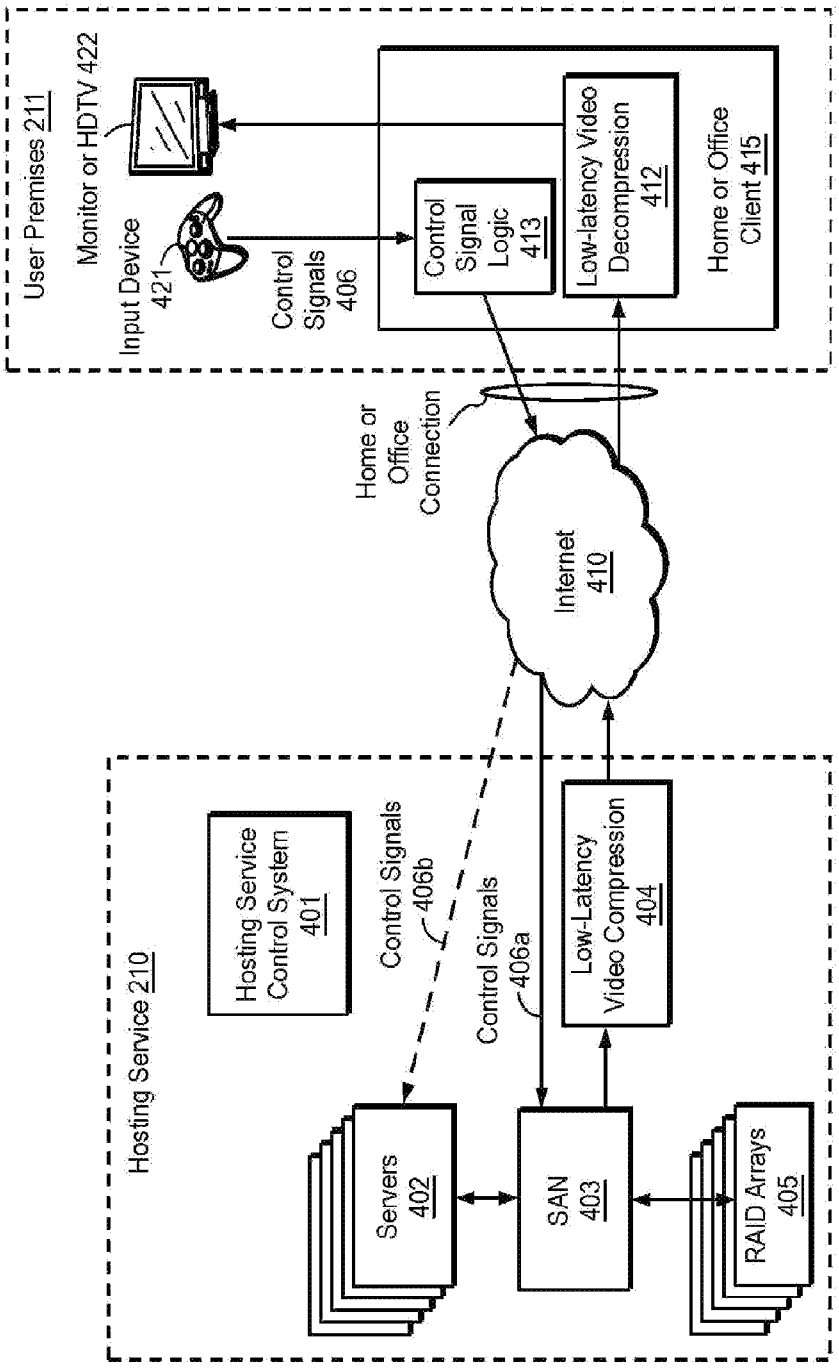

User Premises 211
Monitor or HDTV 422
Input Device 421
Control Signals 406
Control Signal Logic 413
Low-latency Video Decompression 412
Home or Office Client 415
Home or Office Connection
Internet 410
Hosting Service 210
Hosting Service Control System 401
Control Signals 406b
Control Signals 406a
Low-Latency Video Compression 404
Servers 402
SAN 403
RAID Arrays 405

FIG. 4a

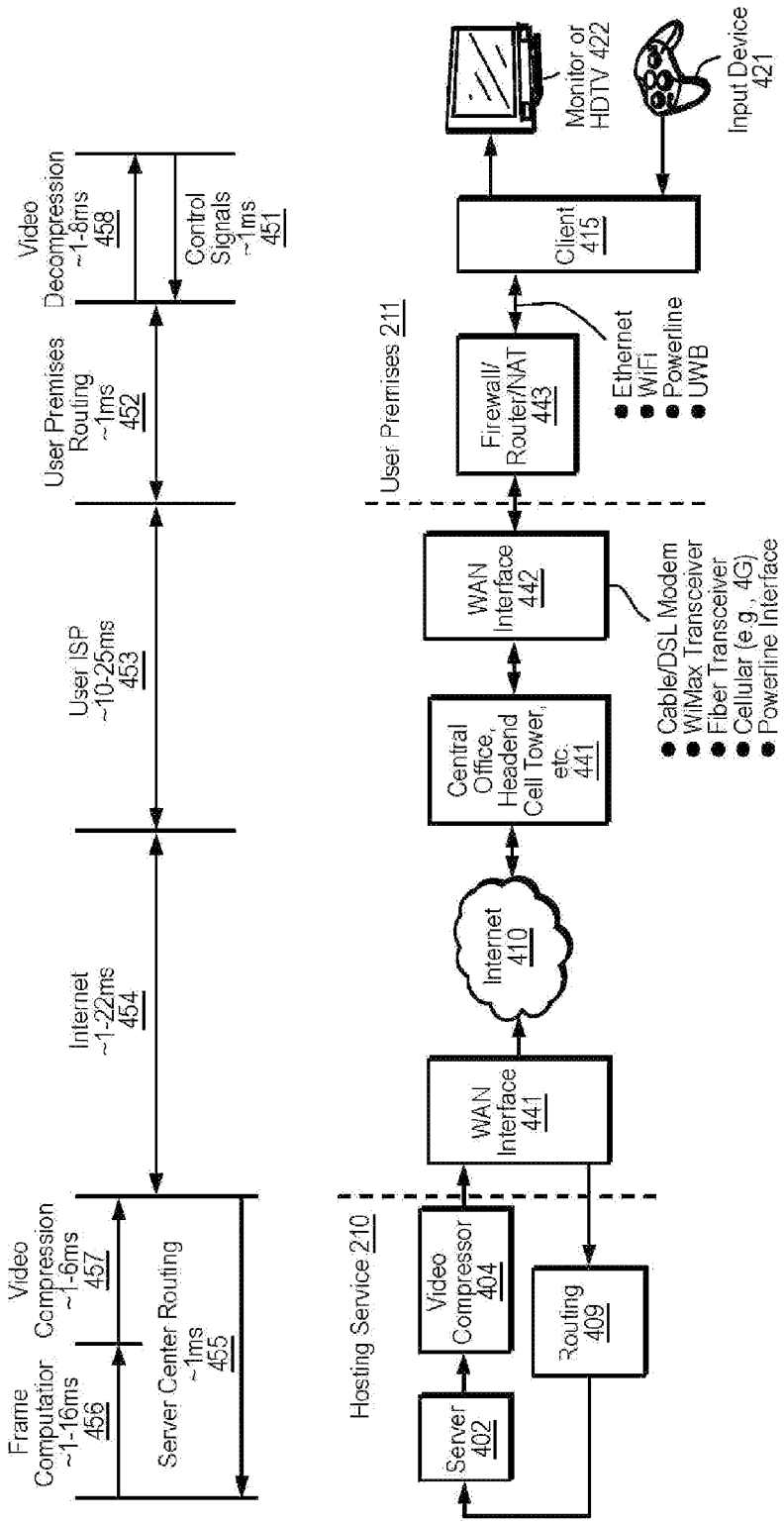

Frame Computation ~1-16ms 456
Video Compression ~1-6ms 457
Server Center Routing ~1ms 455
Internet ~1-22ms 454
User ISP ~10-25ms 453
User Premises Routing ~1ms 452
Video Decompression ~1-8ms 458
Control Signals ~1ms 451
Hosting Service 210
Server 402
Video Compressor 404
Routing 409
WAN Interface 441
Internet 410
Central Office, Headend Cell Tower, etc. 441
WAN Interface 442
Cable/DSL Modem
WiMax Transceiver
Fiber Transceiver
Cellular (e.g., 4G)
Powerline Interface
User Premises 211
Firewall/ Router/NAT 443
Ethernet
WiFi
Powerline
UWB
Client 415
Monitor or HDTV 422
Input Device 421

FIG. 4b

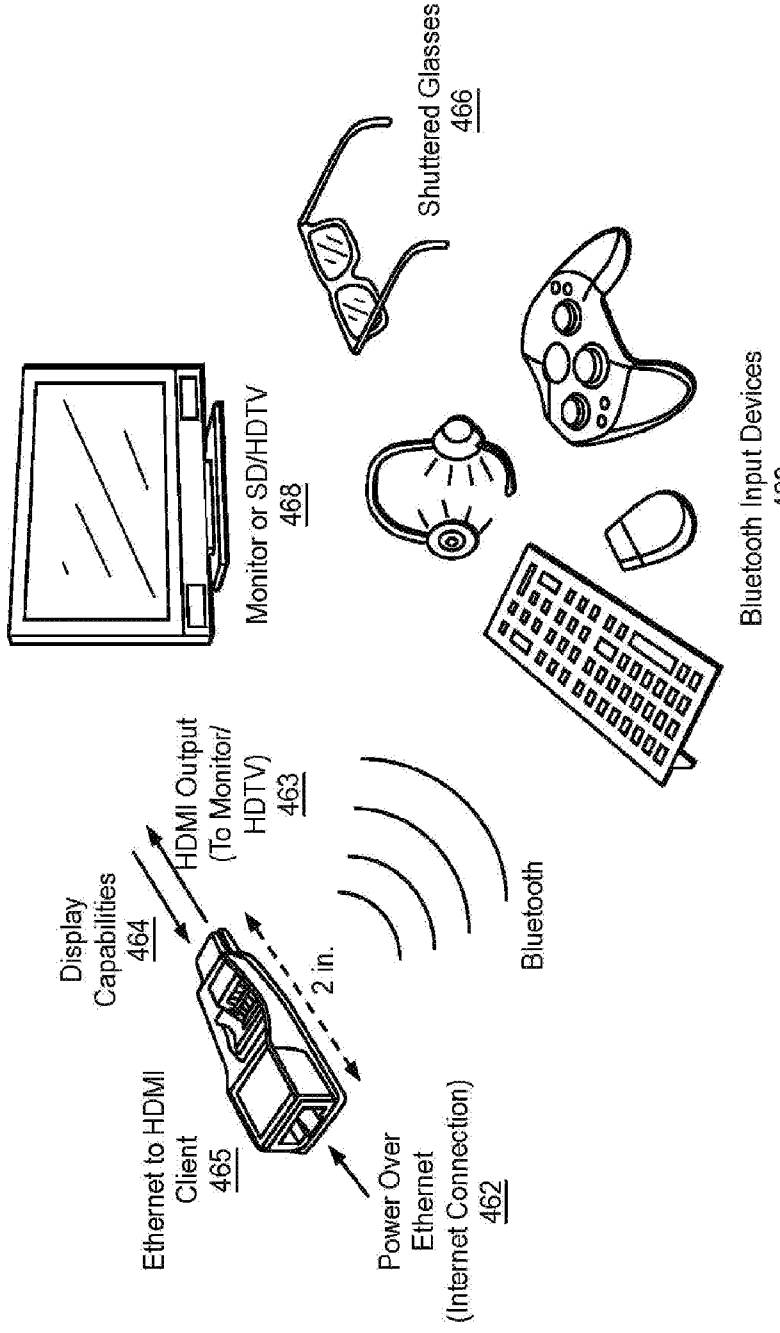

Shuttered Glasses
466
Monitor or SD/HDTV
468
Bluetooth Input Devices
469
HDMI Output
(To Monitor/
HDTV)
463
Display
Capabilities
464
2 in.
Bluetooth
Ethernet to HDMI
Client
465
Power Over
Ethernet
(Internet Connection)
462

FIG. 4c

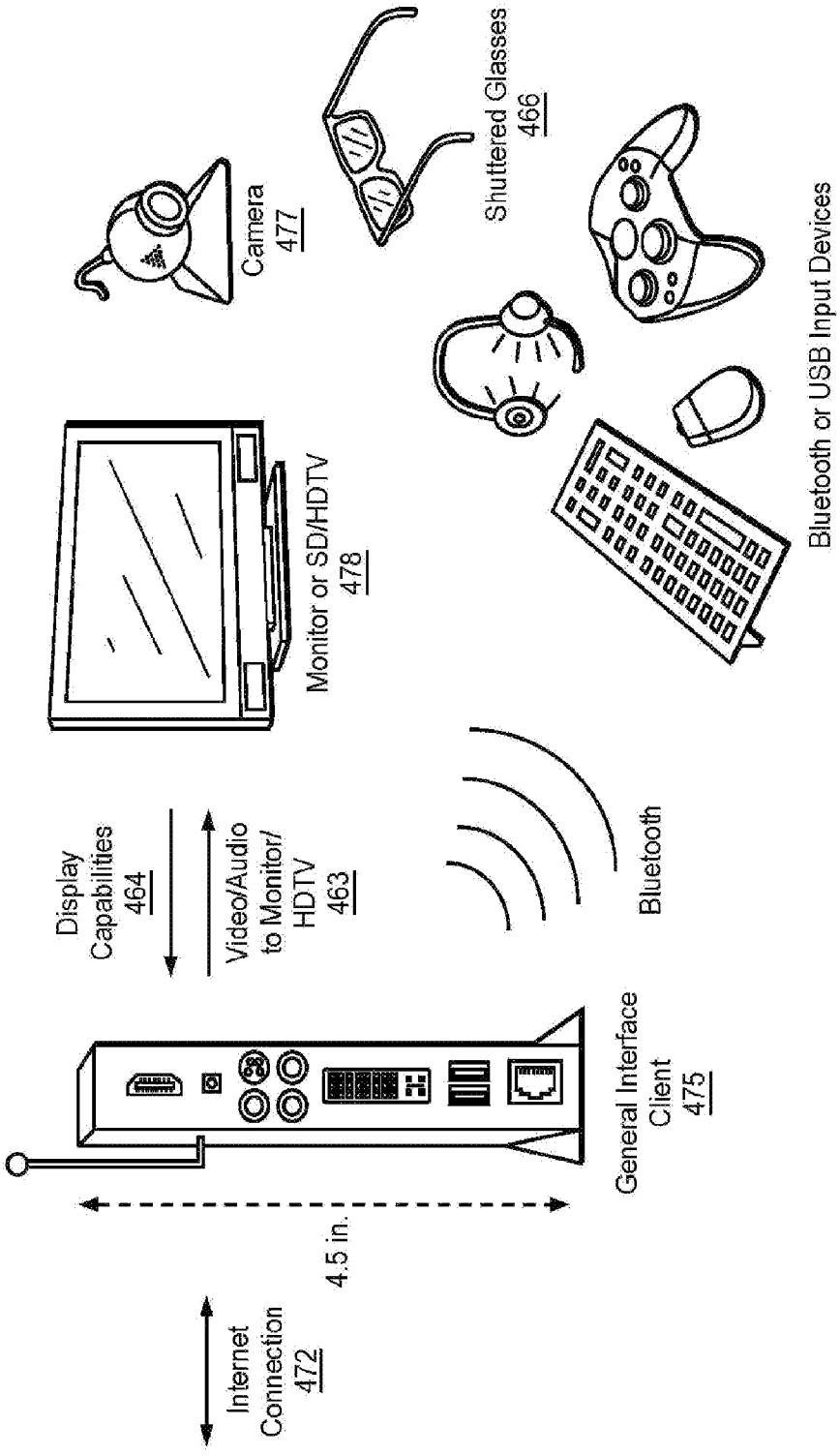

Camera
477
Shuttered Glasses
466
Monitor or SD/HDTV
478
Bluetooth or USB Input Devices
479
Display
Capabilities
464
Video/Audio
to Monitor/
HDTV
463
Bluetooth
General Interface
Client
475
4.5 in.
Internet
Connection
472

FIG. 4d

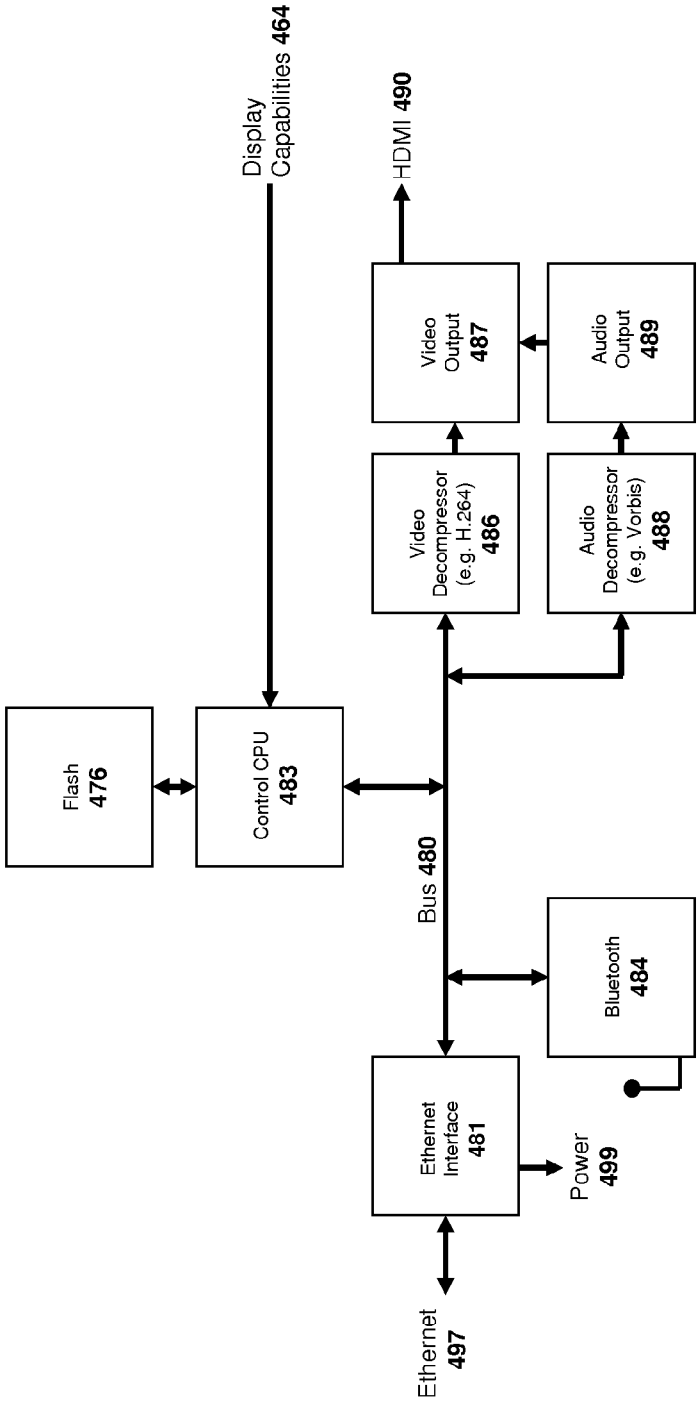

Display Capabilities 464
HDMI 490
Video Output 487
Audio Output 489
Video Decompressor (e.g. H.264) 486
Audio Decompressor (e.g. Vorbis) 488
Flash 476
Control CPU 483
Bus 480
Bluetooth 484
Ethernet Interface 481
Power 499
Ethernet 497

Fig. 4e

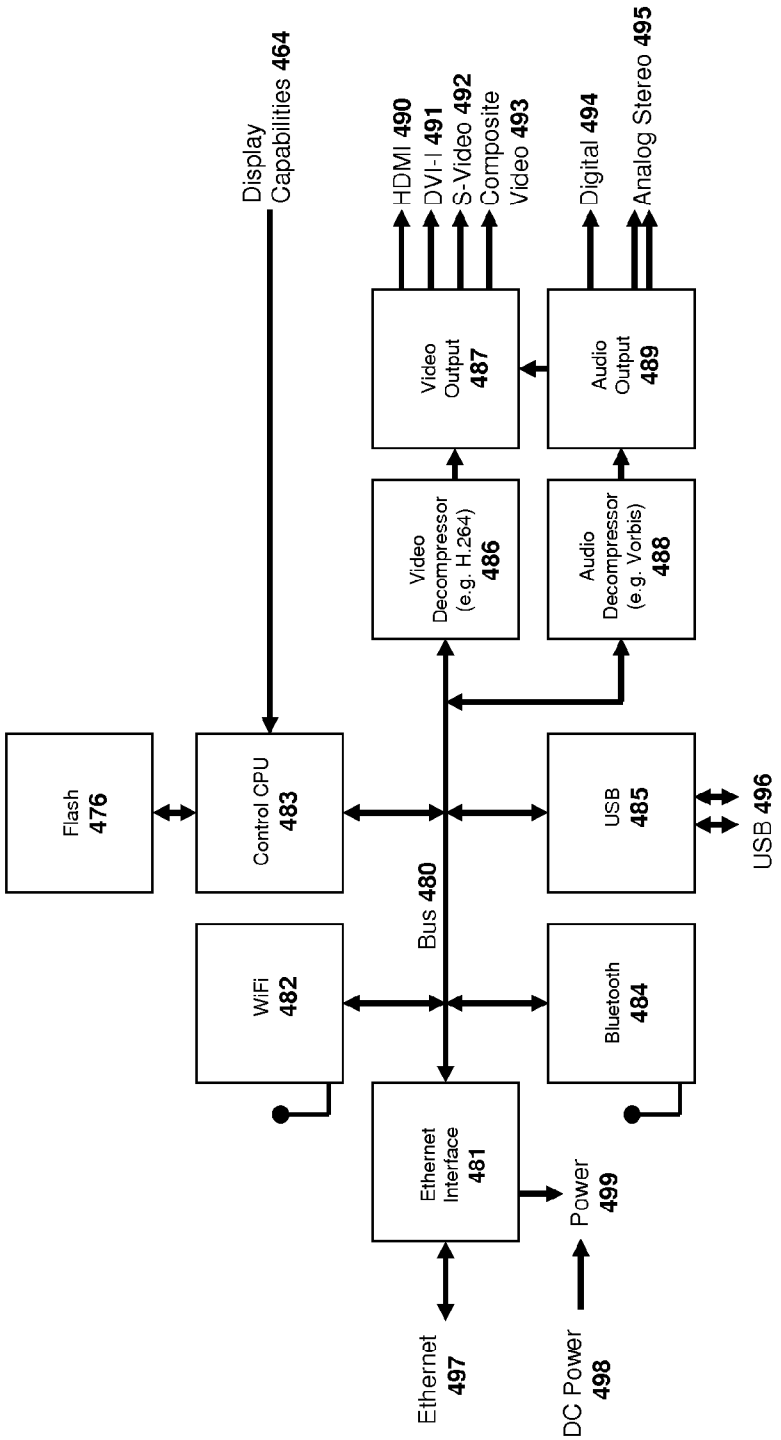

Display Capabilities 464
HDMI 490
DVI-I 491
S-Video 492
Composite Video 493
Digital 494
Analog Stereo 495
Video Output 487
Audio Output 489
Video Decompressor (e.g. H.264) 486
Audio Decompressor (e.g. Vorbis) 488
Flash 476
Control CPU 483
USB 485
USB 496
Bus 480
WiFi 482
Bluetooth 484
Ethernet Interface 481
Power 499
Ethernet 497
DC Power 498

Fig. 4f

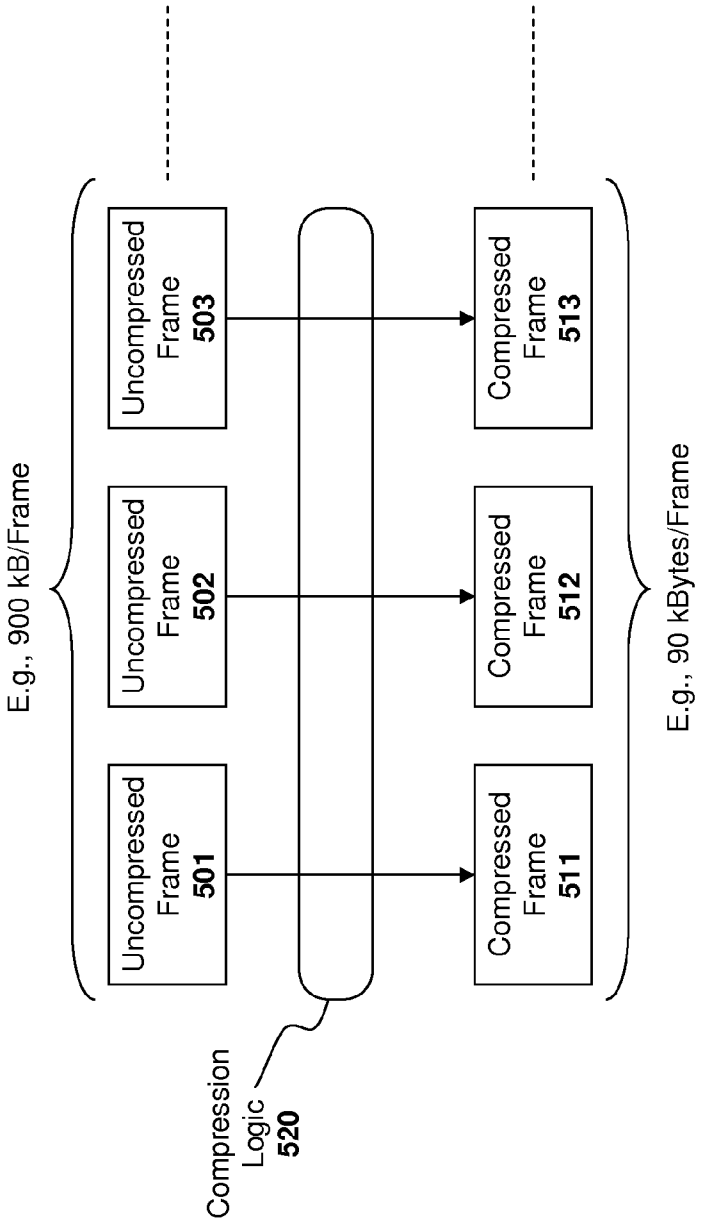


Fig. 5
(prior art)

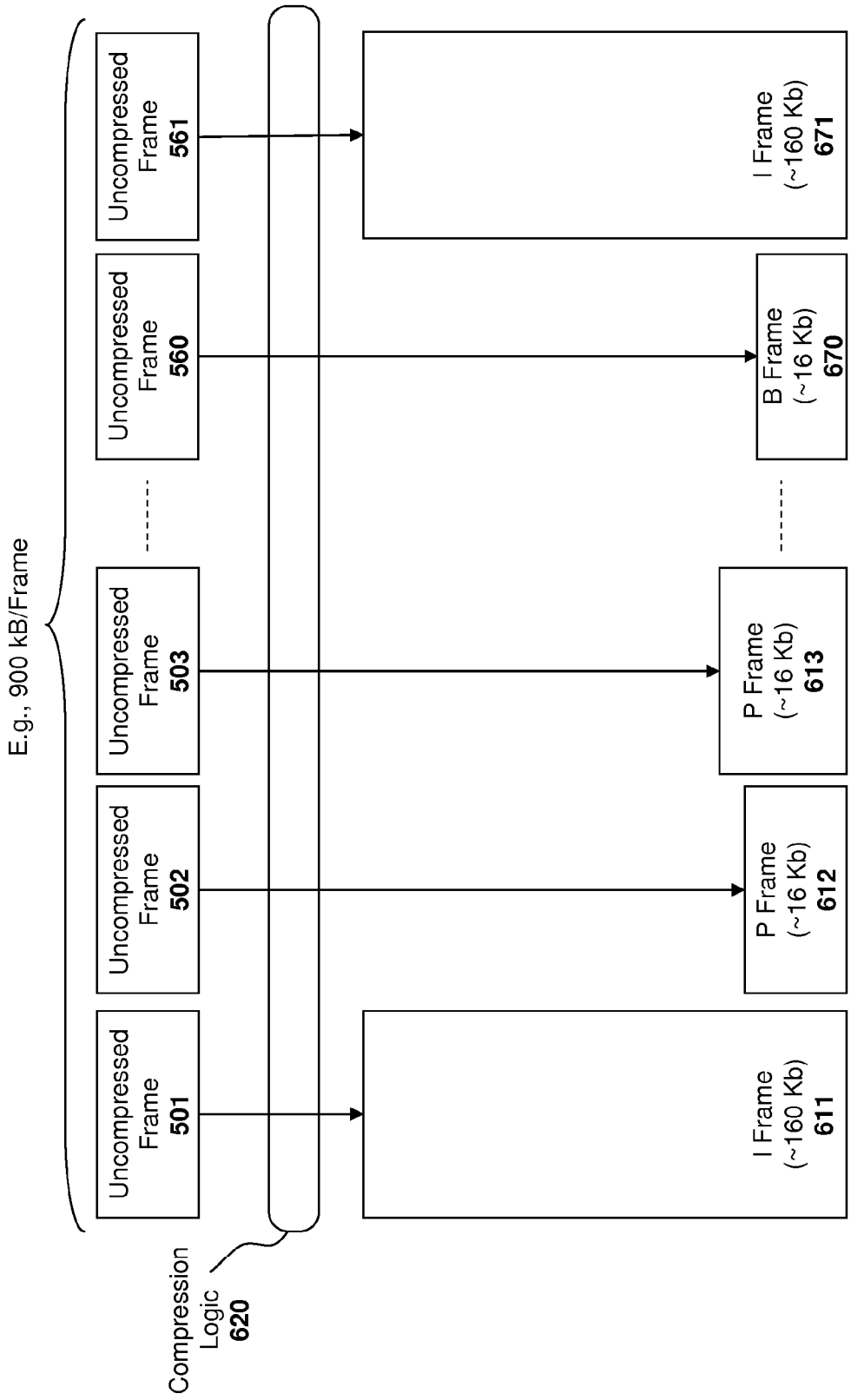


Fig. 6a
(prior art)

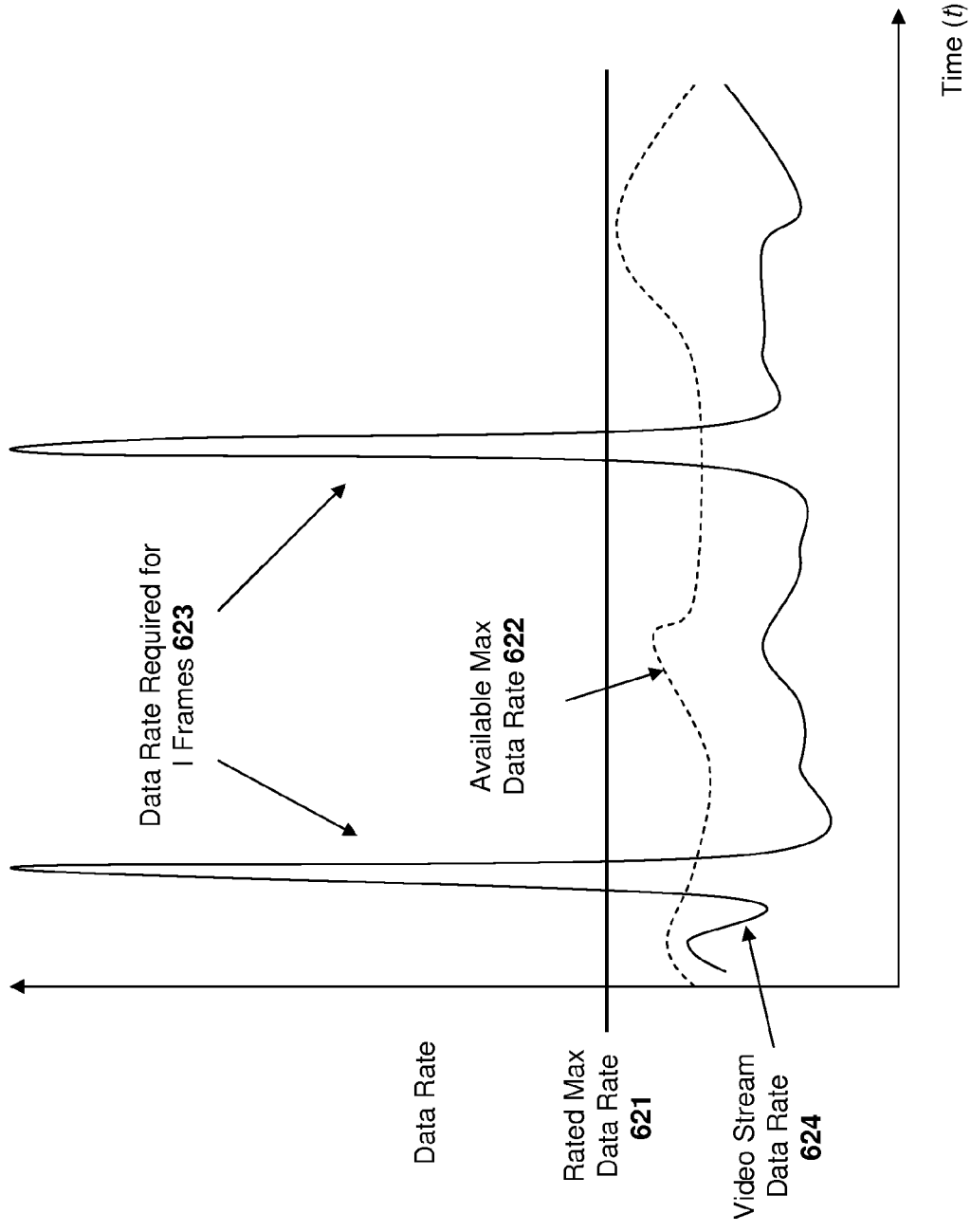

Data Rate Required for I Frames 623
Available Max Data Rate 622
Data Rate
Rated Max Data Rate 621
Video Stream Data Rate 624
Time (t)

Fig. 6b

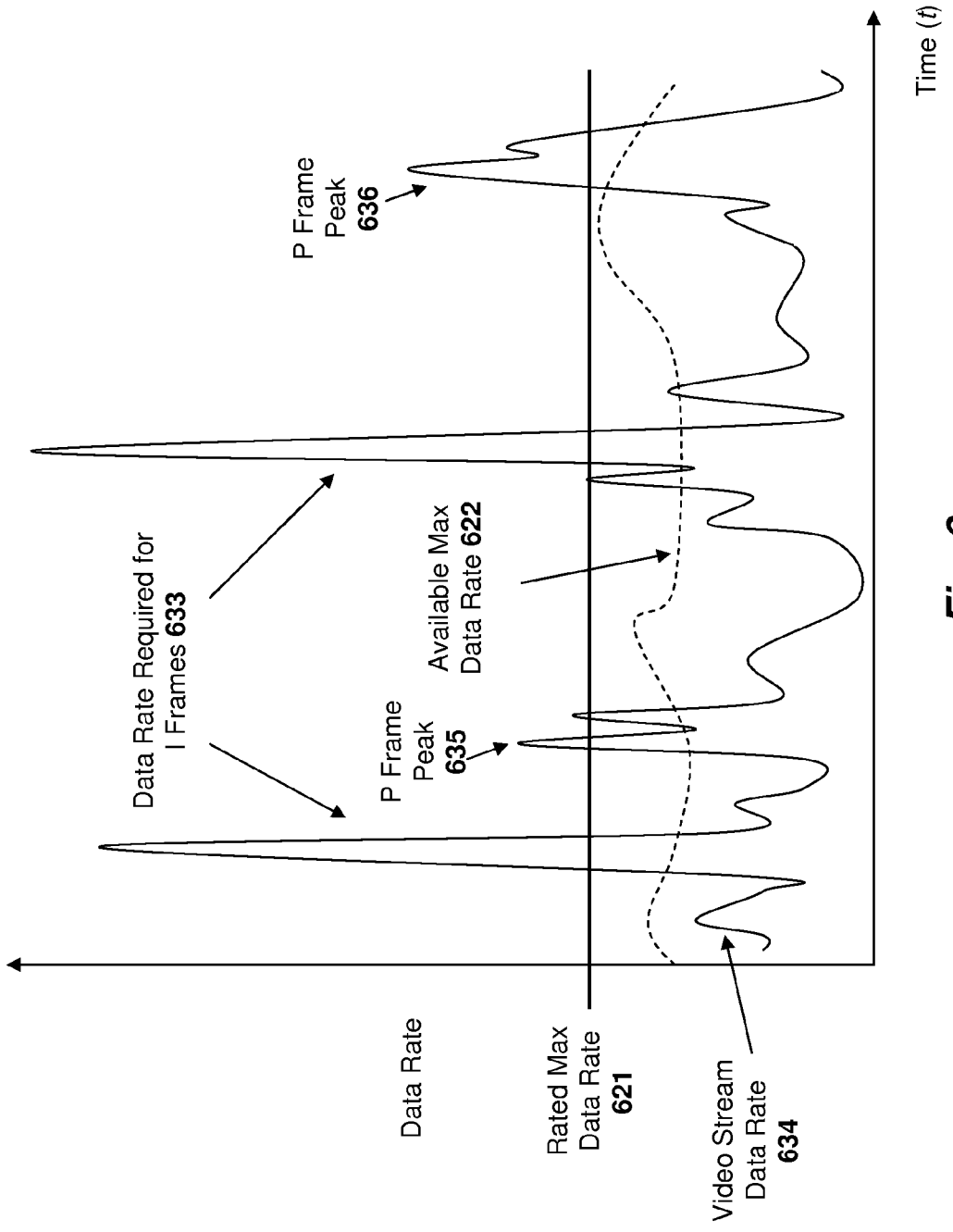

Time (t)
P Frame Peak 636
Data Rate Required for I Frames 633
Available Max Data Rate 622
P Frame Peak 635
Data Rate
Rated Max Data Rate 621
Video Stream Data Rate 634

Fig. 6c

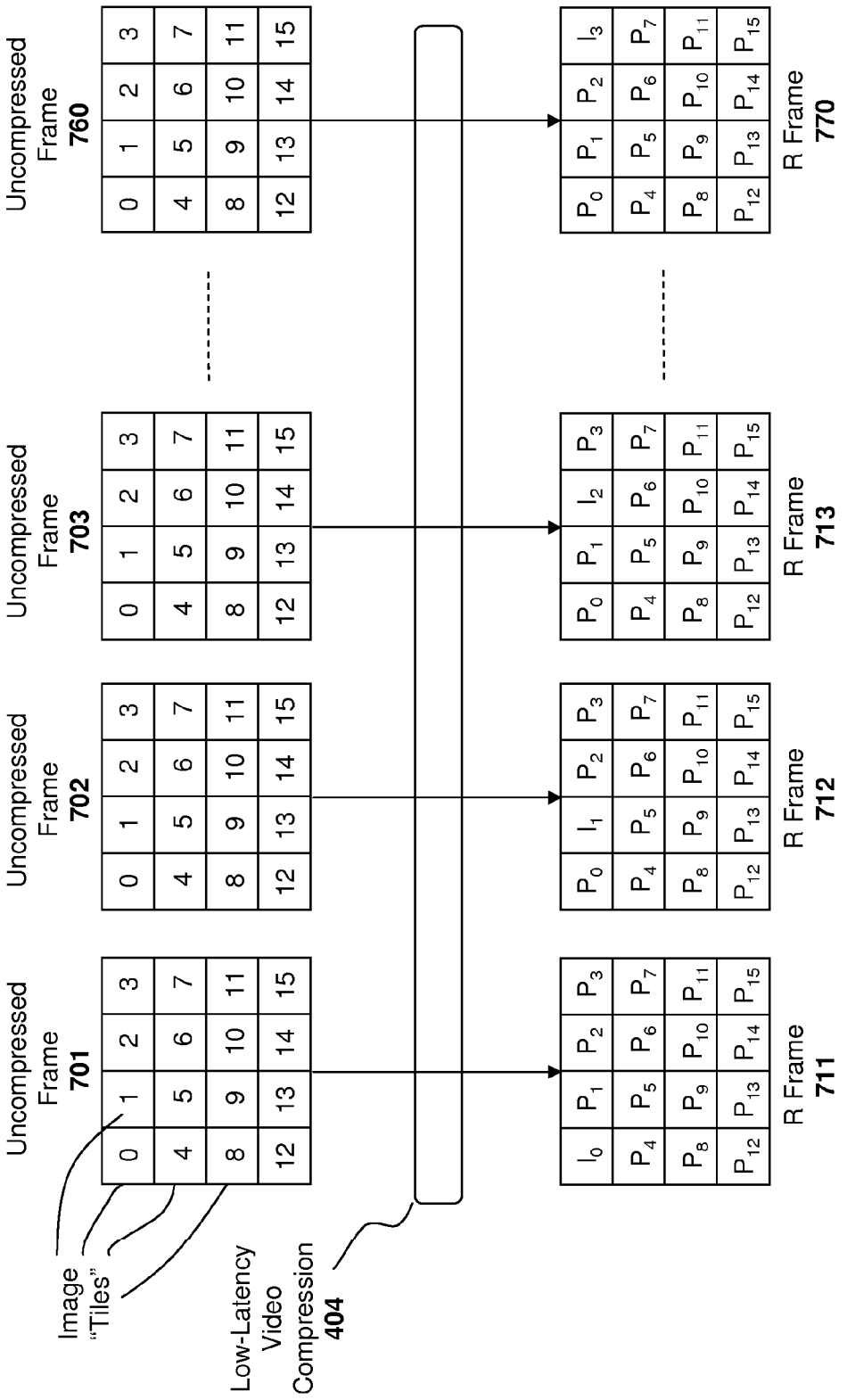

Uncompressed Frame 701
Uncompressed Frame 702
Uncompressed Frame 703
Uncompressed Frame 760
Image "Tiles"
Low-Latency Video Compression 404
R Frame 711
R Frame 712
R Frame 713
R Frame 770

Fig. 7a

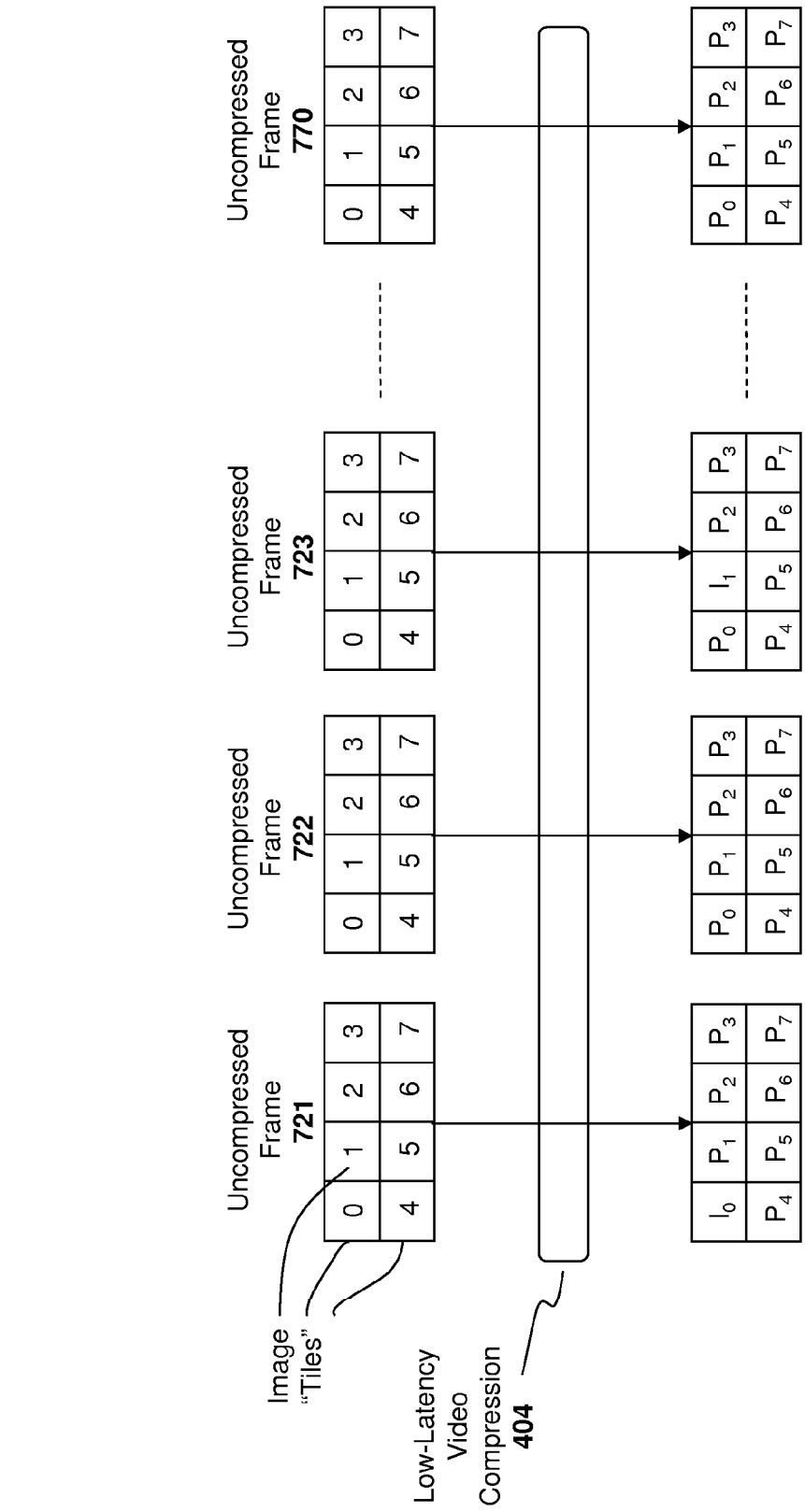

Uncompressed Frame 721
Uncompressed Frame 722
Uncompressed Frame 723
Uncompressed Frame 770
Image "Tiles"
Low-Latency Video Compression 404
R Frame 731
R Frame 732
R Frame 733
R Frame 780

Fig. 7b

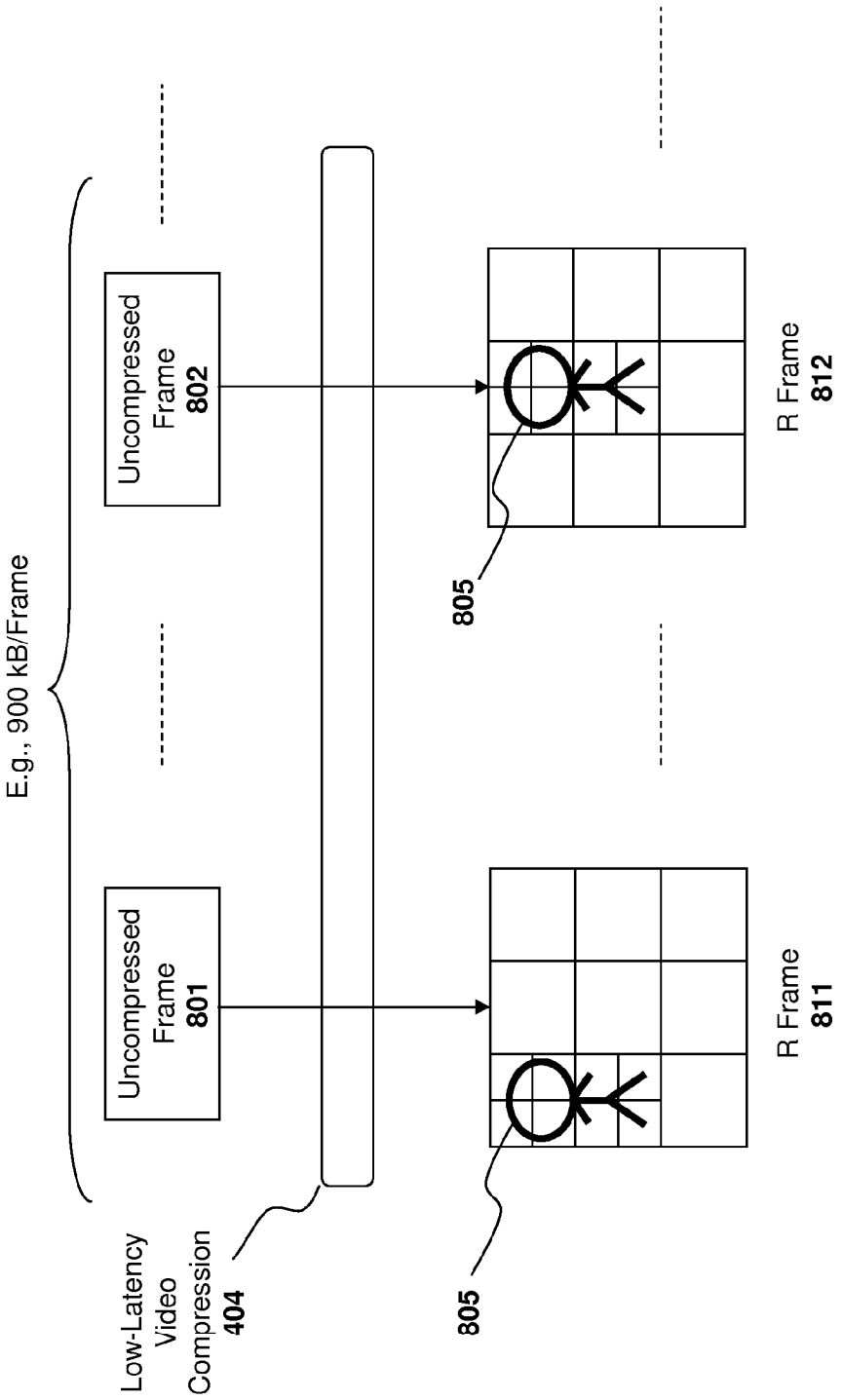

E.g., 900 kB/Frame
Uncompressed Frame 802
Uncompressed Frame 801
Low-Latency Video Compression 404
805
805
R Frame 812
R Frame 811

Fig. 8

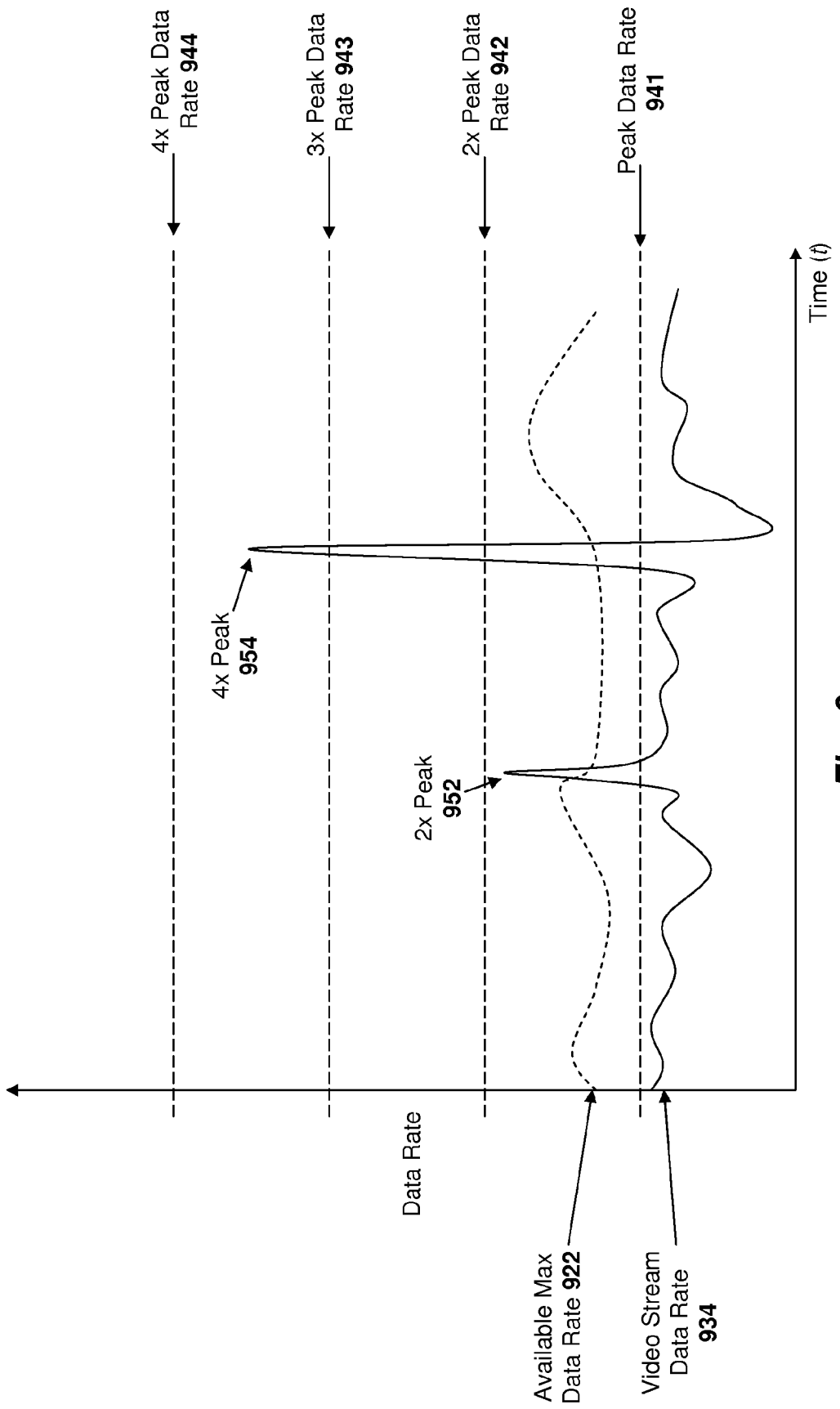

4x Peak Data Rate 944
3x Peak Data Rate 943
2x Peak Data Rate 942
Peak Data Rate 941
4x Peak 954
2x Peak 952
Time (t)
Data Rate
Available Max Data Rate 922
Video Stream Data Rate 934

Fig. 9a

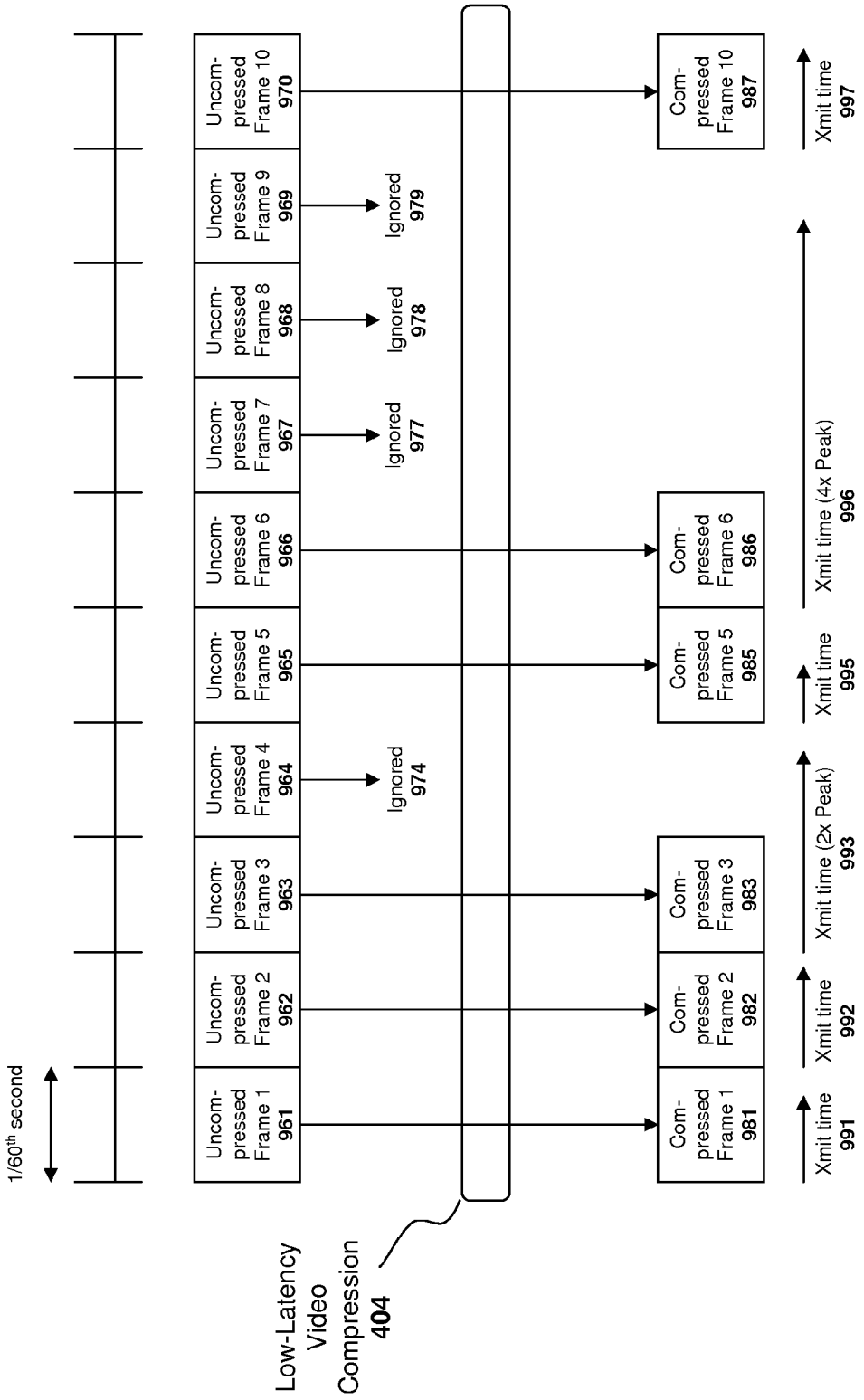

1/60th second
Uncom-pressed Frame 1 961
Uncom-pressed Frame 2 962
Uncom-pressed Frame 3 963
Uncom-pressed Frame 4 964
Uncom-pressed Frame 5 965
Uncom-pressed Frame 6 966
Uncom-pressed Frame 7 967
Uncom-pressed Frame 8 968
Uncom-pressed Frame 9 969
Uncom-pressed Frame 10 970
Ignored 974
Ignored 977
Ignored 978
Ignored 979
Low-Latency Video Compression 404
Com-pressed Frame 1 981
Com-pressed Frame 2 982
Com-pressed Frame 3 983
Com-pressed Frame 5 985
Com-pressed Frame 6 986
Com-pressed Frame 10 987
Xmit time 991
Xmit time 992
Xmit time (2x Peak) 993
Xmit time 995
Xmit time (4x Peak) 996
Xmit time 997

Fig. 9b

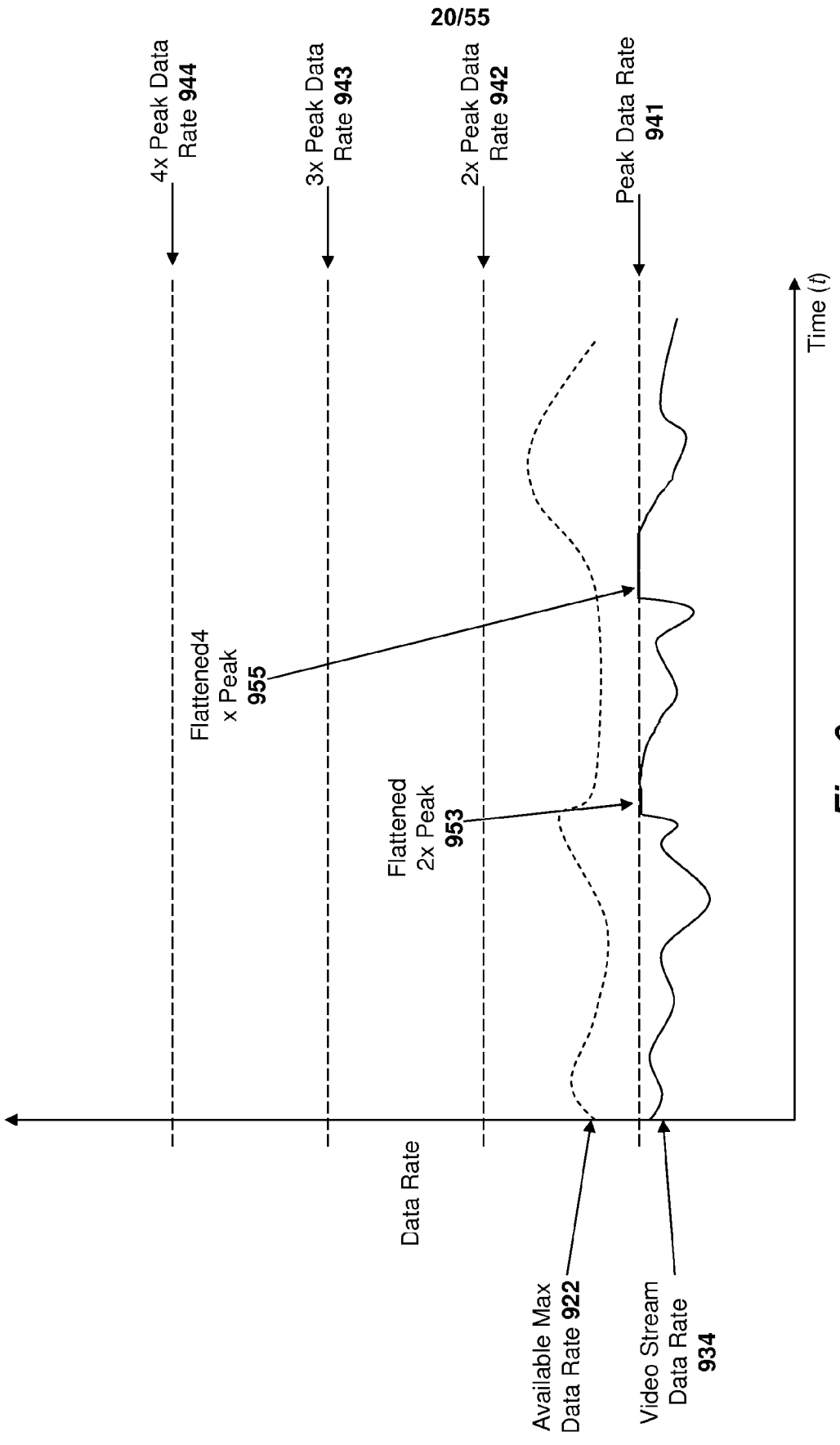

4x Peak Data Rate 944
3x Peak Data Rate 943
2x Peak Data Rate 942
Peak Data Rate 941
Flattened4 x Peak 955
Flattened 2x Peak 953
Data Rate
Time (t)
Available Max Data Rate 922
Video Stream Data Rate 934

Fig. 9c

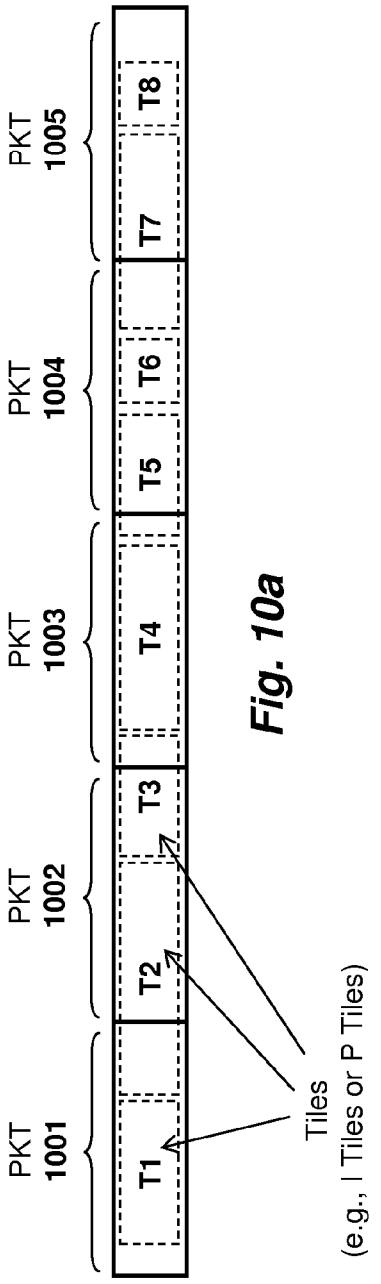

PKT 1001
PKT 1002
PKT 1003
PKT 1004
PKT 1005
T1
T2
T3
T4
T5
T6
T7
T8
Tiles
(e.g., I Tiles or P Tiles)

Fig. 10a

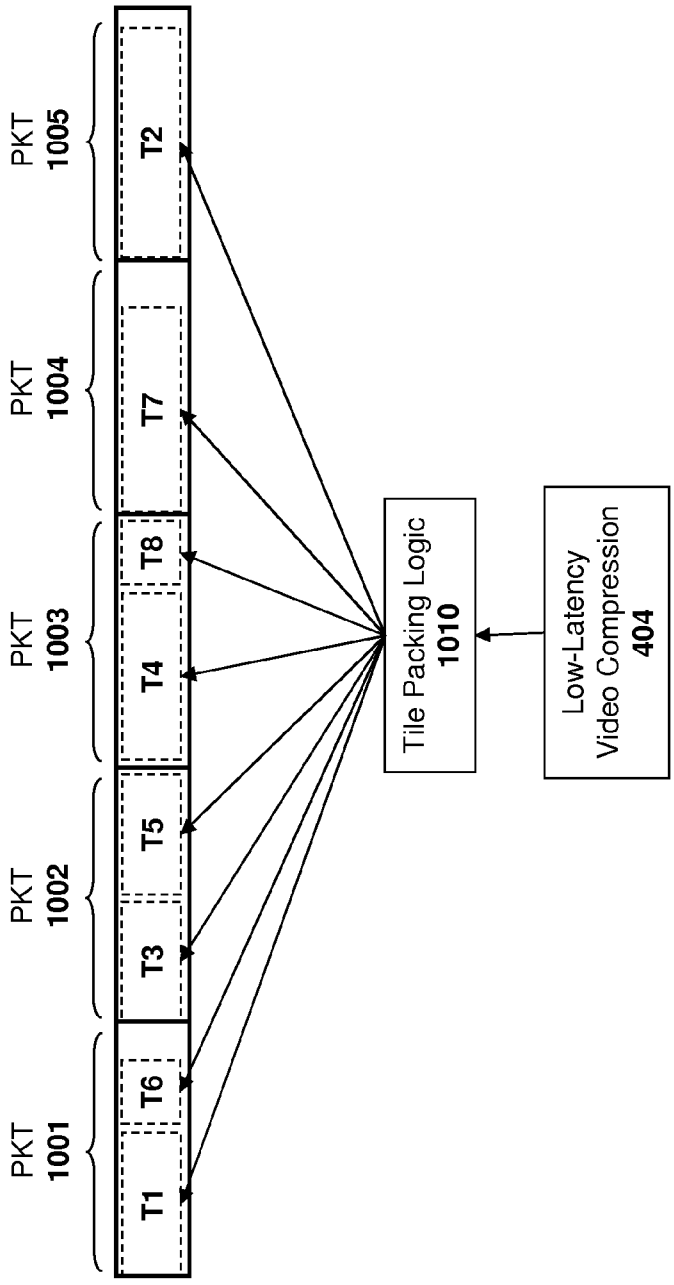

PKT 1001
T1
T6
PKT 1002
T3
T5
PKT 1003
T4
T8
PKT 1004
T7
PKT 1005
T2
Tile Packing Logic 1010
Low-Latency Video Compression 404

Fig. 10b

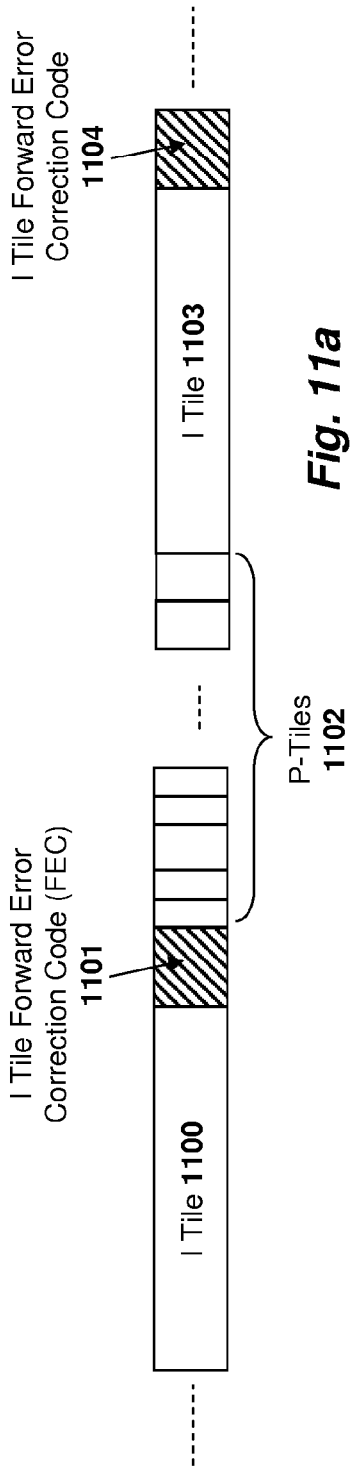


Fig. 11a

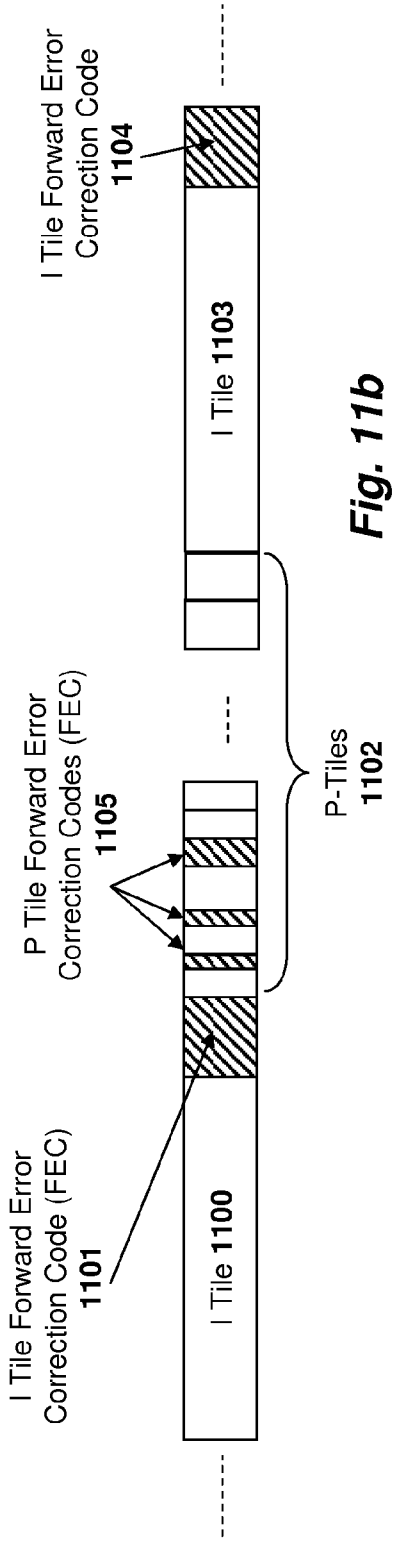


Fig. 11b

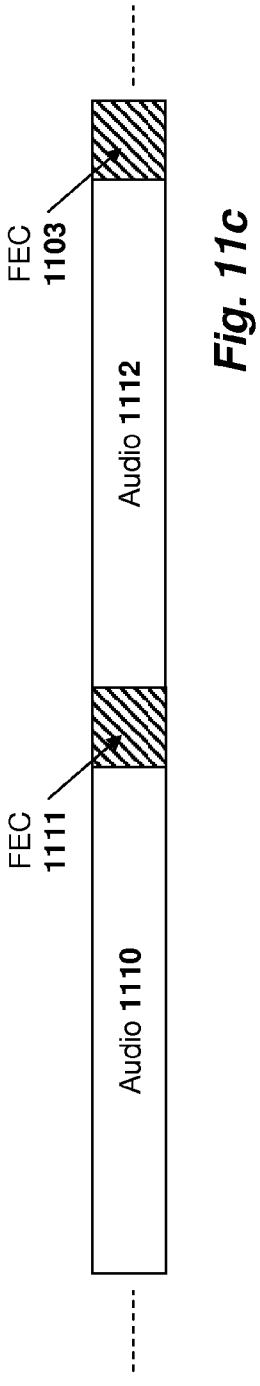


Fig. 11c

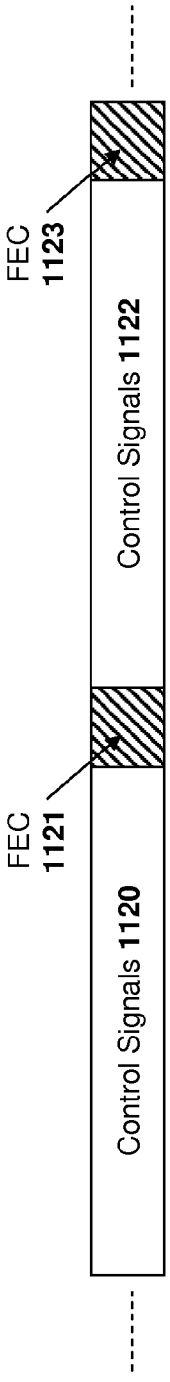

FEC
1123
Control Signals 1122
FEC
1121
Control Signals 1120

Fig. 11d

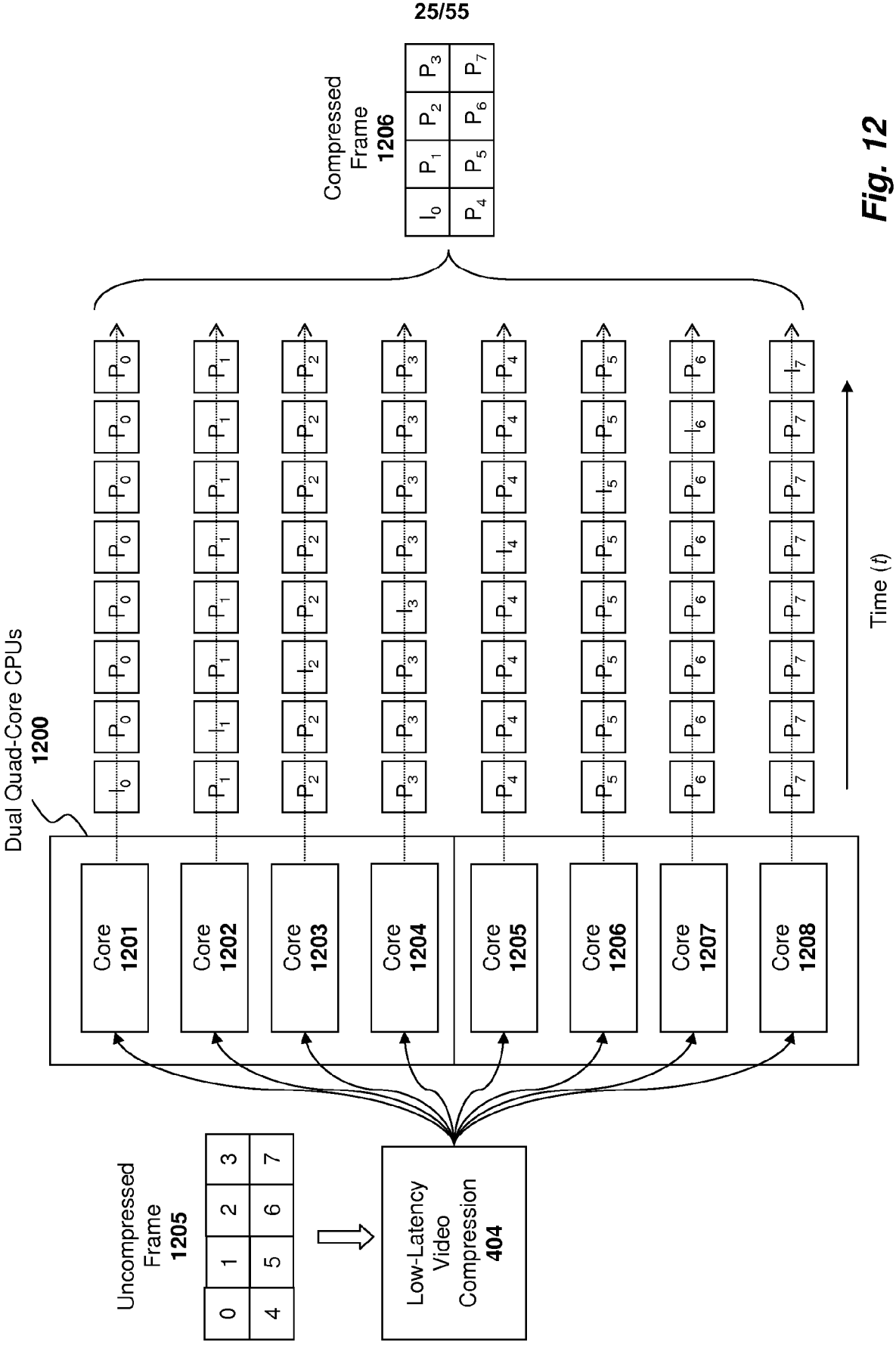

Compressed Frame 1206
I0 P1 P2 P3
P4 P5 P6 P7
Dual Quad-Core CPUs 1200
Core 1201
Core 1202
Core 1203
Core 1204
Core 1205
Core 1206
Core 1207
Core 1208
Time (t)
Uncompressed Frame 1205
0 1 2 3
4 5 6 7
Low-Latency Video Compression 404

Fig. 12

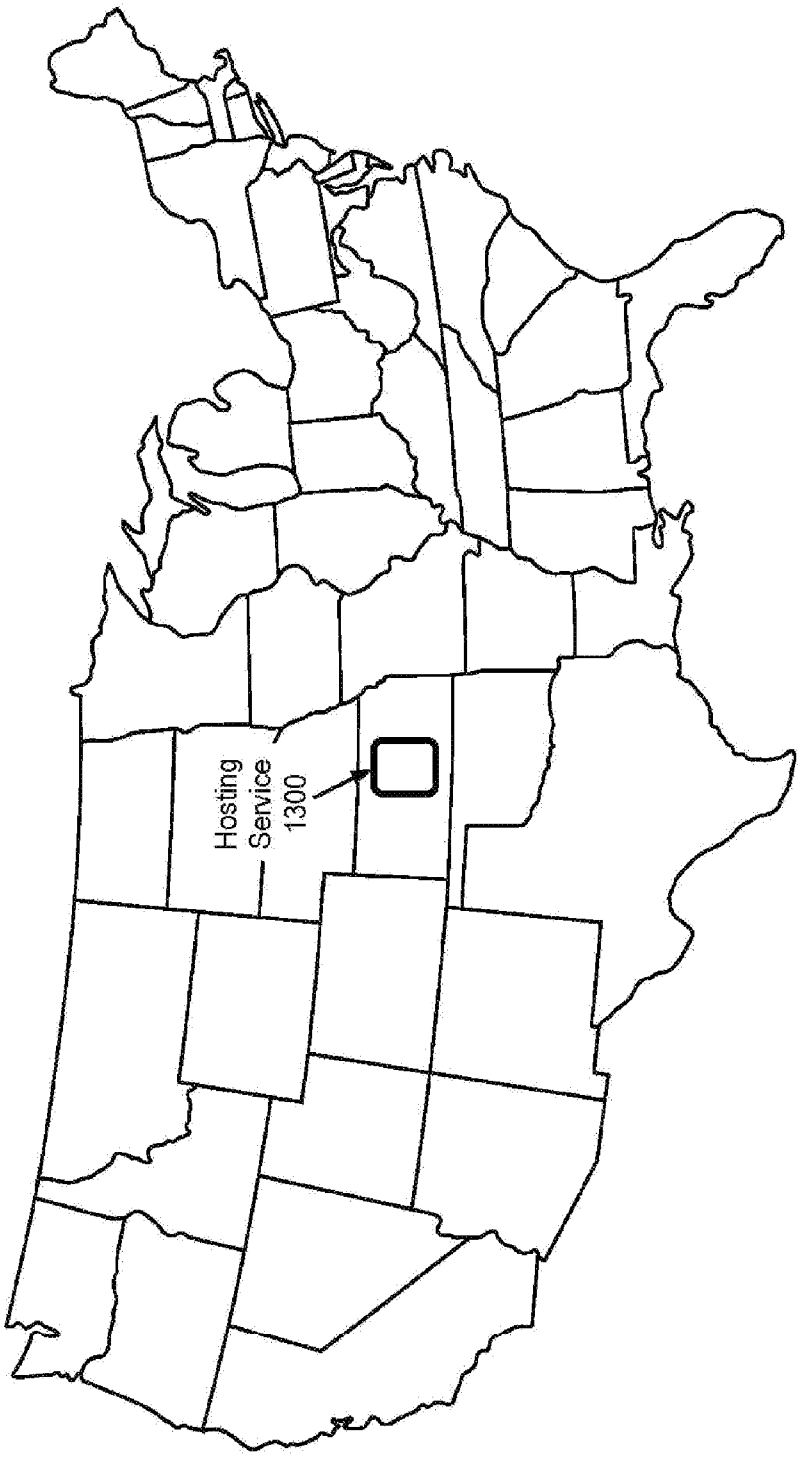


FIG. 13a

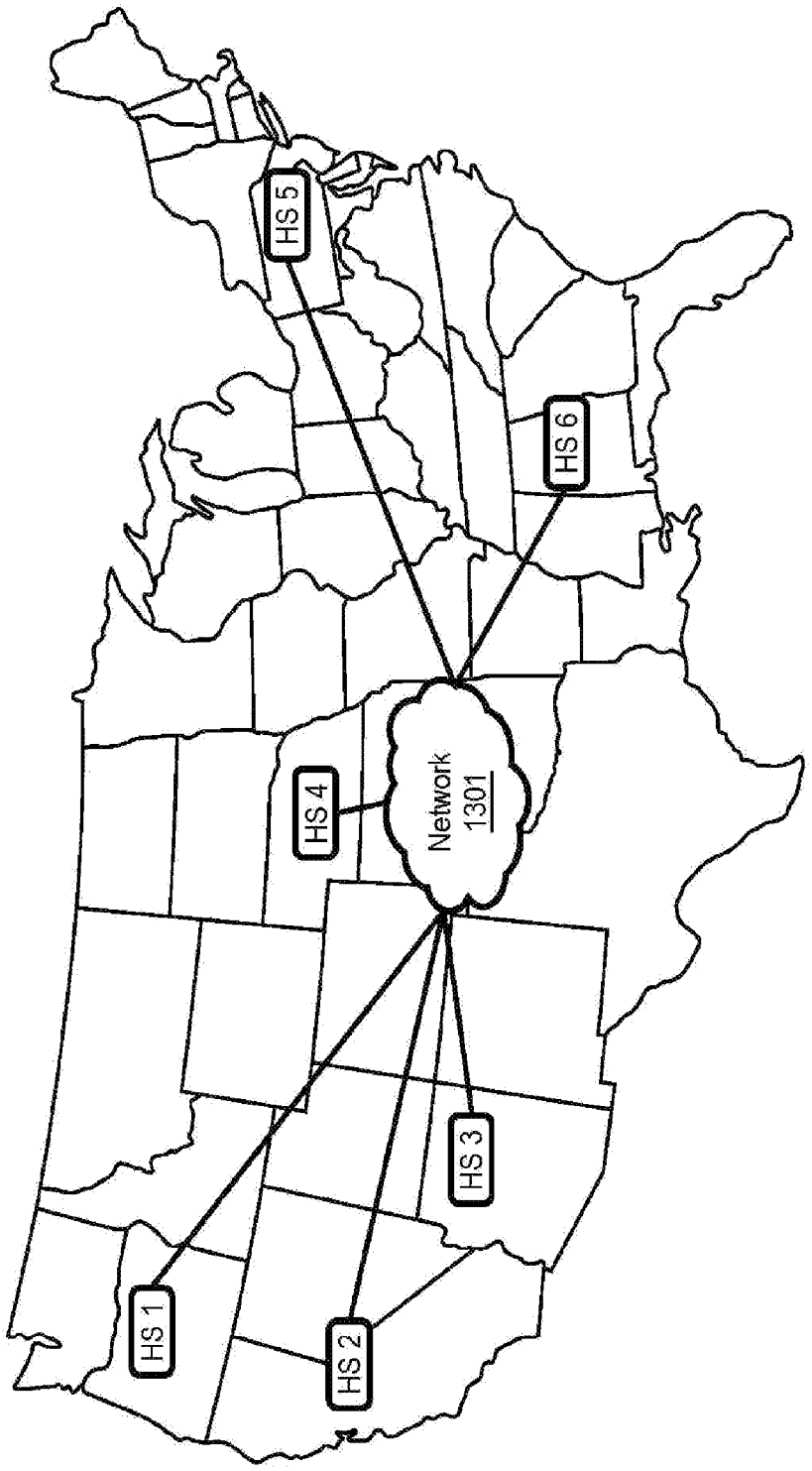


FIG. 13b

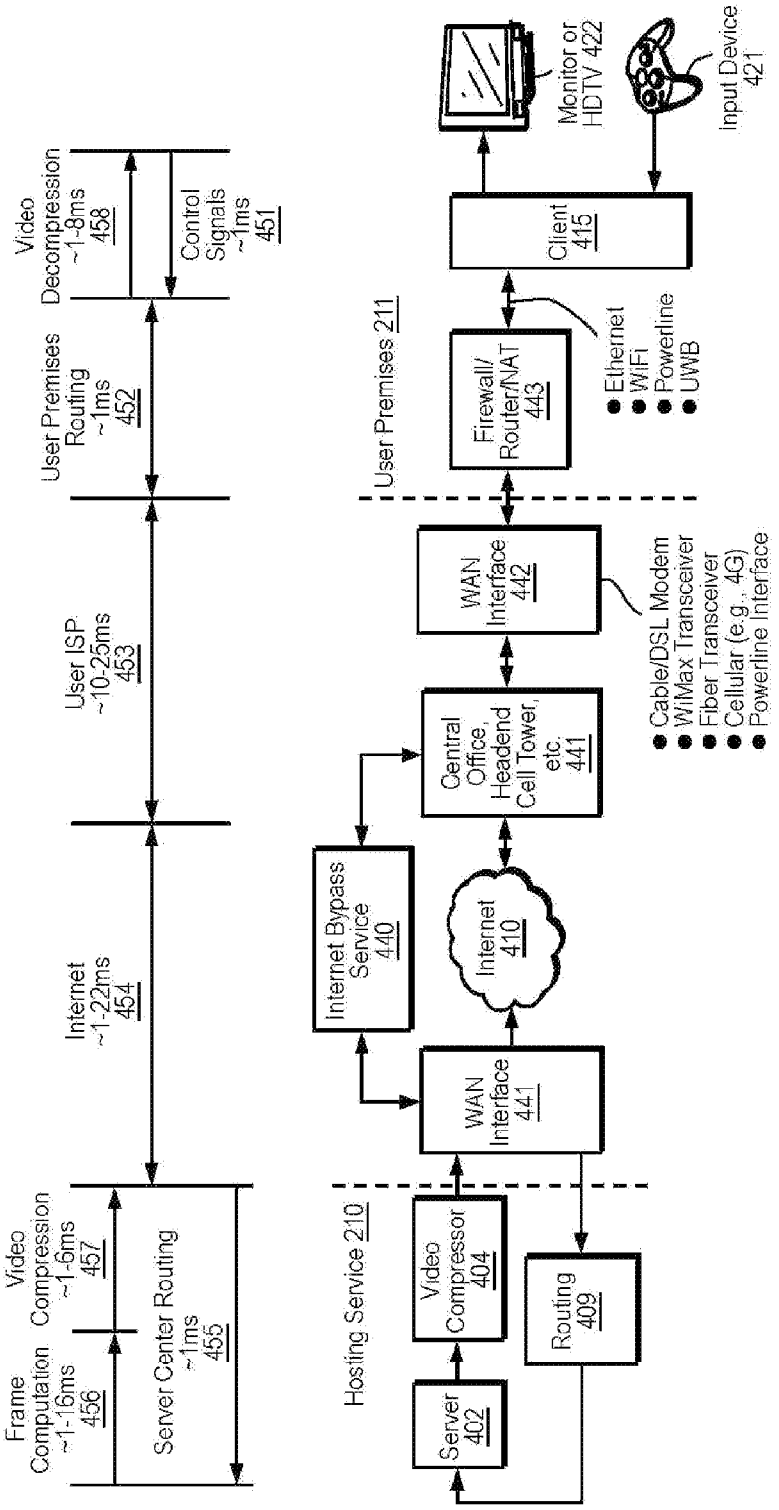

Frame Computation ~1-16ms 456
Video Compression ~1-6ms 457
Server Center Routing ~1ms 455
Internet ~1-22ms 454
User ISP ~10-25ms 453
User Premises Routing ~1ms 452
Video Decompression ~1-8ms 458
Control Signals ~1ms 451
Hosting Service 210
Server 402
Video Compressor 404
Routing 409
WAN Interface 441
Internet Bypass Service 440
Internet 410
Central Office, Headend Cell Tower, etc. 441
WAN Interface 442
Cable/DSL Modem
WiMax Transceiver
Fiber Transceiver
Cellular (e.g., 4G)
Powerline Interface
User Premises 211
Firewall/ Router/NAT 443
Ethernet
WiFi
Powerline
UWB
Client 415
Monitor or HDTV 422
Input Device 421

FIG. 14

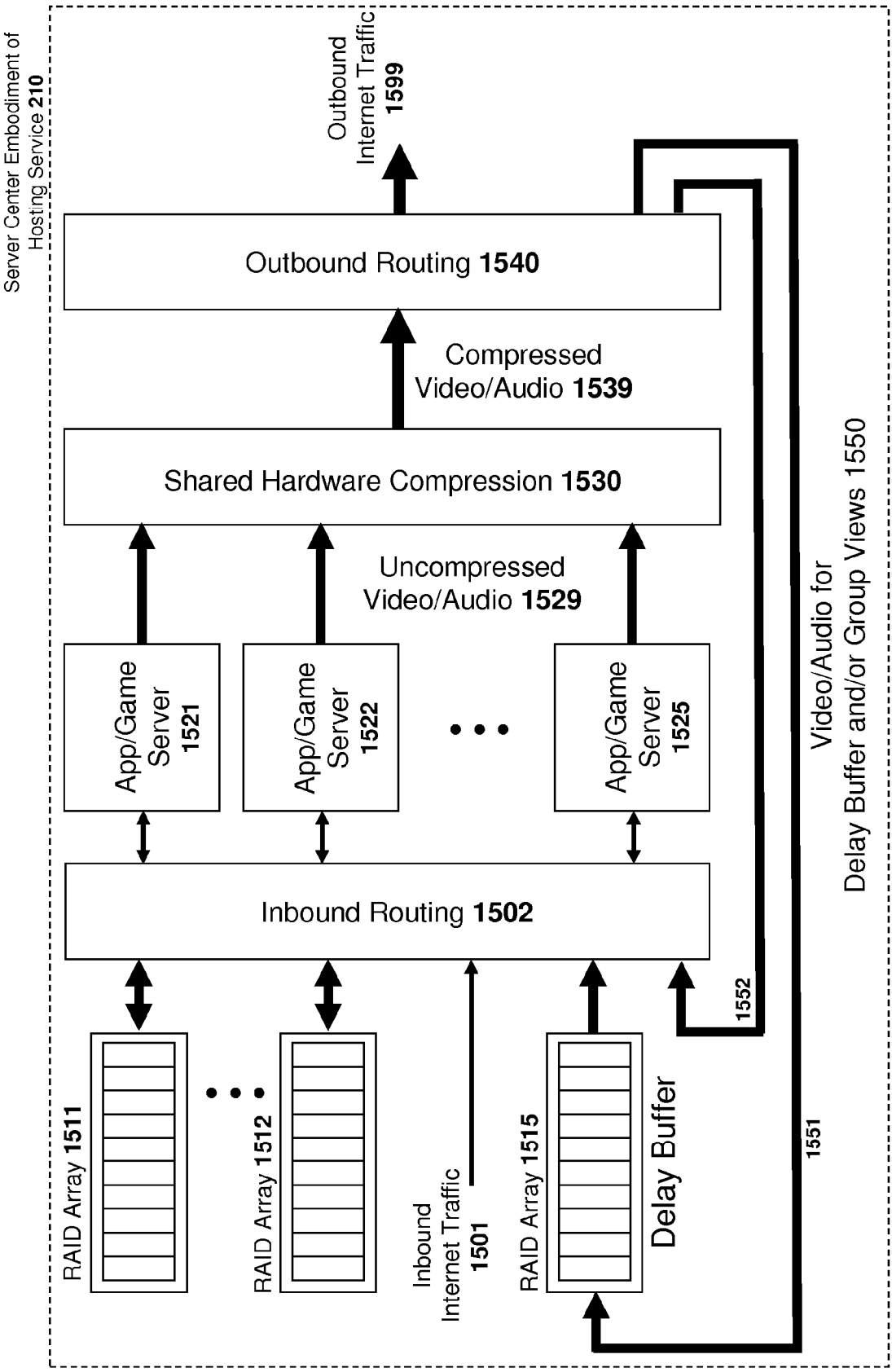

Server Center Embodiment of Hosting Service 210
Outbound Internet Traffic 1599
Outbound Routing 1540
Compressed Video/Audio 1539
Shared Hardware Compression 1530
Uncompressed Video/Audio 1529
App/Game Server 1521
App/Game Server 1522
App/Game Server 1525
Inbound Routing 1502
RAID Array 1511
RAID Array 1512
Inbound Internet Traffic 1501
RAID Array 1515
Delay Buffer
1552
1551
Video/Audio for Delay Buffer and/or Group Views 1550

Fig. 15

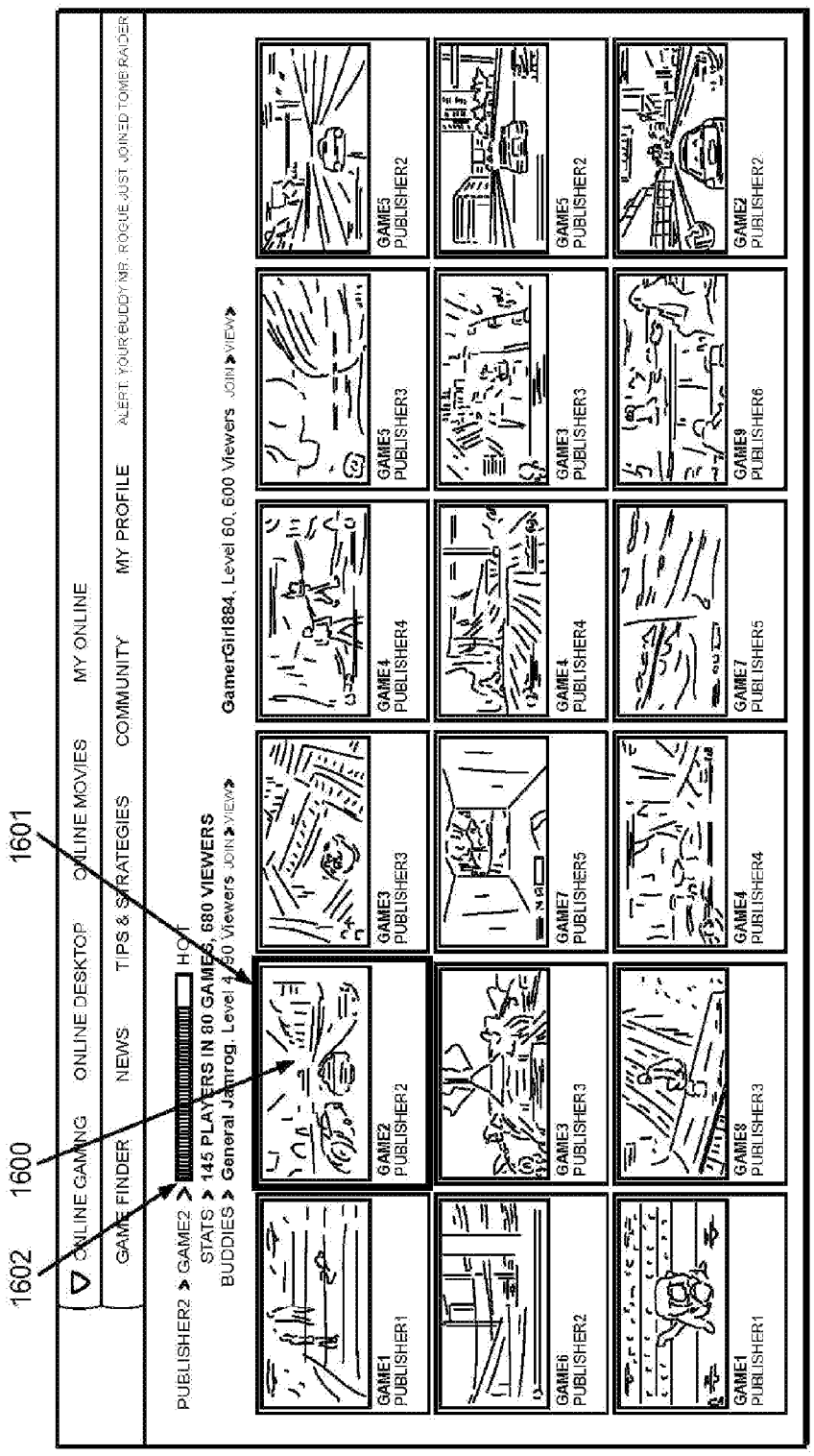

1602
1600
1601
ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
ALERT: YOUR BUDDY MR. ROGUE JUST JOINED TOMB RAIDER
PUBLISHER2 > GAME2 > HOT
STATS > 145 PLAYERS IN 80 GAMES, 680 VIEWERS
BUDDIES > General Jamrog, Level 4, 50 Viewers JOIN > VIEW >
GamerGirl884, Level 60, 600 Viewers JOIN > VIEW >
GAME1 PUBLISHER1
GAME2 PUBLISHER2
GAME3 PUBLISHER3
GAME4 PUBLISHER4
GAME5 PUBLISHER2
GAME6 PUBLISHER2
GAME3 PUBLISHER3
GAME7 PUBLISHER5
GAME4 PUBLISHER4
GAME3 PUBLISHER3
GAME5 PUBLISHER2
GAME1 PUBLISHER1
GAME8 PUBLISHER3
GAME4 PUBLISHER4
GAME7 PUBLISHER5
GAME9 PUBLISHER6
GAME2 PUBLISHER2

FIG. 16

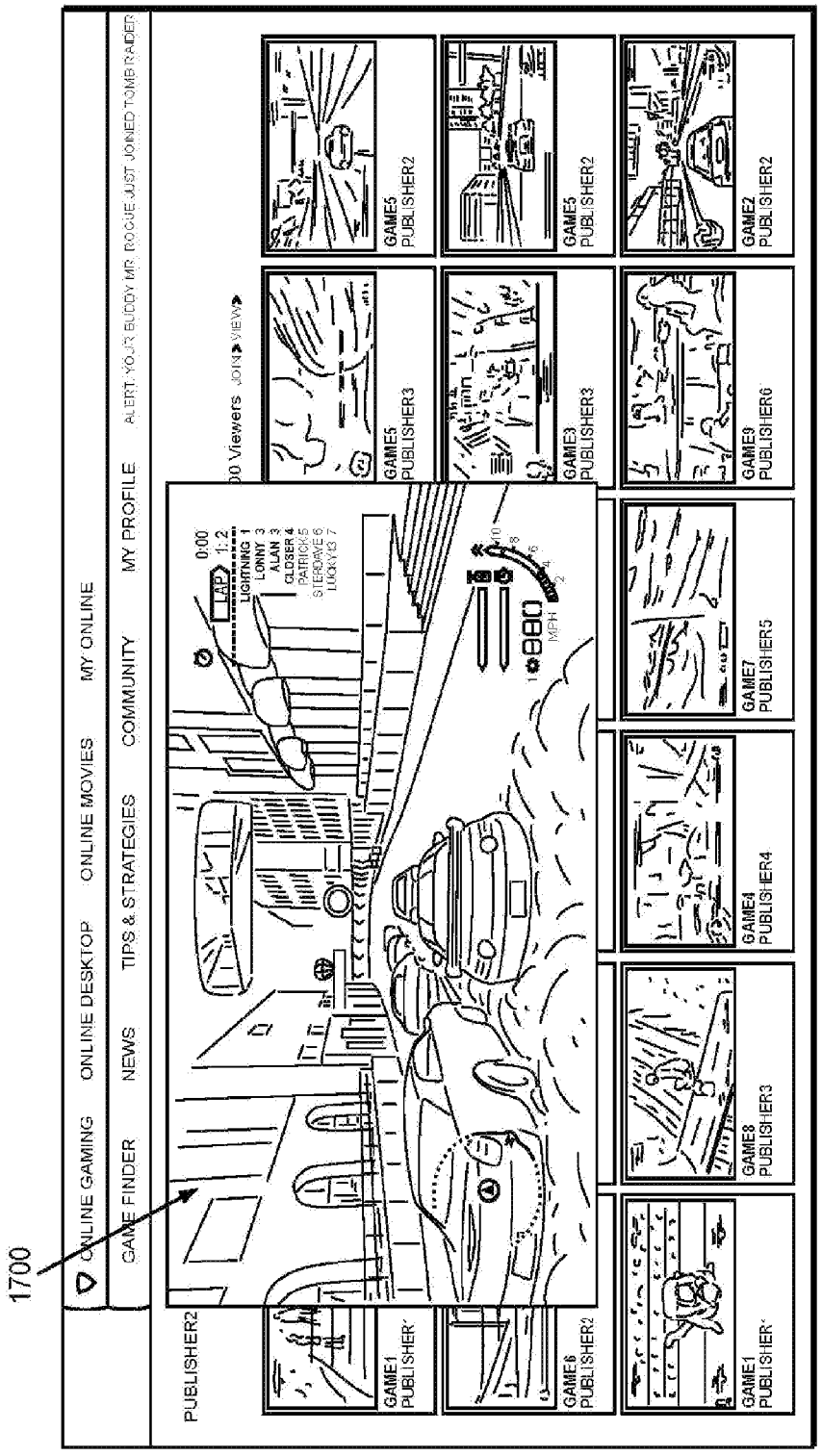

1700
ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
ALERT: YOUR BUDDY MR. ROGUE JUST JOINED TOMB RAIDER
PUBLISHER2
0 Viewers JOIN VIEW
LAP 1/2
0:00
LIGHTNING 1
LONNY 3
ALAN 3
GLOSER 4
PATRICK 5
STERRAVE 6
LUCKY13 7
MPH
GAME1 PUBLISHER
GAME6 PUBLISHER2
GAME8 PUBLISHER3
GAME4 PUBLISHER4
GAME7 PUBLISHER5
GAME3 PUBLISHER3
GAME5 PUBLISHER3
GAME5 PUBLISHER2
GAME9 PUBLISHER6
GAME2 PUBLISHER2

FIG. 17

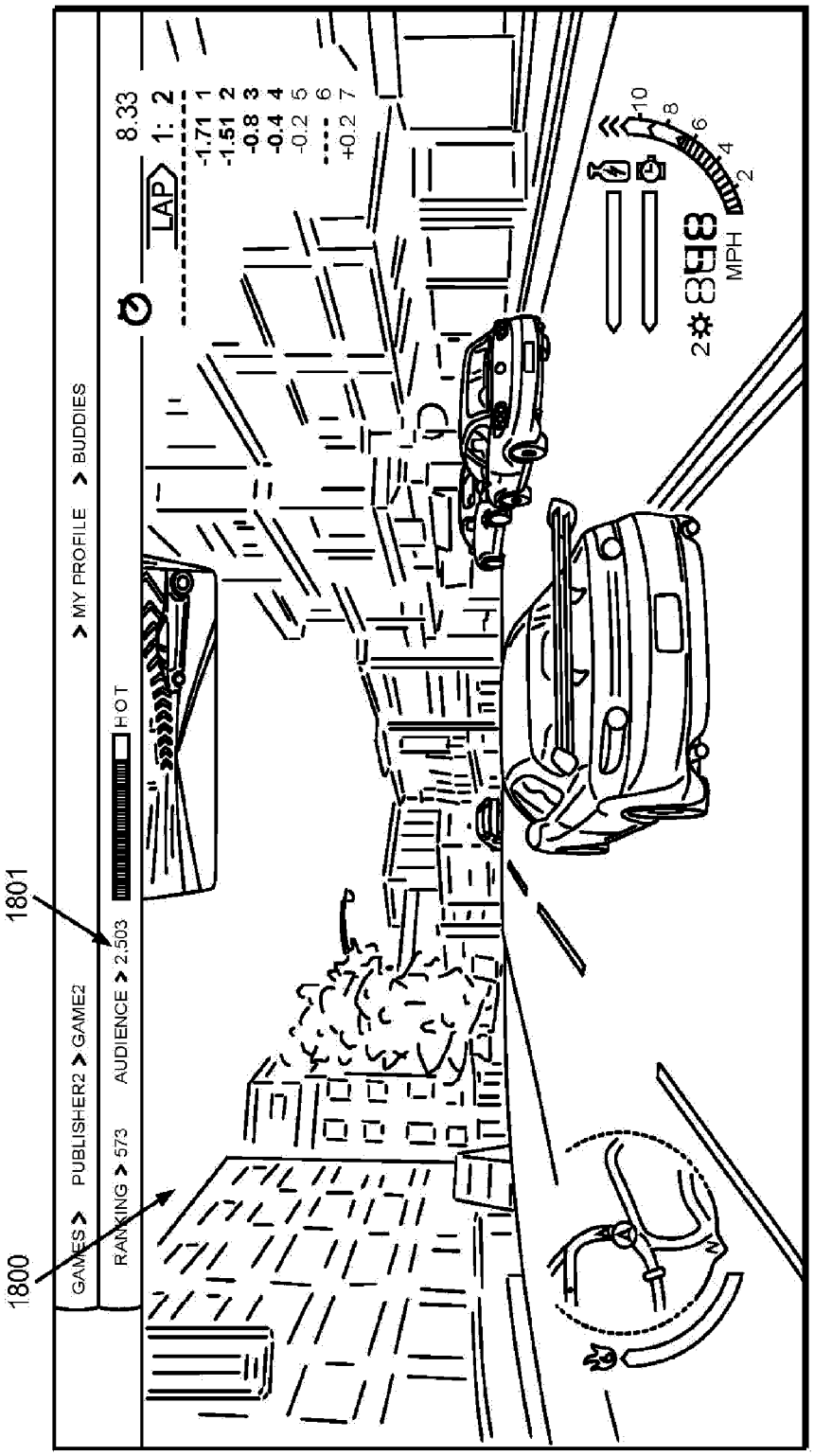

1801
1800
GAMES > PUBLISHER2 > GAME2
RANKING > 573
AUDIENCE > 2,503
HOT
> MY PROFILE > BUDDIES
8.33
LAP 1: 2
-1.71 1
-1.51 2
-0.8 3
-0.4 4
-0.2 5
---- 6
+0.2 7
MPH

FIG. 18

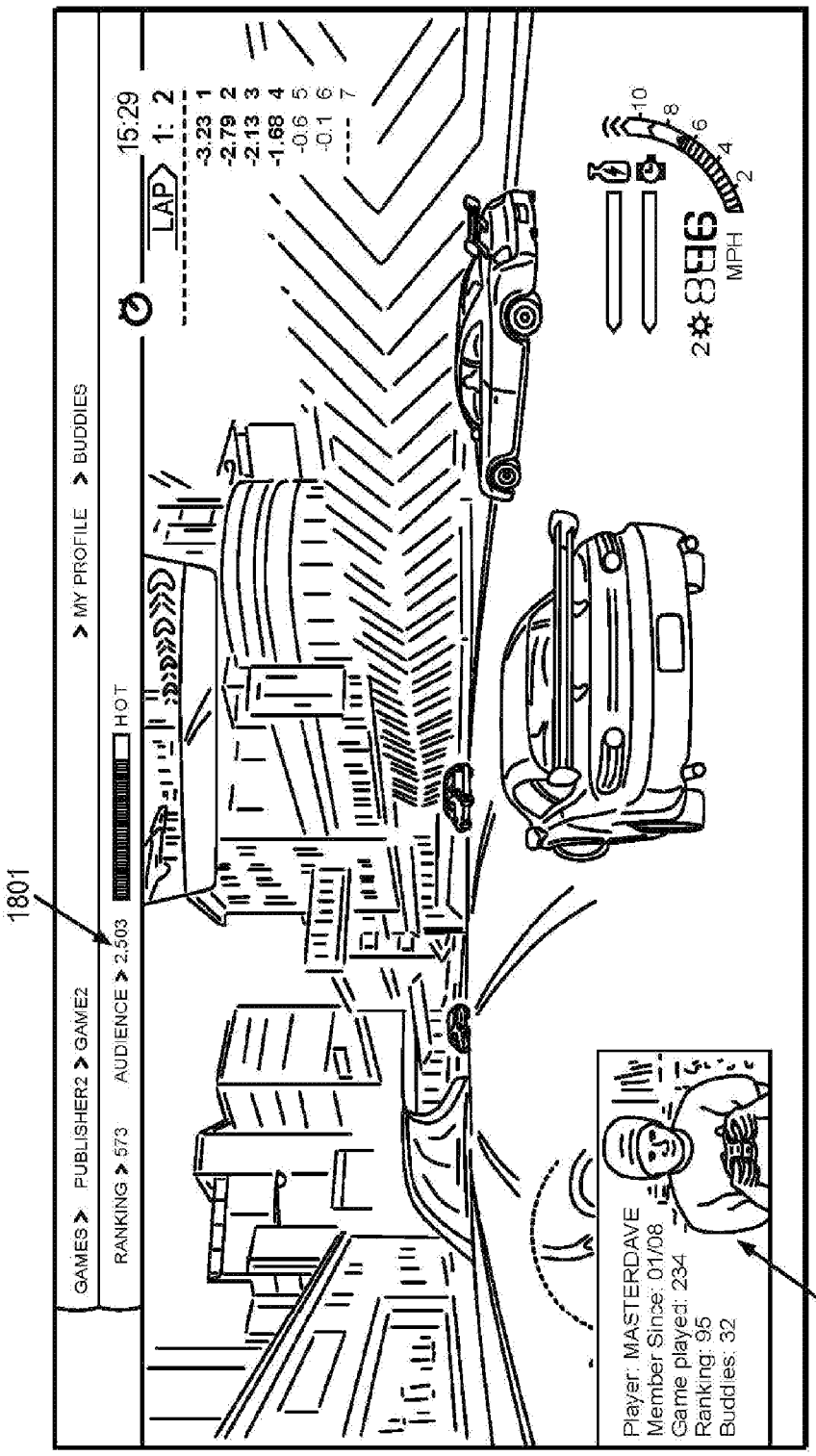

1801
GAMES > PUBLISHER2 > GAME2
> MY PROFILE > BUDDIES
RANKING > 573 AUDIENCE > 2,503 HOT
15:29
LAP 1: 2
-3.23 1
-2.79 2
-2.13 3
-1.68 4
-0.6 5
-0.1 6
7
MPH
Player: MASTERDAVE
Member Since: 01/08
Game played: 234
Ranking: 95
Buddies: 32
1900

FIG. 19

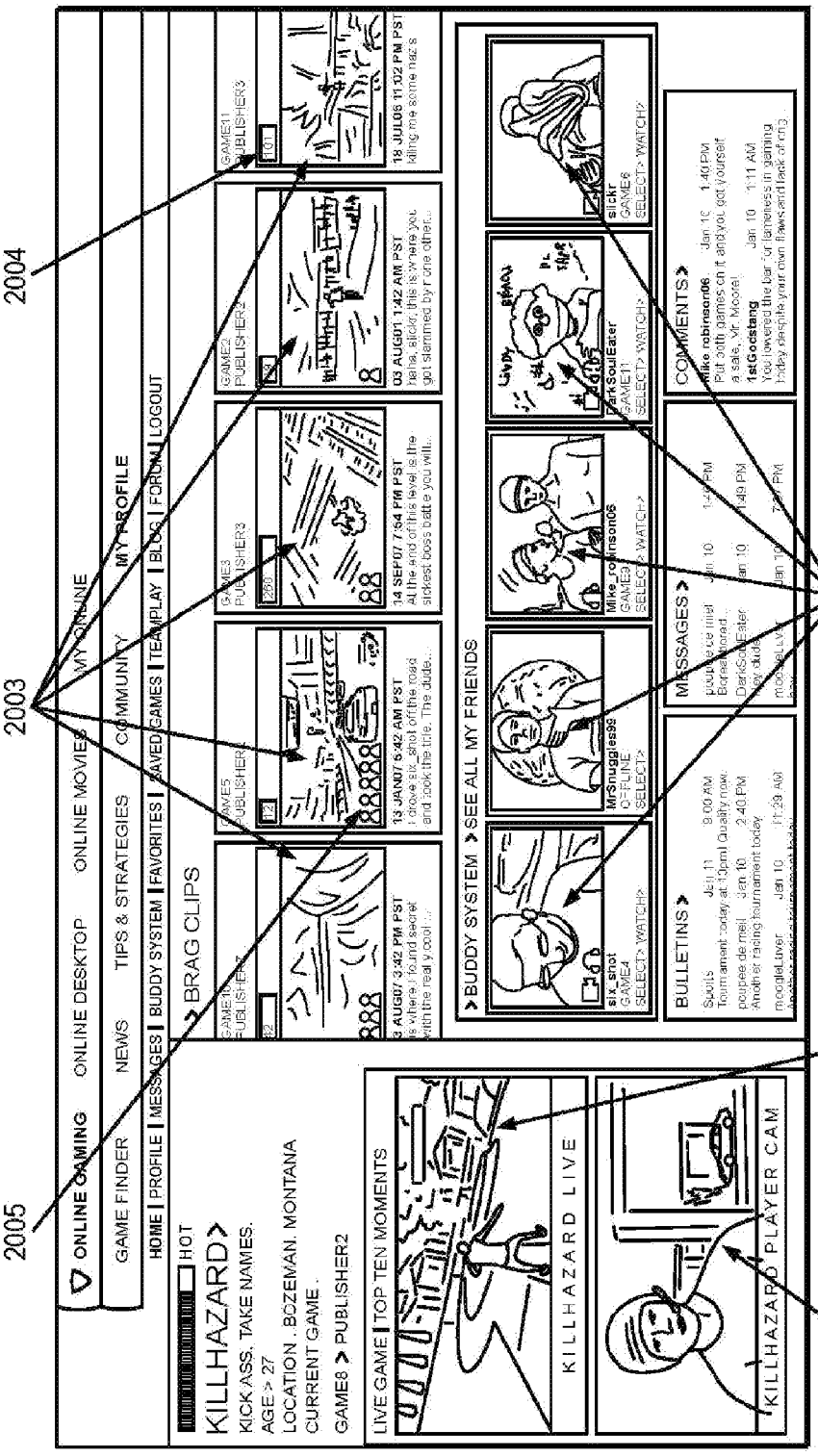

2005
2004
2003
2002
2001
2000
ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
HOME | PROFILE | MESSAGES | BUDDY SYSTEM | FAVORITES | SAVED GAMES | TEAMPLAY | BLOG | FORUM | LOGOUT
HOT
KILLHAZARD>
KICK ASS. TAKE NAMES.
AGE > 27
LOCATION . BOZEMAN. MONTANA
CURRENT GAME .
GAME8 > PUBLISHER2
> BRAG CLIPS
> BUDDY SYSTEM > SEE ALL MY FRIENDS
LIVE GAME | TOP TEN MOMENTS
KILLHAZARD LIVE
KILLHAZARD PLAYER CAM
BULLETINS >
MESSAGES >
COMMENTS >

FIG. 20

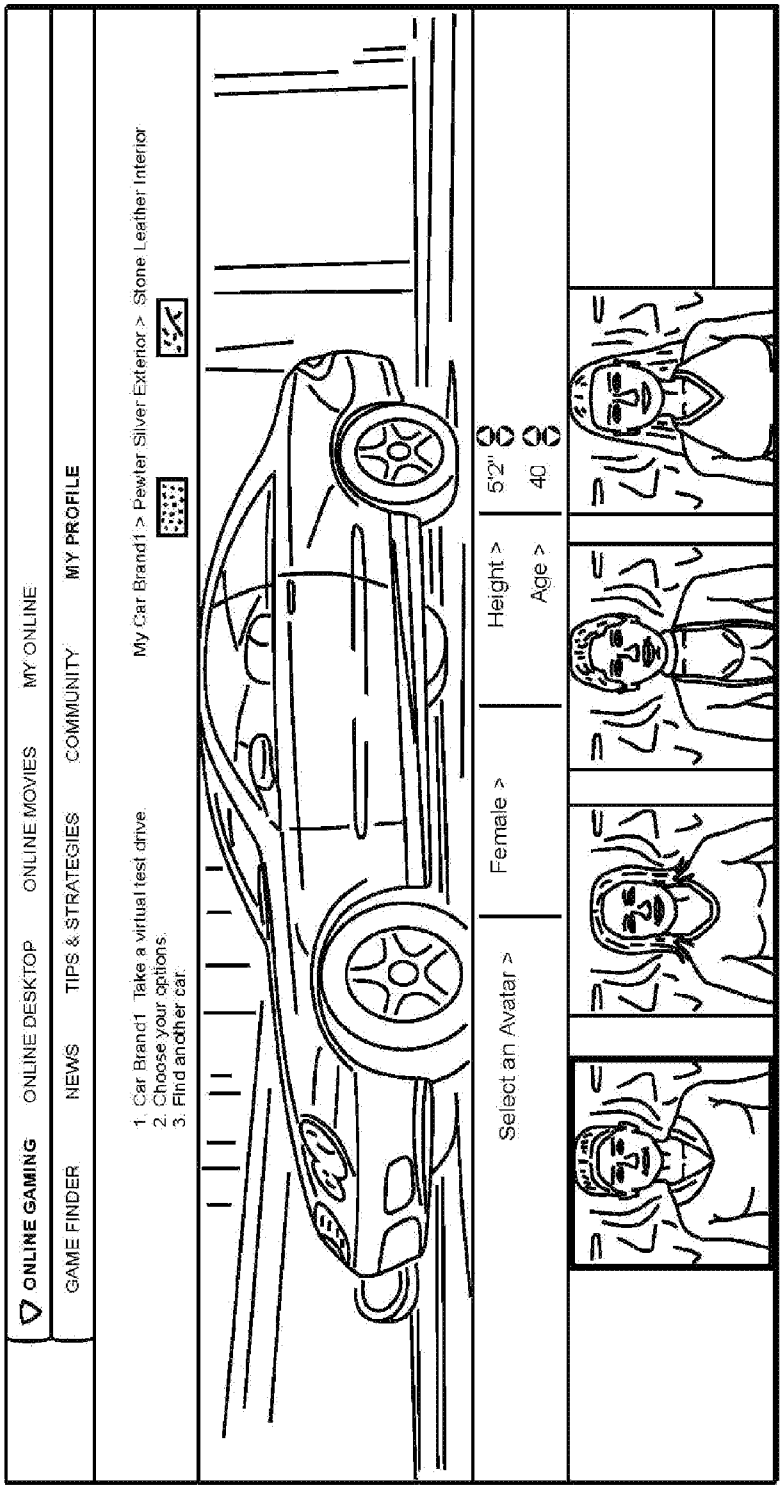

ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
1. Car Brand1 Take a virtual test drive.
2. Choose your options.
3. Find another car.
My Car Brand1 > Pewter Silver Exterior > Stone Leather Interior
Select an Avatar >
Female >
Height >
5'2"
Age >
40

FIG. 21

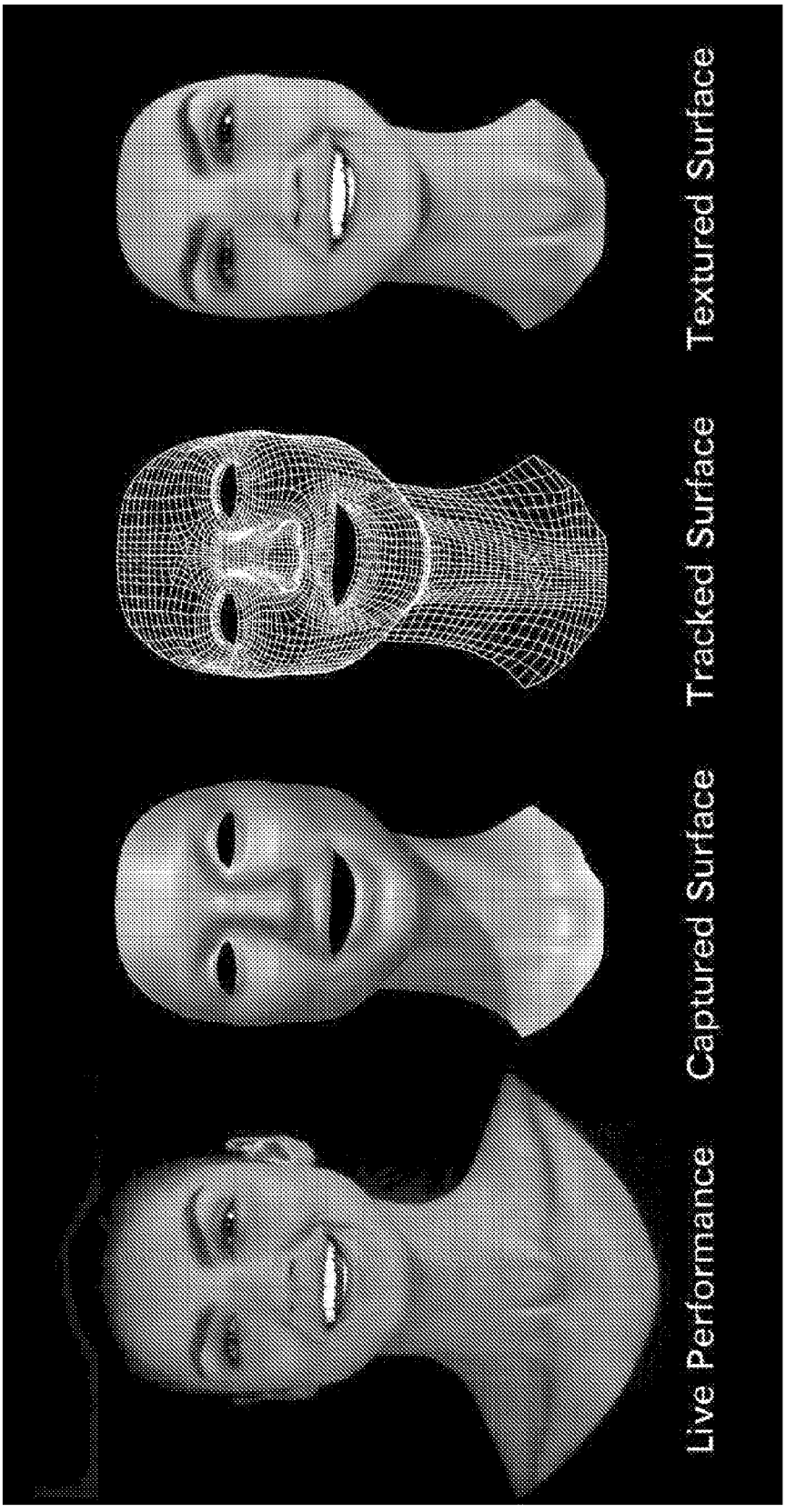

Live Performance
Captured Surface
Tracked Surface
Textured Surface

Fig. 22

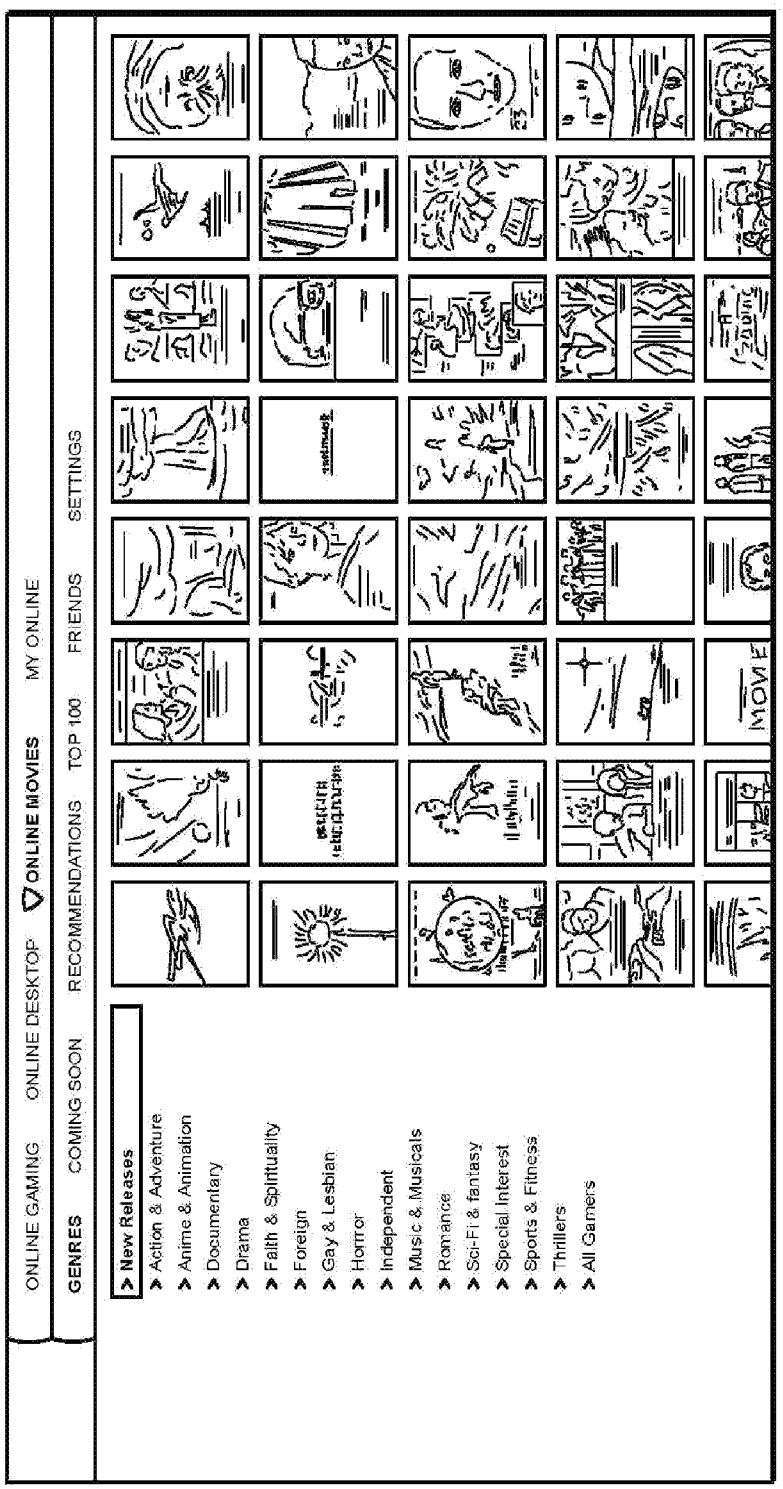

ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GENRES
COMING SOON
RECOMMENDATIONS
TOP 100
FRIENDS
SETTINGS
> New Releases
> Action & Adventure
> Anime & Animation
> Documentary
> Drama
> Faith & Spirituality
> Foreign
> Gay & Lesbian
> Horror
> Independent
> Music & Musicals
> Romance
> Sci-Fi & fantasy
> Special Interest
> Sports & Fitness
> Thrillers
> All Genres

FIG. 23

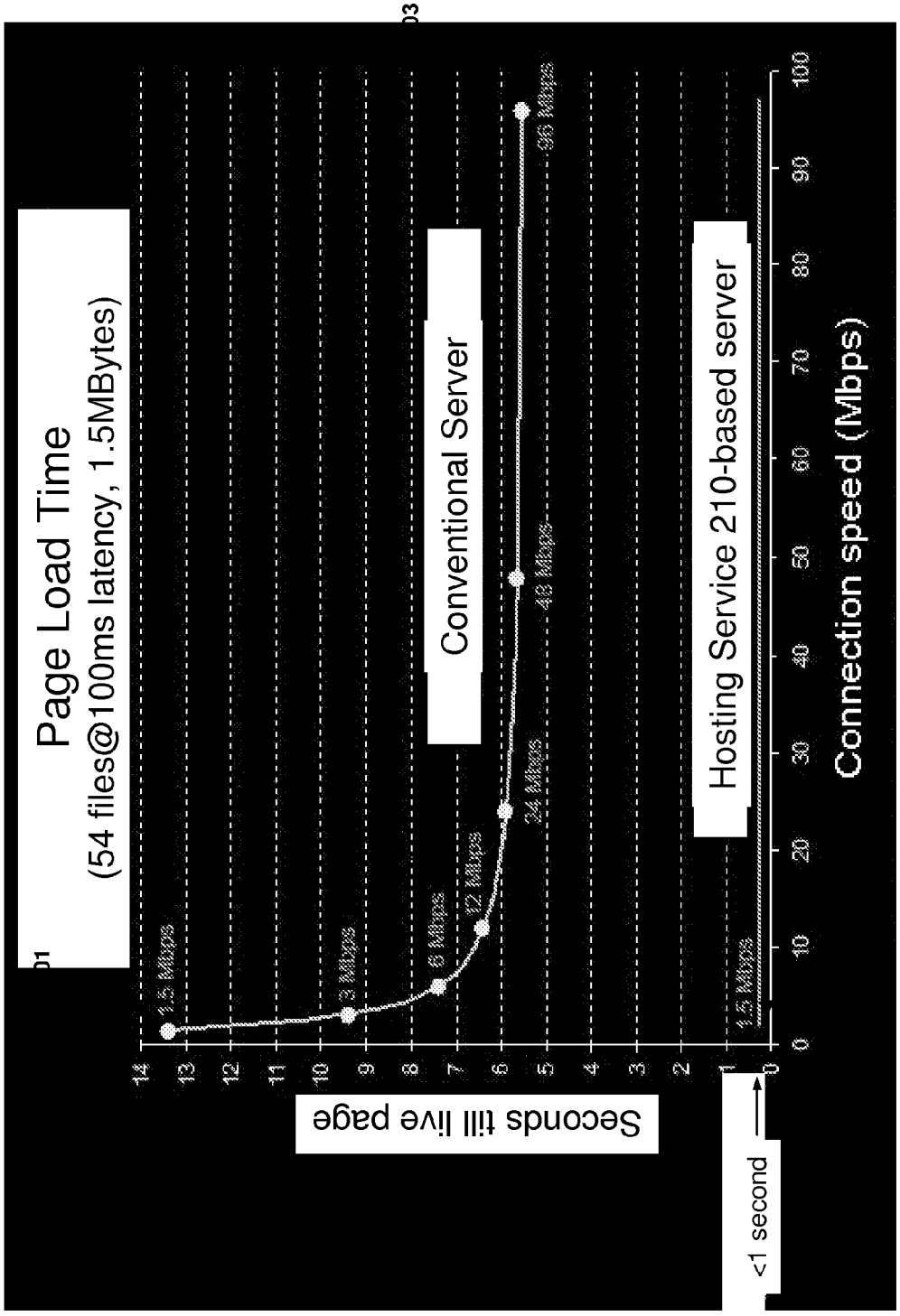

Page Load Time
(54 files@100ms latency, 1.5MBytes)
Conventional Server
Hosting Service 210-based server
Seconds till live page
Connection speed (Mbps)
1.5 Mbps
3 Mbps
6 Mbps
12 Mbps
24 Mbps
48 Mbps
96 Mbps
<1 second

Fig. 24

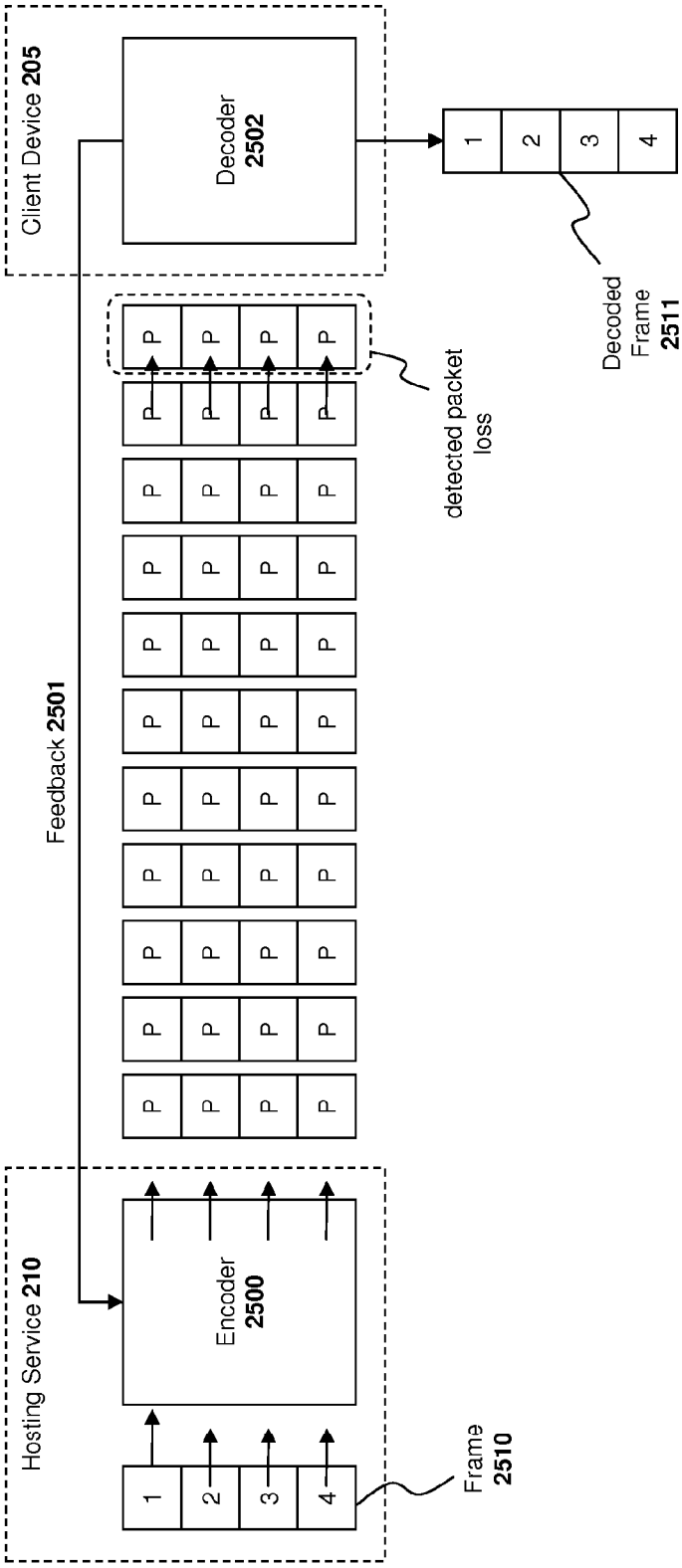

Client Device 205
Decoder 2502
1
2
3
4
Decoded Frame 2511
detected packet loss
P
Feedback 2501
Hosting Service 210
Encoder 2500
1
2
3
4
Frame 2510

Fig. 25a

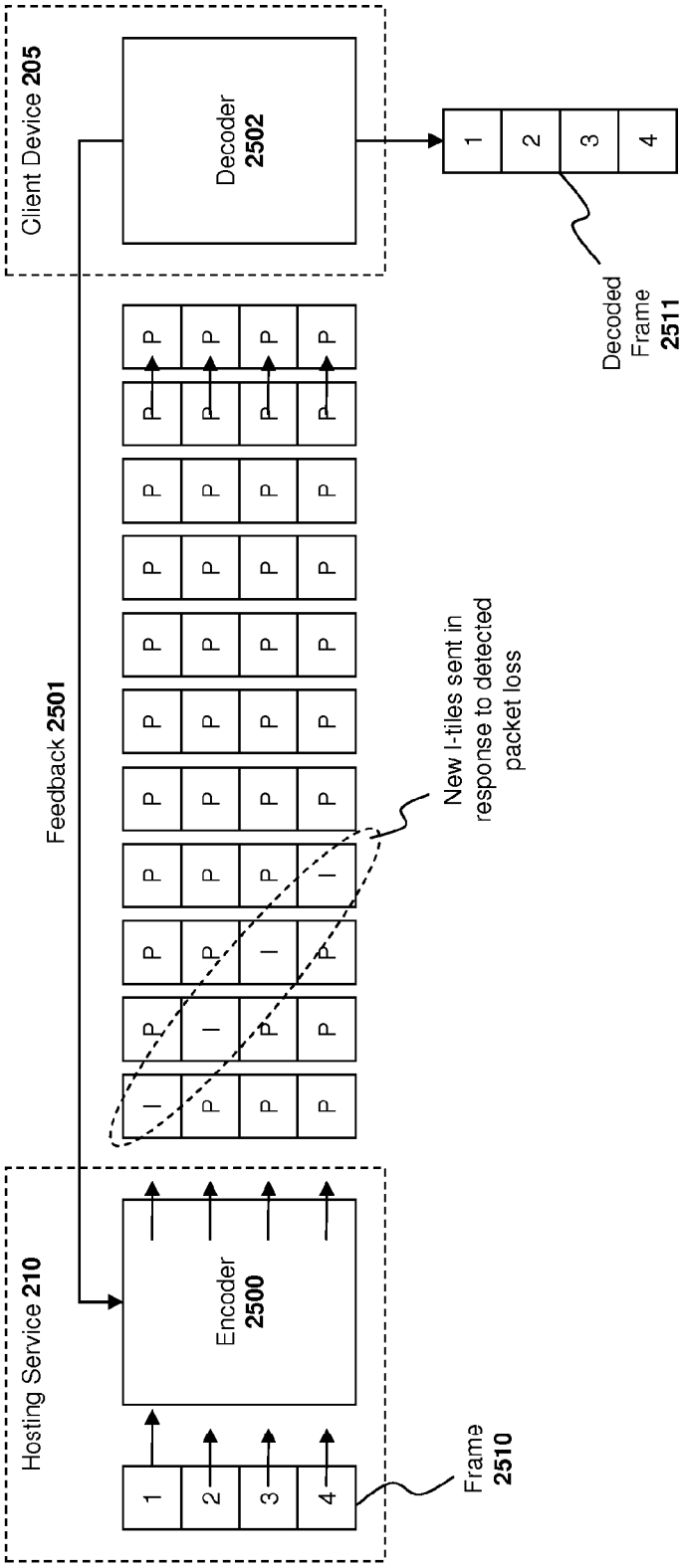

Client Device 205
Decoder 2502
1
2
3
4
Decoded Frame 2511
Feedback 2501
New I-tiles sent in response to detected packet loss
Hosting Service 210
Encoder 2500
Frame 2510

Fig. 25b

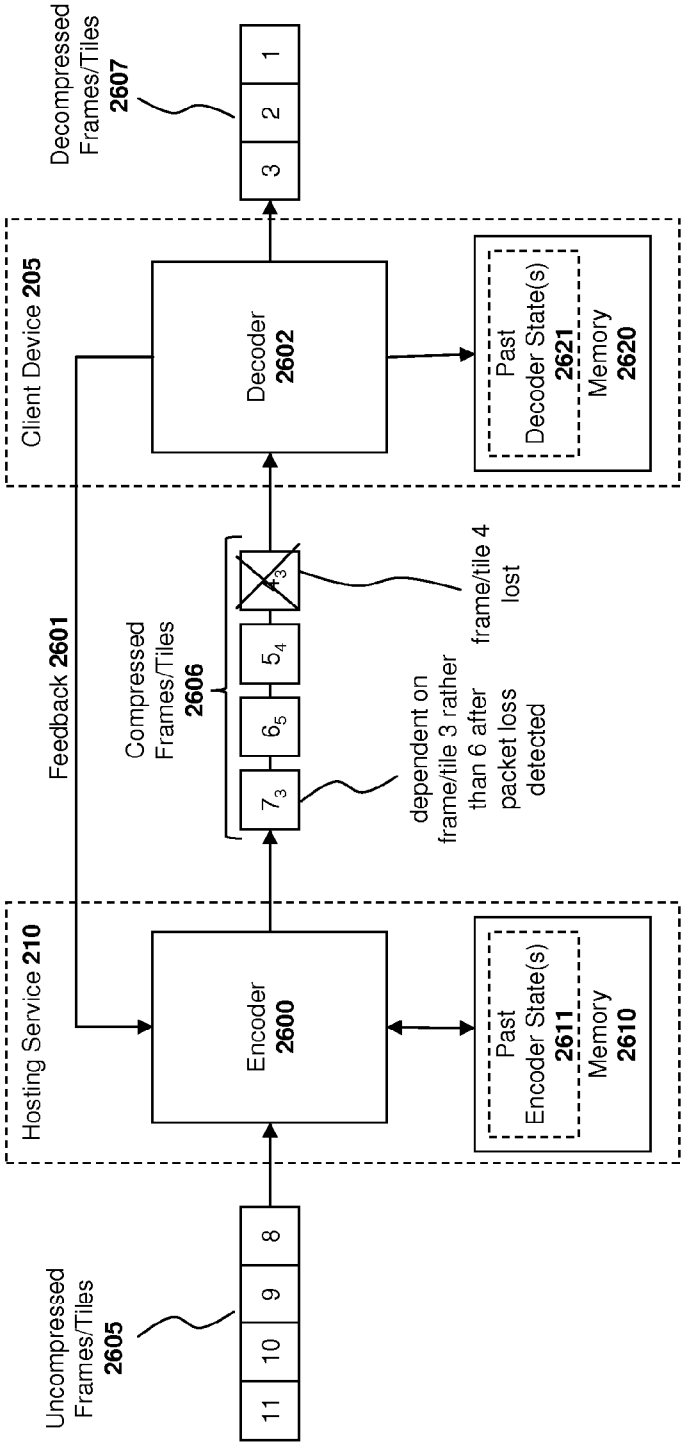

Uncompressed Frames/Tiles 2605
11
10
9
8
Hosting Service 210
Encoder 2600
Past Encoder State(s) 2611
Memory 2610
Feedback 2601
Compressed Frames/Tiles 2606
7_3
6_5
5_4
4_3
dependent on frame/tile 3 rather than 6 after packet loss detected
frame/tile 4 lost
Client Device 205
Decoder 2602
Past Decoder State(s) 2621
Memory 2620
Decompressed Frames/Tiles 2607
3
2
1

Fig. 26a

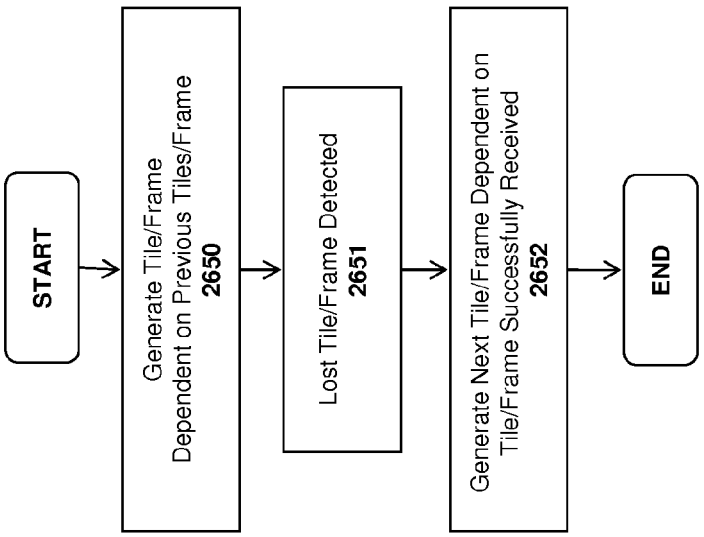

START
Generate Tile/Frame Dependent on Previous Tiles/Frame 2650
Lost Tile/Frame Detected 2651
Generate Next Tile/Frame Dependent on Tile/Frame Successfully Received 2652
END

Fig. 26b

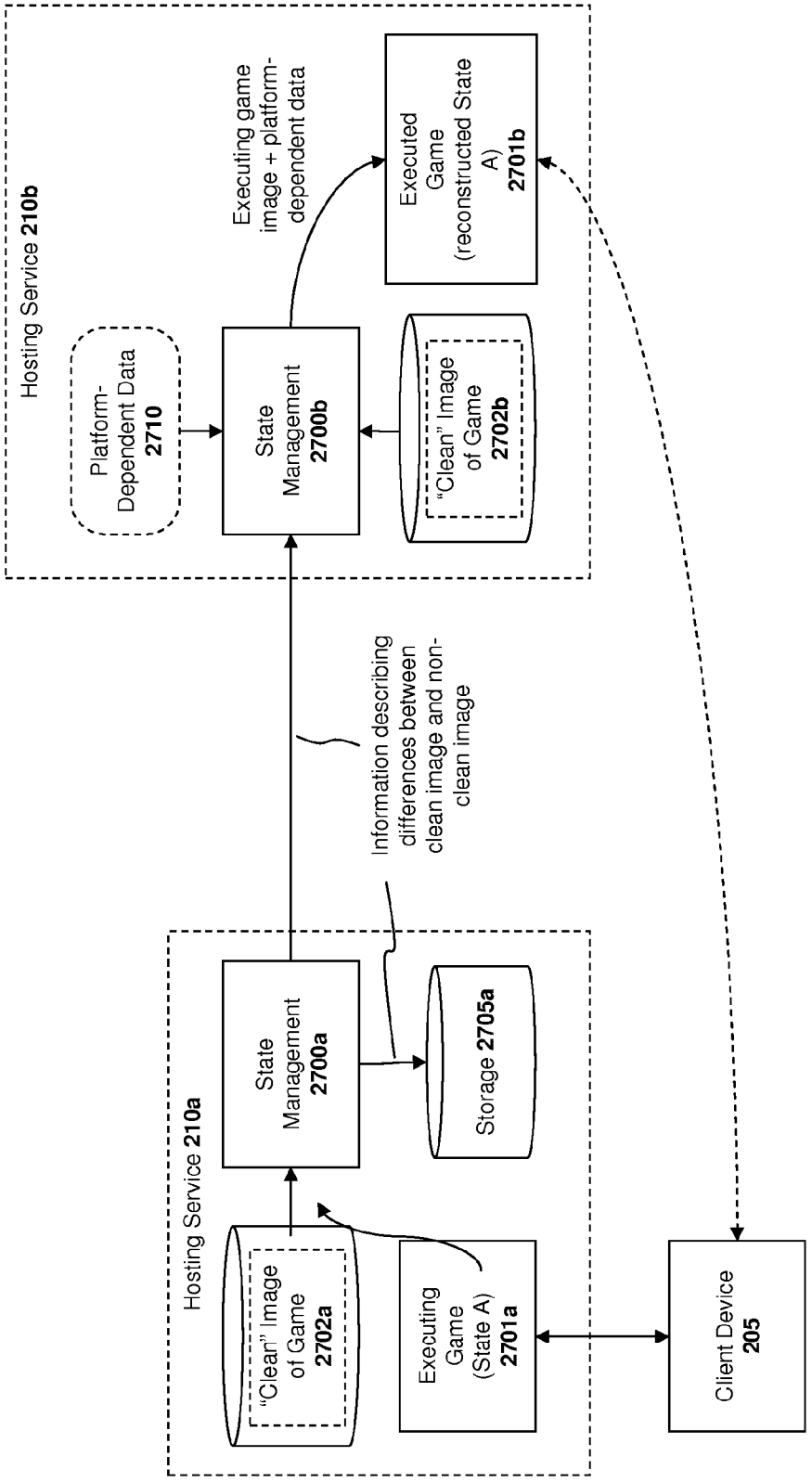

Hosting Service 210b
Platform-Dependent Data 2710
State Management 2700b
"Clean" Image of Game 2702b
Executing game image + platform-dependent data
Executed Game (reconstructed State A) 2701b
Information describing differences between clean image and non-clean image
Hosting Service 210a
State Management 2700a
Storage 2705a
"Clean" Image of Game 2702a
Executing Game (State A) 2701a
Client Device 205

Fig. 27a

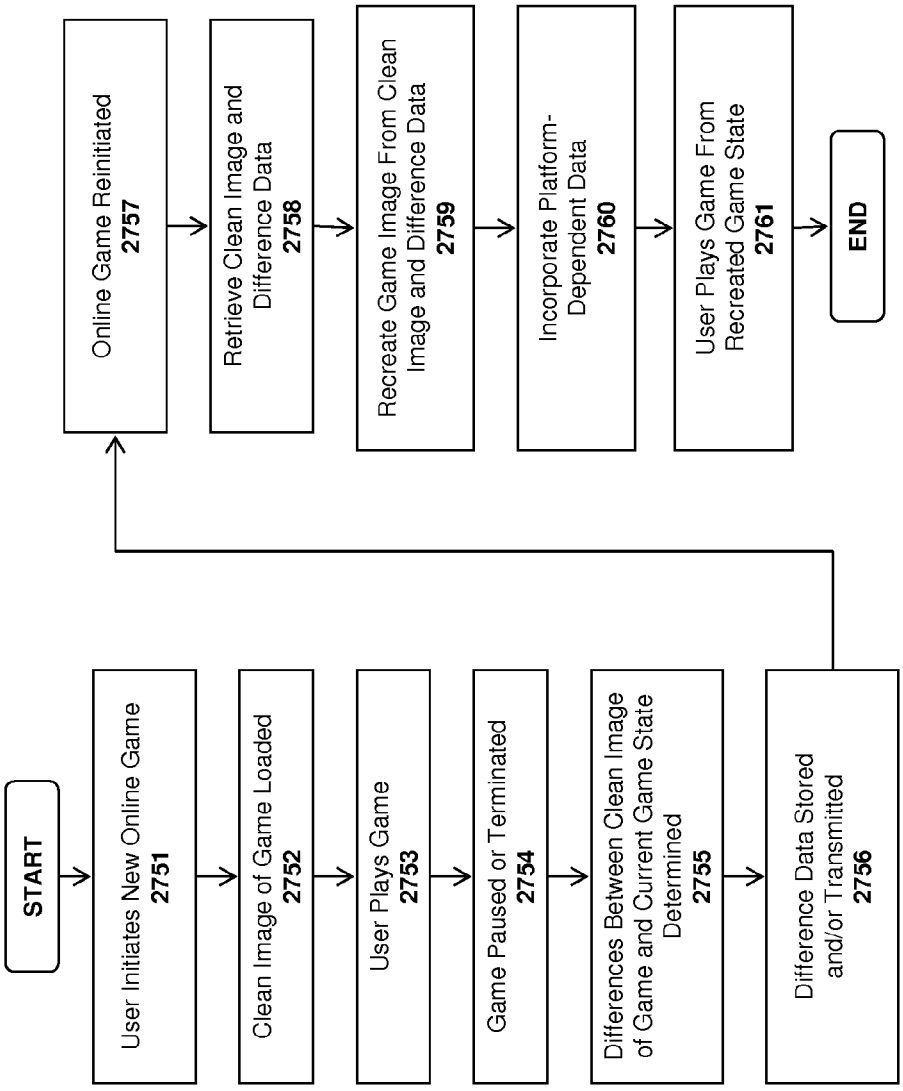

START
User Initiates New Online Game
2751
Clean Image of Game Loaded
2752
User Plays Game
2753
Game Paused or Terminated
2754
Differences Between Clean Image of Game and Current Game State Determined
2755
Difference Data Stored and/or Transmitted
2756
Online Game Reinitiated
2757
Retrieve Clean Image and Difference Data
2758
Recreate Game Image From Clean Image and Difference Data
2759
Incorporate Platform-Dependent Data
2760
User Plays Game From Recreated Game State
2761
END

Fig. 27b

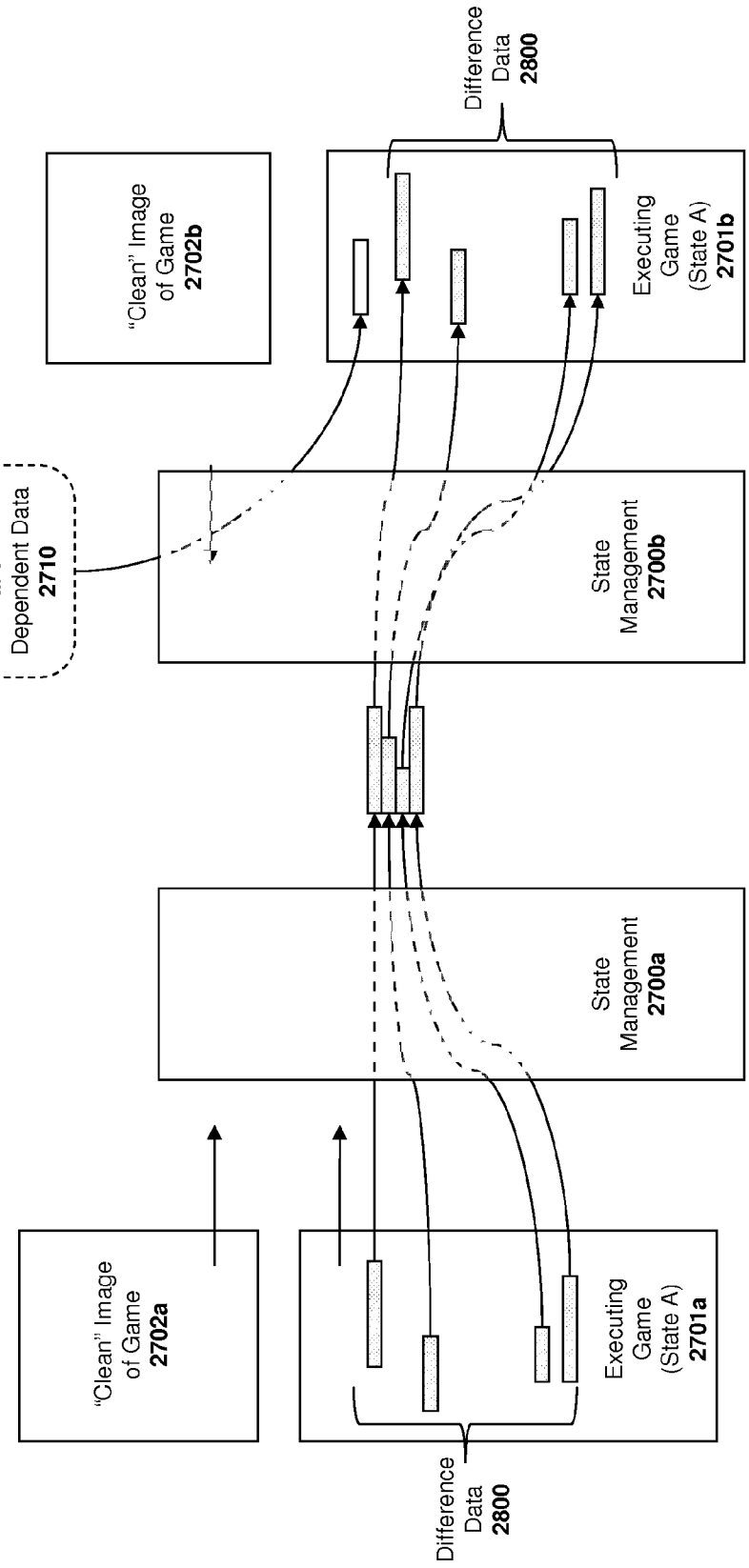

Difference Data 2800
"Clean" Image of Game 2702b
Executing Game (State A) 2701b
Platform-Dependent Data 2710
State Management 2700b
State Management 2700a
"Clean" Image of Game 2702a
Executing Game (State A) 2701a
Difference Data 2800

Fig. 28

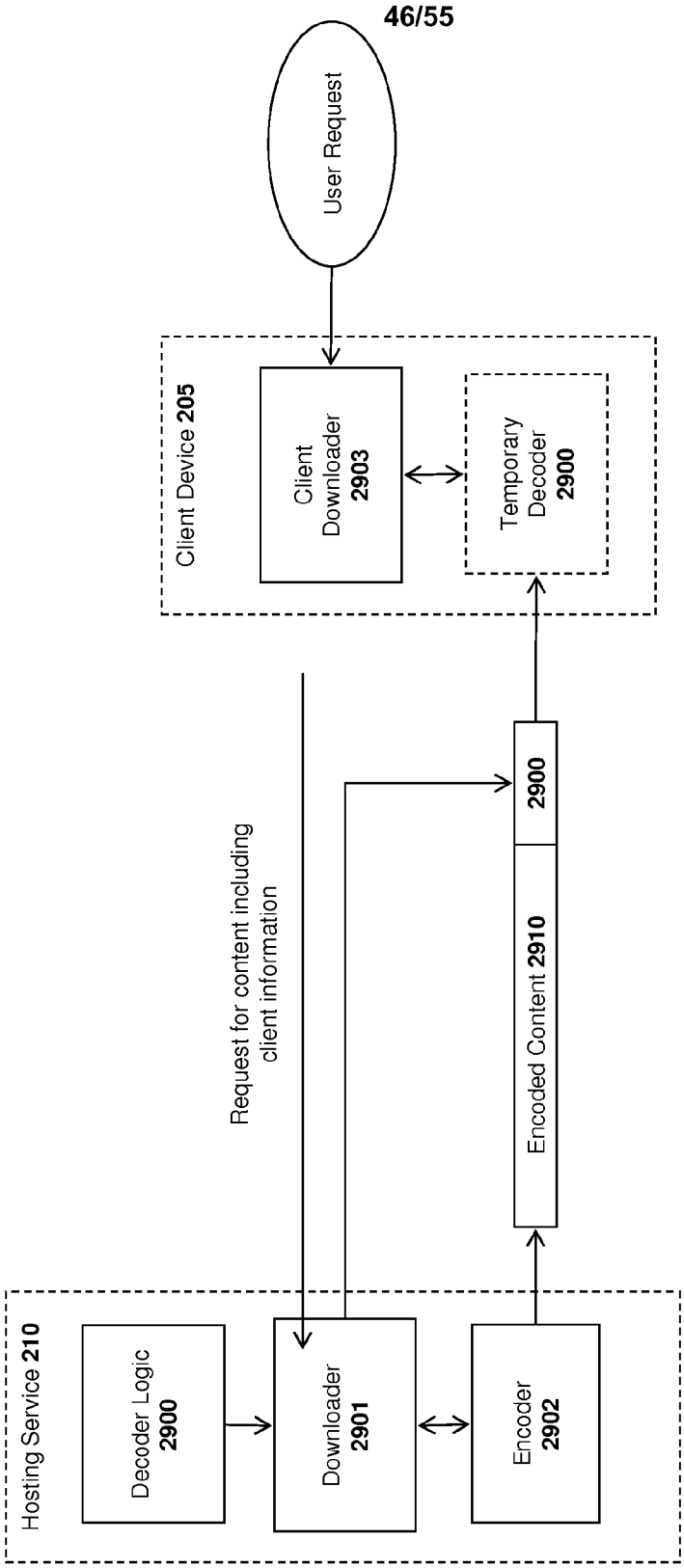
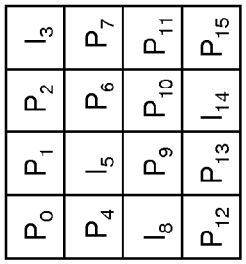
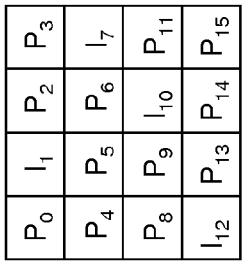
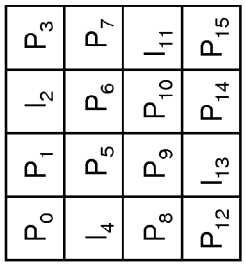

User Request
Client Device 205
Client Downloader 2903
Temporary Decoder 2900
Request for content including client information
2900
Encoded Content 2910
Hosting Service 210
Decoder Logic 2900
Downloader 2901
Encoder 2902

Fig. 29


I_0 P_1 P_2 P_3
P_4 P_5 I_6 P_7
P_8 I_9 P_{10} P_{11}
P_{12} P_{13} P_{14} I_{15}
R Frame
3001


P_0 I_1 P_2 P_3
P_4 P_5 P_6 I_7
P_8 P_9 I_{10} P_{11}
I_{12} P_{13} P_{14} P_{15}
R Frame
3002


P_0 P_1 I_2 P_3
I_4 P_5 P_6 P_7
P_8 P_9 P_{10} I_{11}
P_{12} I_{13} P_{14} P_{15}
R Frame
3003

P_0 P_1 P_2 I_3
P_4 I_5 P_6 P_7
I_8 P_9 P_{10} P_{11}
P_{12} P_{13} I_{14} P_{15}
R Frame
3004

Fig. 30

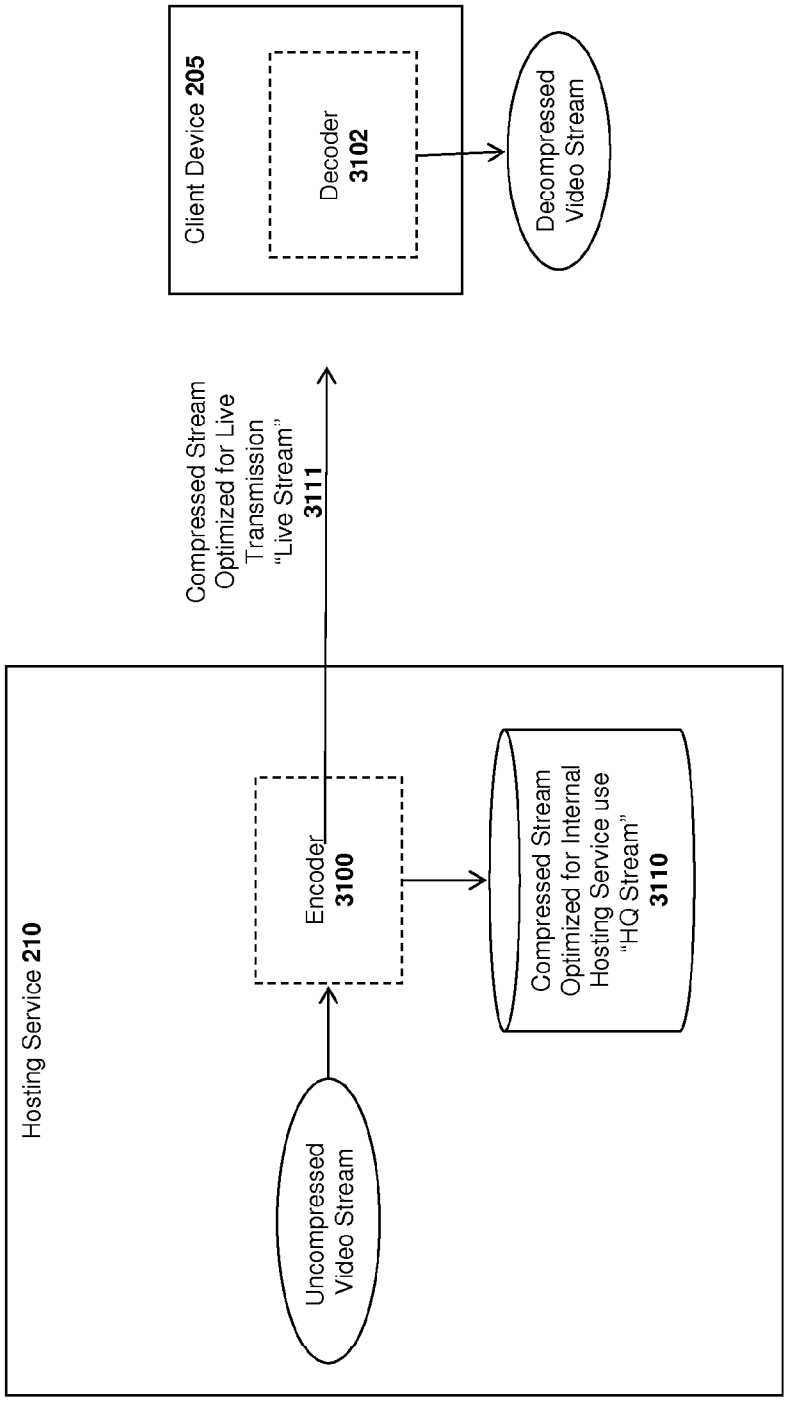

Client Device 205
Decoder 3102
Decompressed Video Stream
Compressed Stream Optimized for Live Transmission "Live Stream" 3111
Hosting Service 210
Encoder 3100
Compressed Stream Optimized for Internal Hosting Service use "HQ Stream" 3110
Uncompressed Video Stream

Fig. 31a

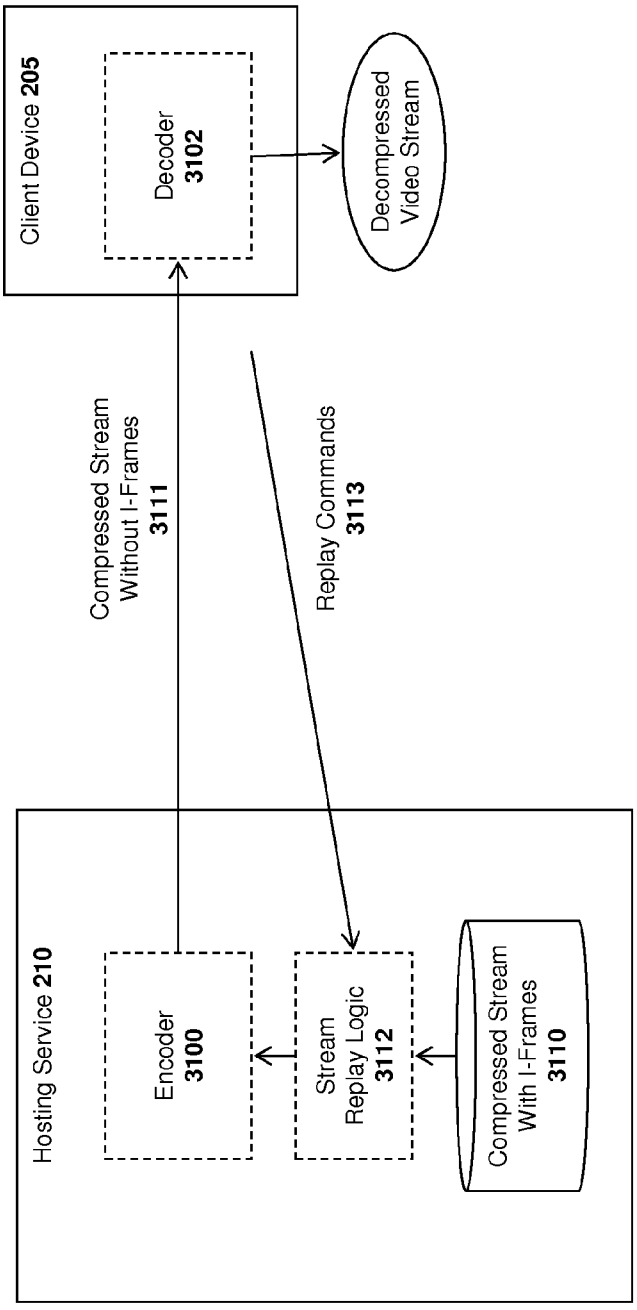

Client Device 205
Decoder 3102
Decompressed Video Stream
Compressed Stream Without I-Frames 3111
Replay Commands 3113
Hosting Service 210
Encoder 3100
Stream Replay Logic 3112
Compressed Stream With I-Frames 3110

Fig. 31b

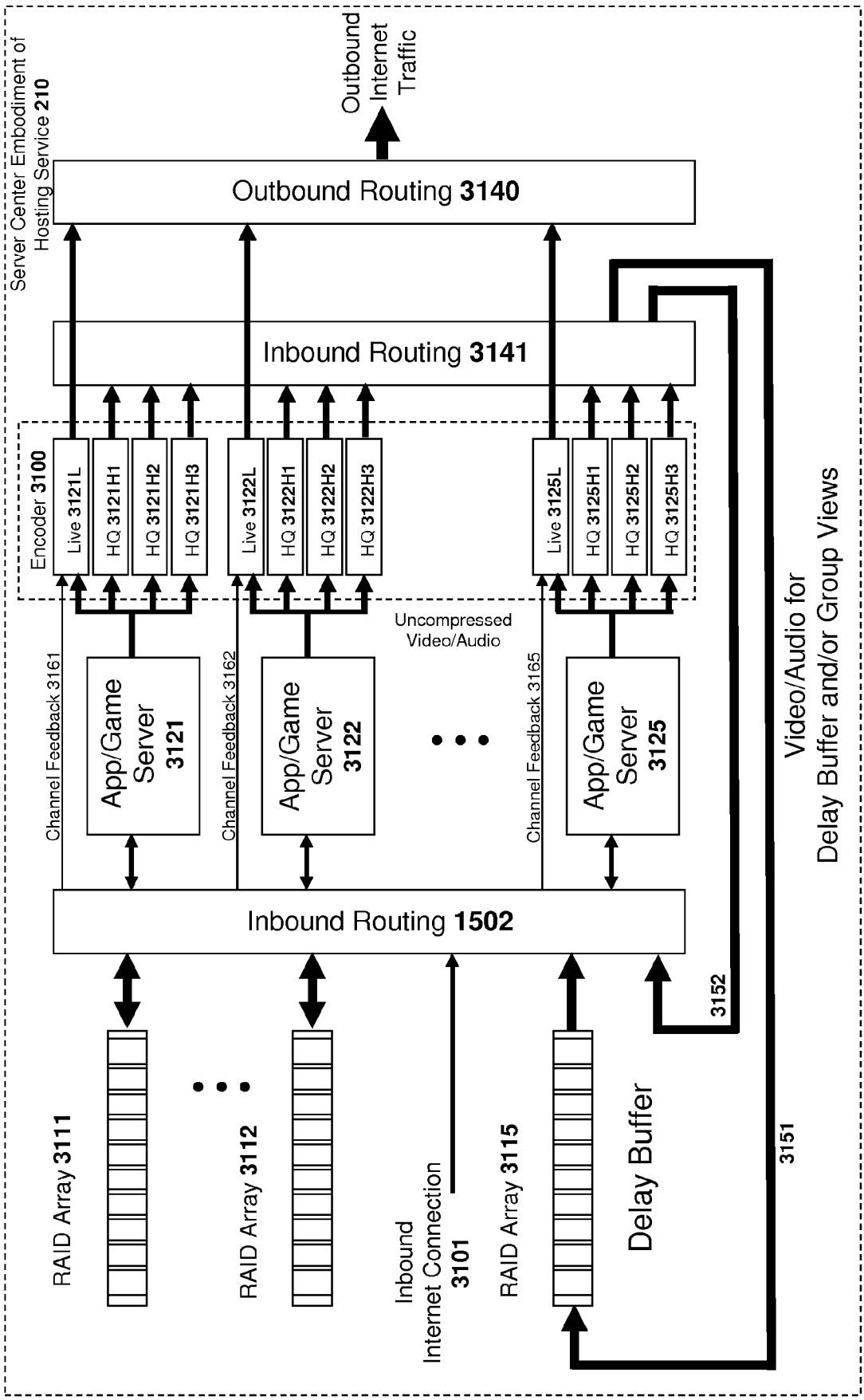

Server Center Embodiment of Hosting Service 210
Outbound Internet Traffic
Outbound Routing 3140
Inbound Routing 3141
Encoder 3100
Live 3121L
HQ 3121H1
HQ 3121H2
HQ 3121H3
Live 3122L
HQ 3122H1
HQ 3122H2
HQ 3122H3
Live 3125L
HQ 3125H1
HQ 3125H2
HQ 3125H3
Uncompressed Video/Audio
Channel Feedback 3161
Channel Feedback 3162
Channel Feedback 3165
App/Game Server 3121
App/Game Server 3122
App/Game Server 3125
Inbound Routing 1502
3152
Video/Audio for Delay Buffer and/or Group Views
3151
RAID Array 3111
RAID Array 3112
Inbound Internet Connection 3101
RAID Array 3115
Delay Buffer

Fig. 31c

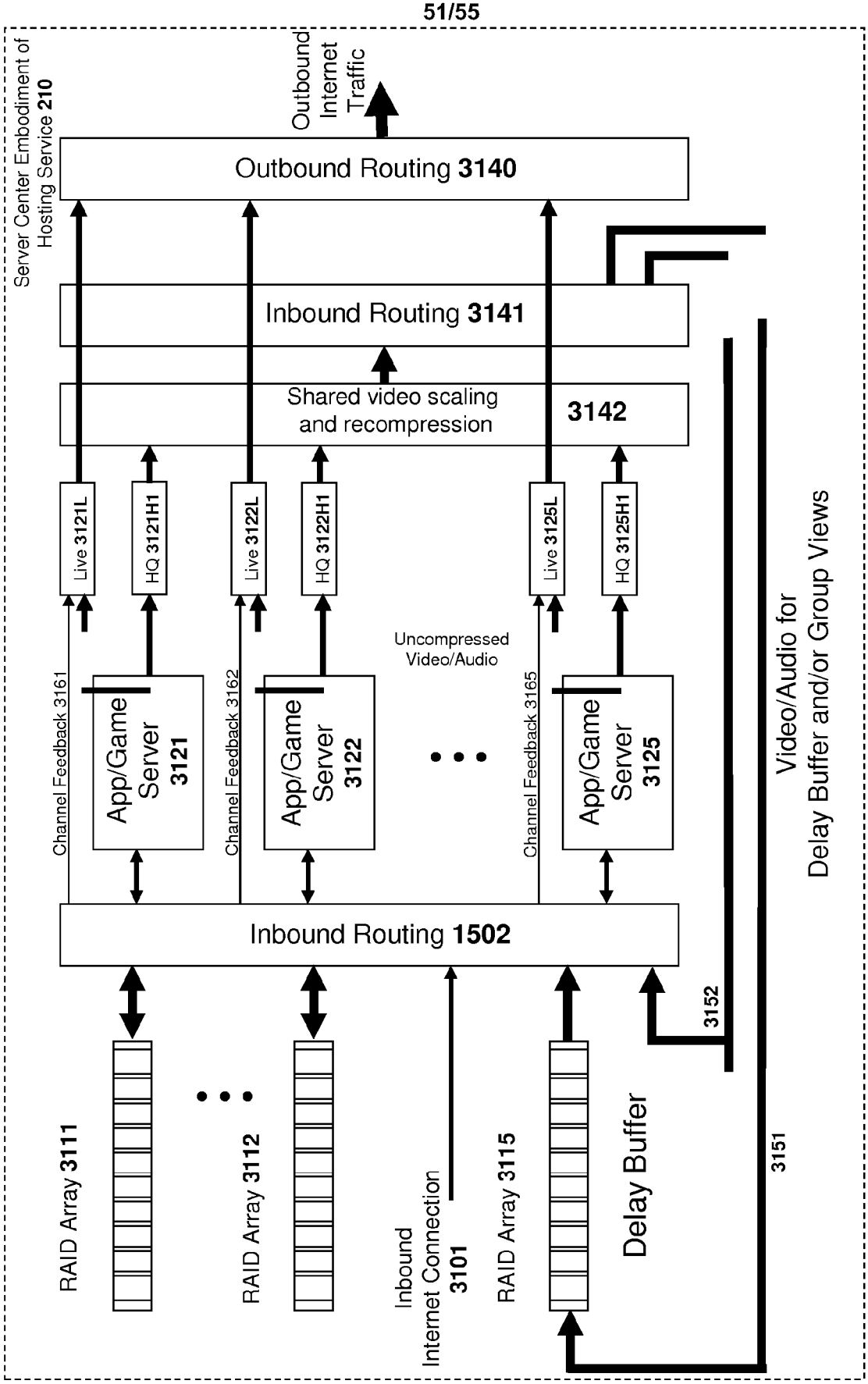

Server Center Embodiment of Hosting Service 210
Outbound Internet Traffic
Outbound Routing 3140
Inbound Routing 3141
Shared video scaling and recompression
3142
Live 3121L
HQ 3121H1
Live 3122L
HQ 3122H1
Live 3125L
HQ 3125H1
Uncompressed Video/Audio
Channel Feedback 3161
Channel Feedback 3162
Channel Feedback 3165
App/Game Server 3121
App/Game Server 3122
App/Game Server 3125
Inbound Routing 1502
3152
Video/Audio for Delay Buffer and/or Group Views
RAID Array 3111
RAID Array 3112
Inbound Internet Connection 3101
RAID Array 3115
Delay Buffer
3151

Fig. 31d

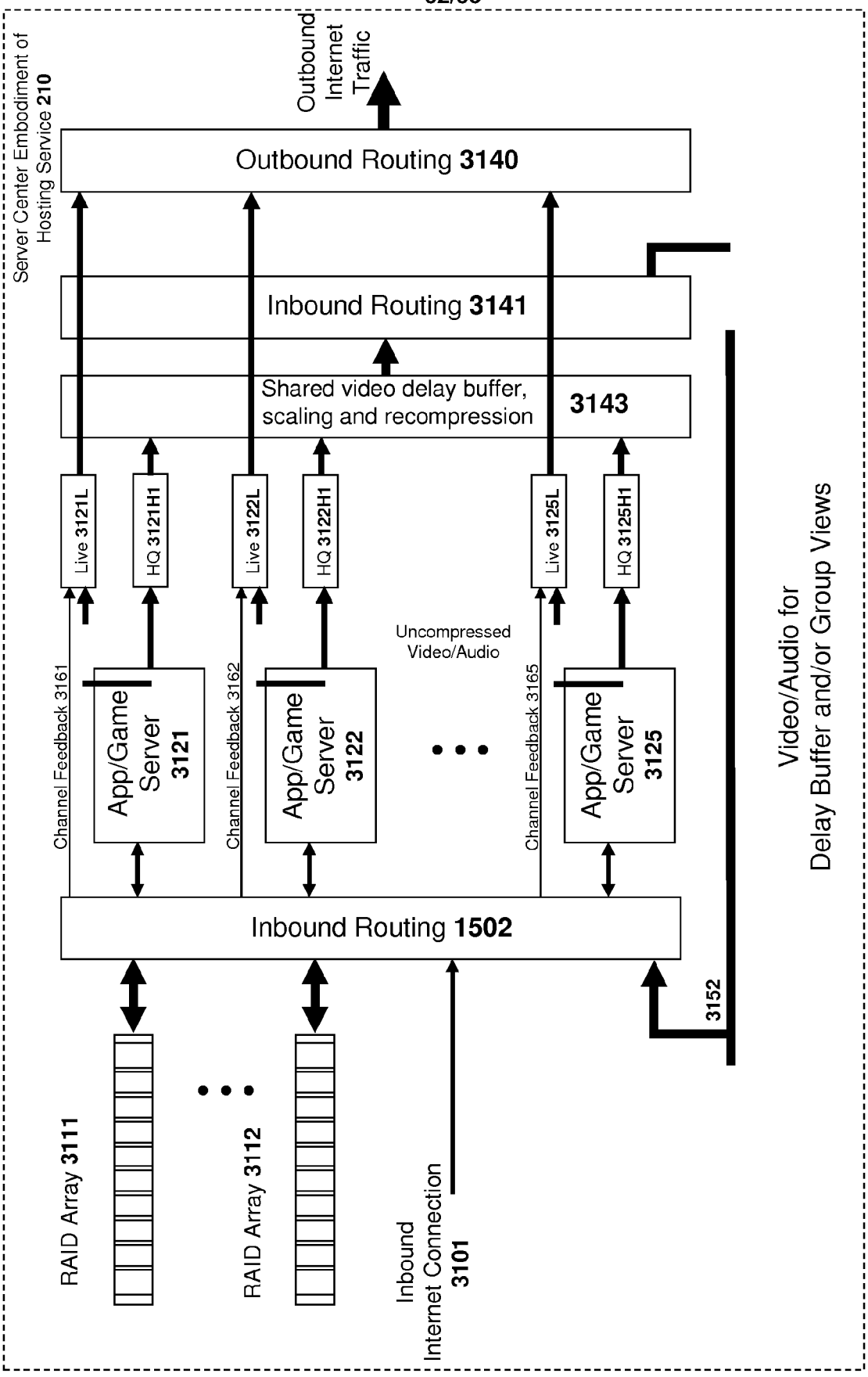

Server Center Embodiment of Hosting Service 210
Outbound Internet Traffic
Outbound Routing 3140
Inbound Routing 3141
Shared video delay buffer, scaling and recompression
3143
Live 3121L
HQ 3121H1
Live 3122L
HQ 3122H1
Live 3125L
HQ 3125H1
Uncompressed Video/Audio
Channel Feedback 3161
Channel Feedback 3162
Channel Feedback 3165
App/Game Server 3121
App/Game Server 3122
App/Game Server 3125
Inbound Routing 1502
3152
Video/Audio for Delay Buffer and/or Group Views
RAID Array 3111
RAID Array 3112
Inbound Internet Connection 3101

Fig. 31e

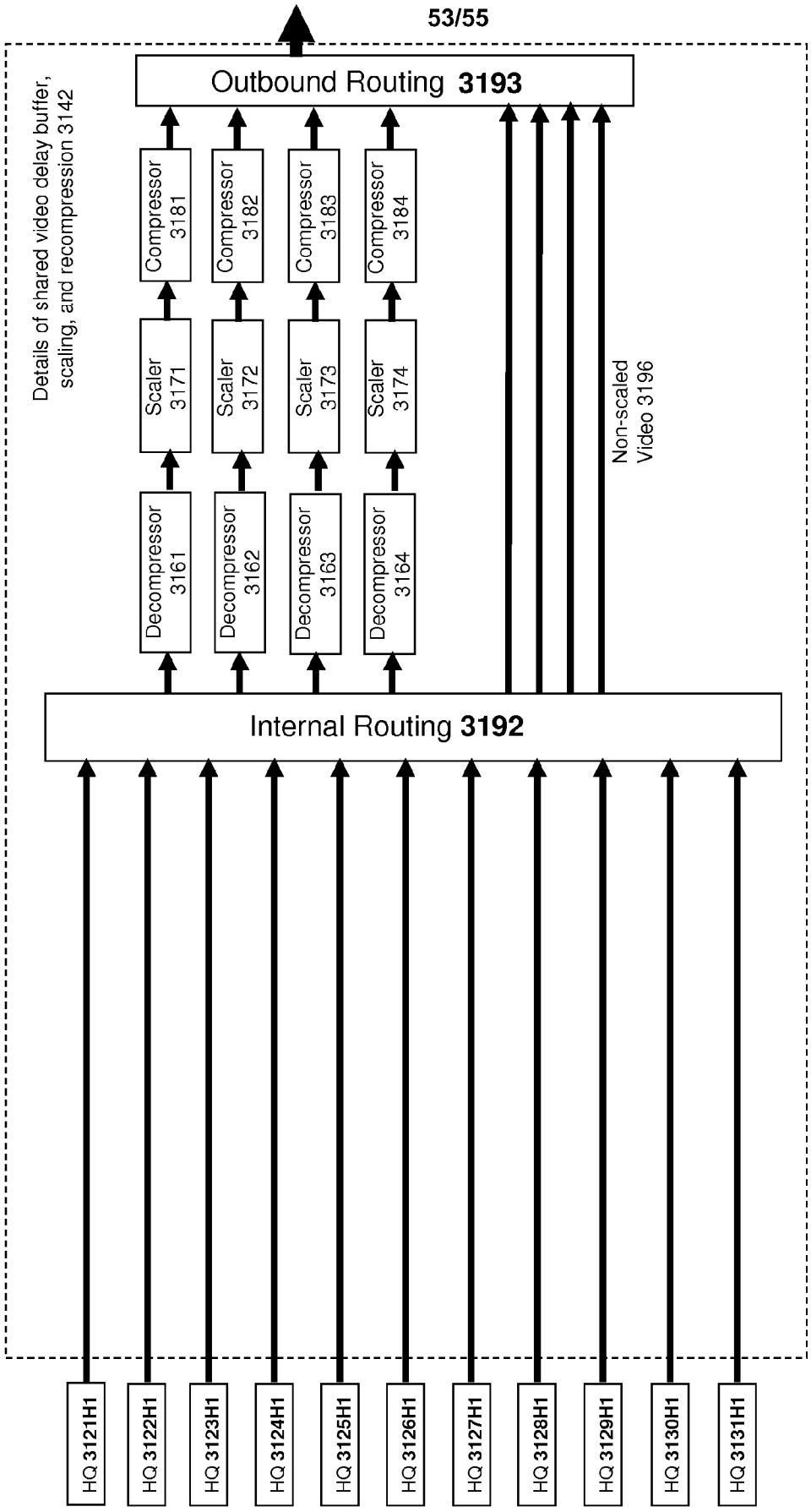

53/55
Outbound Routing 3193
Details of shared video delay buffer, scaling, and recompression 3142
Compressor 3181
Compressor 3182
Compressor 3183
Compressor 3184
Scaler 3171
Scaler 3172
Scaler 3173
Scaler 3174
Decompressor 3161
Decompressor 3162
Decompressor 3163
Decompressor 3164
Non-scaled Video 3196
Internal Routing 3192
HQ 3121H1
HQ 3122H1
HQ 3123H1
HQ 3124H1
HQ 3125H1
HQ 3126H1
HQ 3127H1
HQ 3128H1
HQ 3129H1
HQ 3130H1
HQ 3131H1

Fig. 31f

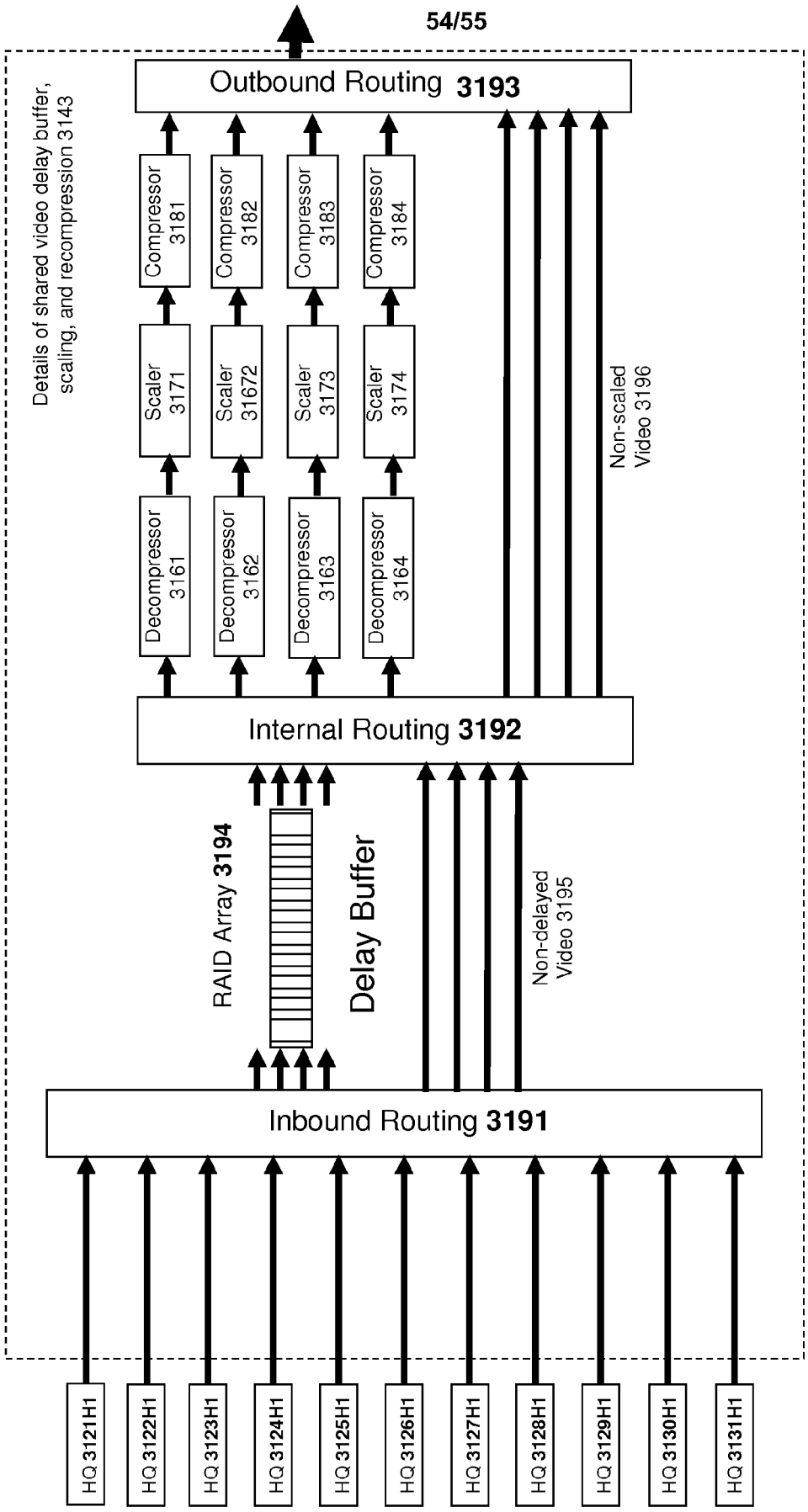

54/55
Outbound Routing 3193
Details of shared video delay buffer, scaling, and recompression 3143
Compressor 3181
Compressor 3182
Compressor 3183
Compressor 3184
Scaler 3171
Scaler 31672
Scaler 3173
Scaler 3174
Decompressor 3161
Decompressor 3162
Decompressor 3163
Decompressor 3164
Non-scaled Video 3196
Internal Routing 3192
RAID Array 3194
Delay Buffer
Non-delayed Video 3195
Inbound Routing 3191
HQ 3121H1
HQ 3122H1
HQ 3123H1
HQ 3124H1
HQ 3125H1
HQ 3126H1
HQ 3127H1
HQ 3128H1
HQ 3129H1
HQ 3130H1
HQ 3131H1

Fig. 31g

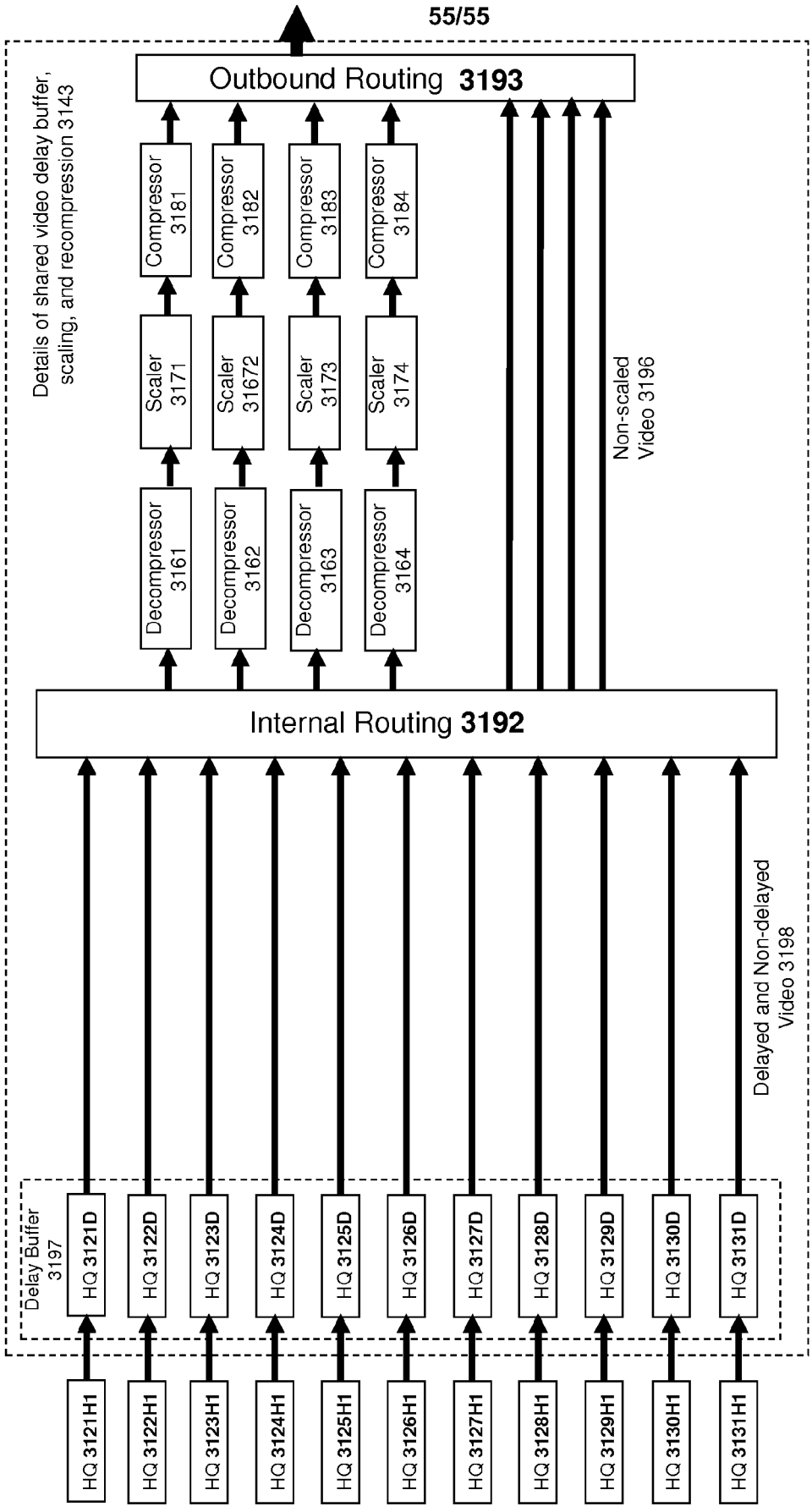

Outbound Routing 3193
Details of shared video delay buffer, scaling, and recompression 3143
Compressor 3181
Compressor 3182
Compressor 3183
Compressor 3184
Scaler 3171
Scaler 31672
Scaler 3173
Scaler 3174
Decompressor 3161
Decompressor 3162
Decompressor 3163
Decompressor 3164
Non-scaled Video 3196
Internal Routing 3192
Delayed and Non-delayed Video 3198
Delay Buffer 3197
HQ 3121D
HQ 3122D
HQ 3123D
HQ 3124D
HQ 3125D
HQ 3126D
HQ 3127D
HQ 3128D
HQ 3129D
HQ 3130D
HQ 3131D
HQ 3121H1
HQ 3122H1
HQ 3123H1
HQ 3124H1
HQ 3125H1
HQ 3126H1
HQ 3127H1
HQ 3128H1
HQ 3129H1
HQ 3130H1
HQ 3131H1

Fig. 31h

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 10/27712

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04N 7/12 (2010.01)
USPC - 348/404.1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
USPC: 348/404.1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 382/239; 714/752; 714/E11.001; 463/4; 348/425.2 (keyword limited - see search terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PubWEST (PGPB, USPT, USOC, EPAB, JPAB); GOOGLE; Google Scholar
Terms: video, compression, encoding, decoding, previously, frame, successful, p-frame, tile, images, predictive, memory, buffer, feedback.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 7,016,419 B2 (Tardif) 21 March 2006 (21.03.2006), entire document, especially abstract, col. 1, ln 19-54, col. 2, ln 10-42, col. 3, ln 30-54, col. 4, ln 6 -12, col. 7, ln 31-38, col. 10, ln 58 to col. 11, ln 17.	1-24
Y	US 2009/0003452 A1 (Au et al.) 01 January 2009 (01.01.2009), entire document, especially abstract, para [0004], [0005], [0006], [0050], [0109].	1-24
A	US 2007/0073779 A1 (Walker et al.) 29 March 2007 (29.03.2007), entire document, especially abstract, para [0009], [0048], [0060], [0071].	1-24

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 April 2010 (26.04.2010)	11 MAY 2010
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)

(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102428699 A
(43)申请公布日 2012.04.25

(21)申请号 201080021758.3

(22)申请日 2010.03.17

(30)优先权数据
61/210,888 2009.03.23 US
12/538,096 2009.08.07 US

(85)PCT申请进入国家阶段日
2011.11.18

(86)PCT申请的申请数据
PCT/US2010/027712 2010.03.17

(87)PCT申请的公布数据
WO2010/111095 EN 2010.09.30

(71)申请人 生命力有限公司
地址 美国加利福尼亚州

(72)发明人 S·G·珀尔曼 R·范德拉安
T·科特 S·弗曼 R·麦库尔
I·巴克利

(74)专利代理机构 北京润平知识产权代理有限公司 11283
代理人 肖冰滨 南毅宁

(51)Int.Cl.
H04N 7/12(2006.01)

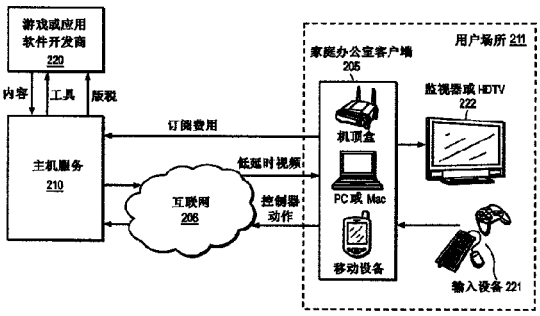
权利要求书 3 页 说明书 77 页 附图 53 页

(54)发明名称

使用包含有关视频内容的成功接收的数据的反馈进行视频压缩的系统及方法

(57)摘要

本发明描述一种用于执行视频压缩的计算机实施的系统和方法。根据一实施例的方法包括对第一多个视频帧或该视频帧的部分进行编码，其中每个经编码的视频帧或该视频帧的部分分别依赖于先前经编码的视频帧或该视频帧的部分。该方法还包括将所述第一多个经编码的视频帧或该视频帧的部分发送给客户端设备，和从所述客户端设备接收反馈信息，所述反馈信息可用于确定包括在所述视频帧或部分中的数据是否未被成功接收和/或解码；响应于检测到视频帧未被成功接收和/或解码，将当前视频帧或该视频帧的部分编码为依赖于已知在所述客户端设备上已被成功接收和/或解码的先前经编码的视频帧或该视频帧的部分；以及将所述当前的视频帧或该视频帧的部分发送到所述客户端设备。



CN 102428699 A

1. 一种用于执行视频压缩的计算机实施的方法，该方法包括：

对第一多个视频帧或该第一多个视频帧的部分进行编码，其中每个经编码的视频帧或该经编码的视频帧的部分分别依赖于先前经编码的视频帧或该先前经编码的视频帧的部分；

将所述第一多个经编码的视频帧或部分传送给客户端设备；

从所述客户端设备接收反馈信息，所述反馈信息能够用于确定包含在所述视频帧或部分中的数据是否未被成功接收和 / 或解码；

响应于检测到视频帧或该视频帧的部分未被成功接收和 / 或解码，将当前视频帧或该当前视频帧的部分编码为依赖于已知在所述客户端设备上已被成功接收和 / 或解码的先前经编码的视频帧或该先前经编码的视频帧的部分；以及

将所述当前视频帧或该当前视频帧的部分传送到所述客户端设备。

2. 根据权利要求 1 所述的方法，其中已知已被成功接收和 / 或解码的所述先前经编码的视频帧或该先前经编码的视频帧的部分包括最后已知的已知已被成功接收和 / 或解码的视频帧或该视频帧的部分。

3. 根据权利要求 1 所述的方法，其中所述视频帧或该视频帧的部分分别包括 P 帧或 P 图像块。

4. 根据权利要求 1 所述的方法，其中所述反馈信息包括所述帧或该帧的部分已在所述客户端设备处被成功接收和 / 或解码的指示。

5. 根据权利要求 1 所述的方法，其中所述反馈信息包括所述帧或该帧的部分未在所述客户端设备处被成功接收和 / 或解码的指示。

6. 根据权利要求 1 所述的方法，其中将当前视频帧或该当前视频帧的部分编码为依赖于已知在所述客户端设备上已被成功接收和 / 或解码的先前经编码的视频帧或该先前经编码的视频帧的部分还包括：

从存储器获取先前状态，所述先前状态包括对所述先前经编码的视频帧或部分进行编码后编码器的状态。

7. 根据权利要求 6 所述的方法，该方法包括：

对所述当前视频帧或该当前视频帧的部分进行解码，其中解码包括从存储器获取先前状态，所述先前状态包括对所述先前经编码的视频帧或部分进行解码后解码器的状态。

8. 根据权利要求 1 所述的方法，该方法还包括：

在所述客户端设备处对所述经编码的视频帧或该经编码的视频帧的部分进行解码；以及

在所述客户端设备上的显示器上显示与每个所述视频帧或该视频帧的部分相关的图像。

9. 一种包括存储器和处理器的系统，所述存储器用于存储程序代码，所述处理器用于处理所述程序代码以执行以下操作：

对第一多个视频帧或该第一多个视频帧的部分进行编码，其中每个经编码的视频帧或该经编码的视频帧的部分分别依赖于先前经编码的视频帧或该先前经编码的视频帧的部分；

将所述第一多个经编码的视频帧或部分传送给客户端设备；

从所述客户端设备接收反馈信息，所述反馈信息能够用于确定包含在所述视频帧或部分中的数据是否未被成功接收和 / 或解码 ；

响应于检测到视频帧或该视频帧的部分未被成功接收和 / 或解码，将当前视频帧或该当前视频帧的部分编码为依赖于已知在所述客户端设备上已被成功接收和 / 或解码的先前经编码的视频帧或该先前经编码的视频帧的部分 ；以及

将所述当前视频帧或该当前视频帧的部分传送到所述客户端设备。

10. 根据权利要求 9 所述的系统，其中已知已被成功接收和 / 或解码的所述先前经编码的视频帧或该先前经编码的视频帧的部分包括最后已知的已知已被成功接收和 / 或解码的视频帧或该视频帧的部分。

11. 根据权利要求 9 所述的系统，其中所述视频帧或该视频帧的部分分别包括 P 帧或 P 图像块。

12. 根据权利要求 9 所述的系统，其中所述反馈信息包括所述帧或该帧的部分已在所述客户端设备处被成功接收和 / 或解码的指示。

13. 根据权利要求 9 所述的系统，其中所述反馈信息包括所述帧或该帧的部分未在所述客户端设备处被成功接收和 / 或解码的指示。

14. 根据权利要求 9 所述的系统，其中将当前视频帧或该当前视频帧的部分编码为依赖于已知已在所述客户端设备上已被成功接收和 / 或解码的先前经编码的视频帧或该先前经编码的视频帧的部分还包括 ：

从存储器获取先前状态，所述先前状态包括对所述先前经编码的视频帧或部分进行编码后编码器的状态。

15. 根据权利要求 14 所述的系统，该系统包括使得所述处理器执行以下操作的附加程序代码 ：

对所述当前视频帧或该当前视频帧的部分进行解码，其中解码包括从存储器获取先前状态，所述先前状态包括对所述先前经编码的视频帧或部分进行解码后解码器的状态。

16. 根据权利要求 9 所述的系统，该系统包括使得所述处理器执行以下操作的附加程序代码 ：

在所述客户端设备处对所述经编码的视频帧或该经编码的视频帧的部分进行解码 ；以及

在所述客户端设备上的显示器上显示与每个所述视频帧或该视频帧的部分相关的图像。

17. 一种具有存储于其上的程序代码的机器可读介质，当所述程序代码由机器执行时促使所述机器执行下述操作 ：

对第一多个视频帧或该第一多个视频帧的部分进行编码，其中每个经编码的视频帧或该经编码的视频帧的部分分别依赖于先前经编码的视频帧或该先前经编码的视频帧的部分 ；

将所述第一多个经编码的视频帧或部分传送给客户端设备 ；

从所述客户端设备接收反馈信息，所述反馈信息能够用于确定包含在所述视频帧或部分中的数据是否未被成功接收和 / 或解码 ；

响应于检测到视频帧或该视频帧的部分未被成功接收和 / 或解码，将当前视频帧或该

当前视频帧的部分编码为依赖于已知在所述客户端设备上已被成功接收和 / 或解码的先前经编码的视频帧或该先前经编码的视频帧的部分 ;以及

将所述当前视频帧或该当前视频帧的部分传送到所述客户端设备。

18. 根据权利要求 17 所述的机器可读介质,其中已知已被成功接收和 / 或解码的所述先前经编码的视频帧或该先前经编码的视频帧的部分包括最后已知的已知已被成功接收和 / 或解码的视频帧或该视频帧的部分。

19. 根据权利要求 17 所述的机器可读介质,其中所述视频帧或该视频帧的部分分别包括 P 帧或 P 图像块。

20. 根据权利要求 17 所述的机器可读介质,其中所述反馈信息包括所述帧或该帧的部分已在所述客户端设备处被成功接收和 / 或解码的指示。

21. 根据权利要求 17 所述的机器可读介质,其中所述反馈信息包括所述帧或该帧的部分未在所述客户端设备处被成功接收和 / 或解码的指示。

22. 根据权利要求 17 所述的机器可读介质,其中将当前视频帧或该当前视频帧的部分编码为依赖于已知已在所述客户端设备上被成功接收和 / 或解码的先前经编码的视频帧或该先前经编码的视频帧的部分还包括 :

从存储器获取先前状态,所述先前状态包括对所述先前经编码的视频帧或部分进行编码后编码器的状态。

23. 根据权利要求 22 所述的机器可读介质,该机器可读介质包括使得所述处理器执行以下操作的附加程序代码 :

对所述当前视频帧或该当前视频帧的部分进行解码,其中解码包括从存储器获取先前状态,所述先前状态包括对所述先前经编码的视频帧或部分进行解码后解码器的状态。

24. 根据权利要求 17 所述的机器可读介质,该机器可读介质包括使得所述处理器执行以下操作的附加程序代码 :

在所述客户端设备处对所述经编码的视频帧或该经编码的视频帧的部分进行解码 ;以及

在所述客户端设备上的显示器上显示与每个所述视频帧或该视频帧的部分相关的图像。

使用包含有关视频内容的成功接收的数据的反馈进行视频压缩的系统及方法

相关申请

[0001] 本申请要求2009年3月23日提交的题为“System And Method For Compressing Video Using Feedback”美国临时专利申请No.61/210,888的优先权，该申请为2009年1月23日提交的题为“System And Method for Protecting Certain Types of Multimedia Data Transmitted Over A Communication Channel”的共同待决的美国申请No.12/359,150的部分继续申请，并为2007年12月5日提交的题为“Hosting And Broadcasting Virtual Events Using Streaming Interactive Video”的共同待决的美国申请No.11/999,475的继续申请，该美国申请No.11/999,475为2002年12月10日提交的、题为“Apparatus and Method for Wireless Video Gaming”的、第10/315,460号的部分继续(CIP)申请，该申请已转让给本CIP申请的受让人。

技术领域

[0002] 本公开总体上涉及改善用户操作及存取音频和视频媒体的能力的数据处理系统的领域。

背景技术

[0003] 自从托马斯·爱迪生(Thomas Edison)时代以来，记录的音频及电影媒体已成为社会的一方面。在20世纪初期，记录的音频媒体（磁柱及唱片）及电影媒体（投币式自动点唱机及电影）广泛发行，但两种技术仍处于其起步阶段。在20世纪20年代后期，在大众市场的基础上将电影与音频组合，之后将彩色电影与音频组合。无线电广播逐渐演变成很大程度上支持广告的形式的广播大众市场音频媒体。当在20世纪40年代中期建立电视(TV)广播标准时，电视与无线电以广播大众市场媒体的形式接合，从而将先前记录的或现场直播的电影带入家庭中。

[0004] 至20世纪中期为止，大部分美国家庭已具有用于播放记录的音频媒体的唱片播放器(phonograph record player)、用于接收现场直播的广播音频的无线电设备，及用于播放现场直播的广播音频/视频(A/V)媒体的电视机。常常将所述3个“媒体播放器”(唱片播放器、无线电设备及TV)组合到共有公共扬声器的橱柜中，变成家庭的“媒体中心”。尽管对于消费者而言媒体选择有限，但媒体“生态系统”非常稳定。大多数消费者知道如何使用“媒体播放器”且能够享受其能力的全部范围。同时，媒体的出版商（大多为电影和电视工作室，及音乐公司）能够将其媒体分配给电影院与家庭两者，而不遭受广泛的盗版或“二次销售”(也即，已使用媒体的重新销售)。通常，出版商不会从二次销售得到收入，且因此，二次销售减少了出版商对于新的销售从原本可自己使用媒体的购买者得到的收入。尽管在20世纪中期期间的确存有已使用的唱片出售，但所述销售不会对唱片出版商有大影响，因为不同于电影或视频节目（其通常被成年人观看一次或仅数次），音乐曲目可被收听数百次或甚至数千次。因此，音乐媒体远比电影/视频媒体“经久”(也即，对于成年消费者而

言，其具有持久价值）。一旦购买了唱片，若消费者喜欢该音乐，则消费者可能将其保持很长时间。

[0005] 自 20 世纪中期到现在，媒体生态系统对于消费者与出版商的利益及损失来说都经历了一系列根本改变。在音频录音机（尤其是具有高质量立体声的盒式磁带）的广泛引入的情况下，的确有较高程度的消费者便利。但其也标志着现在广泛的消费者媒体实践—盗版的开始。的确，许多消费者纯粹出于便利起见而使用盒式磁带来录制其自己的唱片，但日益增加的消费者（例如，宿舍中准备存取彼此的唱片收集的学生）将进行盗版复制。而且，消费者将录制经由无线电播放的音乐，而非从出版商购买唱片或磁带。

[0006] 消费者 VCR 的出现导致更多的消费者便利，因为现在 VCR 被设置为记录 TV 节目，其可在稍后时间观看，且 VCR 也导致视频租赁业的建立，其中电影以及 TV 节目设计可在“点播”基础上进行存取。自 20 世纪 80 年代中期以来的大众市场家庭媒体设备的快速开发已导致消费者的空前的选择及便利程度，且也导致媒体出版市场的快速扩张。

[0007] 现今，消费者面对过多媒体选择以及过多媒体设备，其中许多被绑定到特定形式的媒体或特定出版商。热衷的媒体消费者可能将一堆设备连接到场所各房间中的 TV 及计算机，造成至一或多个电视机和 / 或个人计算机 (PC) 的“鼠窝式”电缆以及一群远程控制。（在本申请的上下文中，术语“个人计算机”或“PC”指代适合于在家庭或办公室中使用的任何种类的计算机），包括台式计算机、Macintosh（麦金托什机器）®或其他非 Windows（视窗）计算机、与 Windows 相容的设备、Unix 变体、笔记本计算机等）。所述设备可包括视频游戏控制台、VCR、DVD 播放器、音频环绕音效处理器 / 放大器、卫星机顶盒、电缆 TV 机顶盒等。此外，对于热衷的消费者，可能由于相容性问题而存在多个类似功能的设备。举例而言，消费者可能拥有 HD-DVD 与蓝光 (Blu-ray)DVD 播放器两者，或 Microsoft Xbox（微软家用游戏机）®与 Sony Playstation（索尼游戏站）®视频游戏系统两者。实际上，由于一些跨游戏控制台版本的的游戏的不相容性，消费者可能拥有 XBox 与稍后的版本（诸如，Xbox 360®）两者。经常地，消费者对于使用哪个视频输入端及哪个远端感到迷惑。甚至在将光碟置放于正确的播放器（例如，DVD、HD-DVD、蓝光、Xbox 或 Playstation）中、选择用于该设备的视频及音频输入端且发现正确远程控制之后，消费者仍面临技术挑战。举例而言，在宽屏 DVD 的状况下，用户可能需要首先确定正确的纵横比（例如，4 ： 3、完全、放大、宽放大、电影院宽等）且接着在其 TV 或监视器屏幕上设定正确的纵横比。类似地，用户可能需要首先确定正确的音频环绕音效系统格式（例如，AC-3、杜比数字、DTS 等）且接着设定正确的音频环绕音效系统格式。时常，消费者未意识到其可能未享受到其电视或音频系统的全部能力下的媒体内容（例如，观看以错误纵横比挤压的电影，或收听立体声的音频而非环绕音效的音频）。

[0008] 日益增加地，已将以基于互联网的媒体设备添加到设备的堆栈中。类似 Sonos（索罗斯）®数字音乐系统的音频设备使音频直接从互联网流动 (stream)。同样地，类似 Slingbox™（视灵宝 ™）娱乐播放器的设备记录视频且使其经由家庭网络流动或经由互联网流动而出，其中可在 PC 上远程观看该视频。且互联网协议电视 (IPTV) 服务经由数字用户线 (DSL) 或其他家庭互联网连接而提供类似电缆 TV 的服务。近来还努力将多个媒体功能整合到单个设备（诸如，Moxi（摩西）®媒体中心及执行 Windows XP 媒体中心版本的 PC) 中。尽管所述设备中的每个设备对其执行的功能提供一点便利，但每个设备缺乏对大多数媒体

的普遍且简单的存取。另外，常常由于昂贵的处理和 / 或本地储存的需要而使得所述设备经常花费数百美元来制造。另外，所述现代的消费者电子设备通常消耗大量电力，甚至当闲置时也消耗大量电力，这意谓着其随着时间而更加昂贵且浪费能源。举例而言，若消费者忘记将设备切断或将其切换到不同视频输入端，则该设备可能继续操作。此外，因为所述设备当中没有一个设备为完全的解决方案，所以必须将其与家庭中的其他设备的堆栈整合在一起，这仍对用户留下鼠窝式线及许多远程控制。

[0009] 此外，当许多较新的以互联网为基础的设备适当地工作时，其通常提供更一般形式（与其原本可能可用的形式相比）的媒体。举例而言，使视频经由互联网流动的设备常常仅使视频材料流动，而不能使常常伴随 DVD 的互动式“额外项目”流动，如视频的“制作”、游戏或导演评论。这是由于以下事实：互动式材料经常是以特定格式制作，该特定格式意欲用于在本地处理互动性的特定设备。举例而言，DVD、HD-DVD 及蓝光光碟中每一者具有其自身的特定互动格式。任何家庭媒体设备或本地计算机（其可能经开发以支持所有流行格式）将需要一定程度的尖端性 (sophistication) 及灵活性，其将可能对于消费者操作而言过于昂贵及复杂。

[0010] 使该问题加重，若稍后在将来引入新格式，则本地设备可能不具有支持新格式的硬件能力，这将意味着消费者将必须购买升级的本地媒体设备。举例而言，若在稍后的日期引入较高分辨率的视频或立体视频（例如，每一只眼一个视频流），则本地设备可能不具有解码该视频的计算能力，或其可能不具有用于以新格式输出该视频的硬件（例如，假定通过与遮光眼镜 (shuttered glassess) 同步的 120fps 视频来实现立体视觉，其中将 60fps 传送到每一只眼，若消费者的视频硬件仅可支持 60fps 视频，则该选项在缺乏升级的硬件购买的情况下将不可用）。

[0011] 当谈及尖端的互动式媒体（尤其是视频游戏）时，媒体设备废弃及复杂度的问题为严重问题。

[0012] 现代视频游戏应用基本上划分成四个主要非便携式式硬件平台：Sony PlayStation® 1、2 及 3(PS1、PS2，及 PS3)；Microsoft Xbox® 及 Xbox360®；及 Nintendo Gamecube（任天堂方糖）®及 Wii™；以及以 PC 为基础的游戏。所述平台中的每一者不同于其他者，使得被编写以在一个平台上执行的游戏通常不会在另一平台上执行。也可能存在一代设备与下一代设备的相容性问题。即使大多数软件游戏开发者建立独立于特定平台而设计的软件游戏，为了在特定平台上执行特定游戏，也需要专有软件层（其经常被称为“游戏开发引擎」”）来调适游戏以在特定平台上使用。每一个平台以“控制台”（也即，附接到 TV 或监视器 / 扬声器的脱机盒 (standalone box)）的形式出售给消费者或其本身为 PC。通常，视频游戏在诸如蓝光 DVD、DVD-ROM 或 CD-ROM 的光学媒体上出售，该光学媒体含有体现为尖端的实时软件应用程序的视频游戏。随着家庭宽带速度增加，视频游戏正日益变得可用于下载。

[0013] 由于高级视频游戏的实时性及高计算要求而使得实现与视频游戏软件的平台相容性的特殊性要求极端苛刻。举例而言，一个人可能期望从一代视频游戏到下一代视频游戏（例如，自 XBox 至 XBox 360，或自 Playstation 2（“PS2”）至 Playstation 3（“PS3”））的完全游戏相容性，正如存在从一个 PC 到具有较快处理单元或核心的另一 PC 的生产力应用程序（例如，Microsoft Word（微软文字处理软件））的普遍相容性。然而，对于视频游

戏并非是这种状况。因为当发行一代视频游戏时，视频游戏制造商通常寻求对于给定价格点的最高可能性能，所以经常对系统进行动态架构改变，以使得被编写以用于前代系统的许多游戏在稍后一代系统上不能工作。举例而言，XBox 基于 x86 系列处理器，而 XBox 360 基于 PowerPC 系列。

[0014] 可利用技术来模仿先前架构，但假定视频游戏为实时应用程序，则在模仿中达成完全相同的行为常常不切实际。这是对消费者、视频游戏控制台制造商及视频游戏软件出版商的损失。对于消费者而言，这意味着将旧的一代视频游戏控制台与新的一代视频游戏控制台两者保持接通到 TV 以便能够玩所有游戏的必要性。对于控制台制造商而言，这意味着与新控制台的模仿及较慢采用相关的成本。且对于出版商而言，这意味着可能必须发行新游戏的多个版本以便涵盖所有潜在的消费者 - 不仅发行用于视频游戏的每个商标（例如，XBox、Playstation）的版本，而且常常发行用于给定商标的每个版本（例如，PS2 及 PS3）的版本。举例而言，开发艺电有限公司 (Electronic Arts) 的“疯狂橄榄球 08”的单独版本以用于 XBox、XBox 360、PS2、PS3、Gamecube、Wii 及 PC 平台以及其他平台。

[0015] 便携式设备（诸如，移动电话及便携式媒体播放器）也对游戏开发商提出挑战。日益增加地，所述设备连接到无线数据网络且能够下载视频游戏。但是，市场中存在具有多种不同显示分辨率及计算能力的多种移动电话及媒体设备。而且，因为所述设备通常具有电力消耗、成本及重量约束，所以其通常缺乏类似于图形处理单元（“GPU”）的高级图形加速硬件（诸如，由美国加州的圣克拉拉的 NVIDIA（英伟达公司）制造的设备）。因此，游戏软件开发商通常开发同时用于许多不同类型的便携式设备的给定游戏标题。用户可发现 :给定游戏标题不可用于其特定移动电话或便携式媒体播放器。

[0016] 在家庭游戏控制台的状况下，硬件平台制造商通常向软件游戏开发商收取用于在其平台上发布游戏的能力的版税。移动电话无线通信公司通常也向游戏出版商收取用于将游戏下载到移动电话中的版税。在 PC 游戏的状况下，不存在用于发布游戏所支付的版税，但由于用于支持多种 PC 配置及可能引起的安装问题的较高消费者服务负担而使得游戏开发商通常面临高成本。而且，PC 通常较少阻碍游戏软件的盗版，因为其可很容易地由精通技术的用户重新编程且游戏可更容易地被盗版且更容易地被分配（例如，经由互联网）。因此，对于软件游戏开发商而言，在游戏控制台、移动电话及 PC 上发行具有成本及不利之处。

[0017] 对于控制台及 PC 软件的游戏出版商而言，成本不止于此。为了经由零售通道分配游戏，出版商向零售商收取低于出售价格的批发价格以使零售商具有利润率。出版商通常也必须支付制造及分配保存游戏的物理媒体的成本。零售商经常也向出版商收取“价格保护费”以涵盖可能的意外费用（诸如，游戏售不出，或游戏的价格降低，或零售商必须退还部分或所有批发价格和 / 或从购买者收回游戏）。另外，零售商通常也向出版商收取用于帮助在广告传单中销售游戏的费用。此外，零售商日益增加地从已玩完游戏的用户购买回游戏，且接着将所述游戏以使用过的游戏出售，通常不与游戏出版商分享使用过的游戏的收入。以下事实增加了施加给游戏出版商的成本负担 :游戏常常被经由互联网盗版及分配以供用户下载及进行免费复制。

[0018] 随着互联网宽带速度增加且宽带连接性在美国及全世界变得更广泛（更具体地，到家庭和到租赁连接互联网的 PC 的“网吧”），游戏被更多地经由下载而分配到 PC 或控制台。而且，宽带连接更多地用于玩多人及大型多人在线游戏（该两者在本公开中由首字母

缩写词“MMOG”来指代）。这些改变减轻了与零售分配相关的一些成本及问题。下载在线游戏解决了游戏出版商的一些不利之处，因为分配成本通常较小且存在较少或不存在未出售媒体的成本。但已下载的游戏仍被盗版，且由于其大小（大小常常为几十亿字节）而使得其可能花费非常长的时间来下载。另外，多个游戏可装满小磁盘驱动器，例如连同便携式计算机一起或连同视频游戏控制台一起出售的那些磁盘驱动器。然而，就游戏或 MMOG 需要在线连接以使得游戏可玩的程度而言，盗版问题得以减轻，因为通常需要用户具有有效的用户帐户。不同于可由相机拍摄显示屏幕的视频或由麦克风记录来自扬声器的音频来复制的线性媒体（例如，视频及音乐），每个视频游戏体验是唯一的，且不可使用简单的视频 / 音频记录来复制。因此，甚至在未强力执行版权法且盗版猖獗的区域中，也可保护 MMOG 免于被盗版，从而可支持商业。举例而言，已成功地部署 Vivendi SA（维旺迪公司）的“魔兽世界”MMOG，而在全世界未遭受盗版。且许多在线或 MMOG 游戏（诸如，Linden Lab（林登实验室）的“第二人生”MMOG）通过建在游戏中的经济模型而产生游戏运营商的收入，其中资产可使用在线工具而带来、出售且甚至建立。因此，可使用除传统游戏软件购买或订阅之外的机制来为在线游戏的使用付费。

[0019] 尽管由于在线或 MMOG 的性质而使得常常可减轻盗版，但在线游戏运营商仍面临其余挑战。许多游戏需要大量的本地（也即，家庭内）处理资源以供在线或 MMOG 适当地工作。若用户具有低性能的本地计算机（例如，不具有 GPU 的计算机，诸如低端笔记本计算机），则其可能不能够玩该游戏。另外，随着游戏控制台老化，其远落后于目前技术状态且可能不能够处理更高级的游戏。即使假定用户的本地 PC 能够处理游戏的计算要求，常常也存在安装复杂度。可能存在驱动器不相容性（例如，若下载新游戏，则可能安装新版本的图形驱动器，其致使依赖于旧版本图形驱动器的先前已安装的游戏不可操作）。随着下载更多游戏，控制台可能用完本地磁盘空间。当发现缺陷并修复时或若对游戏进行了修改（例如，若游戏开发商发现游戏的级别太难玩或太容易玩），复杂游戏通常随着时间推移而从游戏开发商接收下载的补丁 (patch)。该补丁需要新的下载。但有时并非所有用户完成所有补丁的下载。在其他时候，下载的补丁引入其他相容性或磁盘空间消耗问题。

[0020] 而且，在游戏播放期间，可能需要大数据下载以将图形或行为信息提供到本地 PC 或控制台。举例而言，若用户进入 MMOG 中的一个房间中，且遇到由图形数据组成或具有在用户的本地机器上不可用的行为的场景或人物，则必须下载那个场景或人物的数据。若互联网连接不够快，则此可导致玩游戏期间的实质延迟。此外，若所遇到的场景或人物需要超过本地 PC 或控制台的储存空间或计算能力的储存空间或计算能力，则其可产生下述情形：其中用户不能在游戏中继续，或必须以质量降低的图形继续。因此，在线或 MMOG 游戏常常限制其储存和 / 或计算复杂度要求。另外，其常常限制游戏期间的数据传送的量。在线或 MMOG 游戏也可使可玩游戏的用户的市场变窄。

[0021] 此外，精通技术的用户越来越多地对游戏的本地复本进行反向工程且修改游戏以使得他们可以作弊。作弊可能与进行比用人力可能的速度快的重复按钮按压（例如，为了非常快速地射击）一样简单。在支持游戏中资产交易的游戏中，作弊可达到导致欺骗性交易涉及具有实际经济价值的资产的欺诈程度。当在线或 MMOG 经济模型基于所述资产交易时，这可导致对游戏运营商的实质有害后果。

[0022] 开发新游戏的成本随着 PC 及控制台能够制作越来越尖端的游戏（例如，具有更逼

真的图形（诸如，实时光线追踪），及更逼真的行为（诸如，实时物理学仿真））而增长。在视频游戏业的早期，视频游戏开发是应用程序软件开发非常类似的过程；也即，大多数开发成本在软件的开发中（与图形、音频及行为要素或“资产”的开发相对比），诸如可被开发以用于具有广泛特殊效果的电影的那些软件开发。现今，许多尖端的视频游戏开发成果比软件开发更类似于富有特效的电影开发。举例而言，许多视频游戏提供 3-D 世界的仿真，且产生更加真实（也即，看似与摄影拍摄的实景 (live action) 图像一样逼真的计算机图形）的人物、道具及环境。照片一样逼真的游戏开发的最具挑战方面中的一者为创建不能区别于实景人脸的计算机产生的人脸。面部捕获技术（诸如，由加州的圣弗朗西斯科的 Mova 开发的 Contour™（轮廓 ™）真实性捕获系统）捕获表演者的面部的精确几何形状并在表演者处于运动中时以高分辨率追踪表演者的面部的精确几何形状。此技术允许在 PC 或游戏控制台上再现 (render) 3D 面部，该 3D 面部实际上不能区别于所捕获的实景面部。精确地捕获及再现“照片一样逼真的”人脸在多个方面是有用的。首先，高度可识别的名人或运动员常常用于视频游戏中（常常被高成本雇用），且不完美性可能对于用户而言显而易见，从而使观看体验 (viewing experience) 分心或令人不愉快。经常地，需要高度细节来实现高度的照片一样的逼真感 - 潜在地需要再现大量多边形及高分辨率纹理（在多边形和 / 或纹理在帧接帧的基础上随着面部移动而改变的情况下）。

[0023] 当具有详细纹理的高多边形计数场景快速改变时，支持游戏的 PC 或游戏控制台可能不具有足够的 RAM 来储存用于游戏片段中所产生的所需数目的动画帧的足够多边形及纹理数据。另外，通常可用于 PC 或游戏控制台上的单个光学驱动器或单个磁盘驱动器通常比 RAM 缓慢得多，且通常不能跟上 GPU 在再现多边形及纹理中可接受的最大数据速率。当前游戏通常将大多数多边形及纹理载入到 RAM 中，这意味着给定场景在复杂度及持续时间上很大程度上受 RAM 的容量限制。在例如面部动画制作的状况下，这可能将 PC 或游戏控制台限制于并无真实感的低分辨率面部，或限制于仅可在游戏暂停且载入用于更多帧的多边形及纹理（及其他数据）之前在有限数目的帧中制作成动画的真实感面部。

[0024] 当 PC 或控制台显示类似于“正在载入 ...”的消息时，观看进程条跨屏幕缓慢地移动被现今的复杂视频游戏的用户公认为是内在缺点。下一个场景从磁盘（除非另外有条件，否则本文中的“磁盘”指非易失性光学媒体或磁性媒体，以及诸如半导体“闪存”存储器的非磁盘媒体）载入时的延迟可花费若干秒或甚至若干分钟。这浪费时间且可能使游戏玩家相当沮丧。如先前所述，大量或所有延迟可能是由于来自磁盘的多边形、纹理或其他数据的载入时间，但也可能是以下状况：当 PC 或控制台中的处理器和 / 或 GPU 准备用于场景的数据时，花费一部分载入时间。举例而言，英式足球视频游戏可允许玩家在大量玩家、小组、运动场及天气条件当中选择。因此，取决于选择什么特定组合，可能需要用于场景的不同多边形、纹理及其他数据（统称“对象”）（例如，不同小组在其制服上具有不同色彩及图案）。可能要列举各种排列中的许多或所有排列且提前预先计算对象中的许多或所有对象并将对象储存在用于储存游戏的磁盘上。但是，若排列的数目很大，则所有对象所需的储存量可能过大以致不能安装在磁盘上（或太不切实际以致不能下载）。因此，现有的 PC 及控制台系统通常在给定场景的复杂度与播放持续时间两者上受约束且对于复杂场景遭受长的载入时间。

[0025] 先前技术的视频游戏系统及应用程序软件系统的另一显著限制在于：其越来越多

地使用例如 3D 对象的大数据库（诸如，多边形及纹理），所述大数据库需要被载入到 PC 或游戏控制台中以用于处理。如上所述，当将所述数据库在本地储存于磁盘上时，所述数据库可花费长时间来载入。然而，若数据库系储存于远程位置且经由互联网来存取，则载入时间通常严重得多。在此种情形下，下载大数据库可能花费几分钟、几小时或甚至几天。另外，所述数据库常常产生大量费用（例如，用于游戏、电影或历史记录片中的详细的高的有桅帆船的 3D 模型）且意欲用于销售给本地终端用户。然而，一旦数据库被下载至本地用户，其就有被盗版的风险。在许多状况下，用户仅为了评估数据库来观看其是否适合用户的需要（例如，当用户执行特定移动时，用于游戏人物的 3D 服装是否具有满意的外观或外表）的目的而希望下载数据库。对于在决定进行购买之前评估 3D 数据库的用户而言，长载入时间可能是阻碍。

[0026] 类似问题在 MMOG（更具体地，如允许用户利用更加定制化的人物的游戏）中出现。对于显示人物的 PC 或游戏控制台，其需要能够存取具有 3D 几何形状（多边形、纹理等）以及所述人物的行为（例如，若人物具有盾牌，则盾牌是否足够强以使矛偏转）的数据库。通常，当 MMOG 由用户初次玩时，用于人物的大量数据库在游戏的初始复本下已经可用，游戏的初始复本在本地在游戏光盘上可用或被下载到磁盘。但是，随着游戏进展，若用户遇到数据库在本地不可用的人物或对象（例如，若另一用户已建立一定制人物），则在可显示该人物或对象之前，必须下载其数据库。这可导致游戏的实质延迟。

[0027] 给定视频游戏的尖端性及复杂度，则在先前技术视频游戏控制台情况下对视频游戏开发商及出版商的另一挑战在于：开发视频游戏经常花费 2 年到 3 年，成本在数千万美元。假定新视频游戏控制台平台以大致每隔五年一次的速率引入，则游戏开发商需要在新游戏控制台发行之前的数年开始那些游戏的开发工作，以便在发行新平台时使视频游戏同时可用。来自竞争性制造商的若干个控制台有时大约同时发行（例如，彼此在一年或两年内），但尚待分晓的是每个控制台的流行性（例如，哪个控制台将产生最大的视频游戏软件销售）。举例而言，在最近的控制台周期中，Microsoft XBox360、Sony Playstation 3 及 Nintendo Wii 计划在大约相同的大体时段引进。但在所述引进之前的数年中，游戏开发商实质上必须“压注 (place bets)”哪些控制台平台将比其他者更成功，且相应地投入其开发资源。电影制作公司也必须在电影发行之前很长时间基于其估计可能成功的电影而分摊其有限的制作资源。给定视频游戏所需的投资的增长程度，则游戏制作越加变得类似电影制作，且游戏制作公司常规上基于其对特定视频游戏的将来成功的估计而投入其制作资源。但是，不同于电影公司，此压注并非仅基于制作本身的成功；而是，其依据于游戏要在其上执行的游戏控制台的成功。同时在多个控制台上发行游戏可减轻风险，但此额外努力增加成本，且经常延迟游戏的实际发行。

[0028] PC 上的应用程序软件及用户环境正变得更为计算上密集、动态及互动，不仅使其在视觉上更吸引用户，而且使其更有用及直观。举例而言，新 Windows Vista（视窗远景）™ 操作系统与 Macintosh® 操作系统的后续版本两者并入了视觉动画效应。高级图形工具（诸如，来自 Autodesk（欧特克）公司的 Maya™（玛雅 ™））提供非常尖端的 3D 再现及动画制作能力（其推动了目前技术状态的 CPU 及 GPU 的限制）。然而，这些新工具的计算要求对于所述产品的用户及软件开发商而言产生了许多实际问题。

[0029] 因为操作系统 (OS) 的视觉显示必须在多种计算机（包括不再出售但仍可随着新

OS 而升级的前代计算机）上工作，OS 图形要求在很大程度上受 OS 要用于的计算机（其通常包括不包括 GPU 的计算机）的最少共同点限制。这严重地限制 OS 的图形能力。此外，电池供电的便携式计算机（例如，笔记本计算机）限制视觉显示能力，因为 CPU 或 GPU 中的高计算活动通常导致较高电力消耗及较短电池寿命。便携式计算机通常包括在不利用处理器时自动地减低处理器活动性以降低电力消耗的软件。在一些计算机型号中，用户可手动地减低处理器活动性。举例而言，Sony 的 VGN-SZ280P 笔记本计算机包括在一侧上标记为“Stamina（持久性）”（用于低性能，更长电池寿命）且另一侧上标记为“Speed（速度）”（用于高性能，较短电池寿命）的交换器。在便携式计算机上执行的 OS 必须能够即使在计算机以其峰值性能能力的一小部分执行的情况下也可用地起作用。因此，OS 图形性能常常保持为远低于目前技术状态的可用计算能力。

[0030] 经常出售高端的计算上密集的应用程序（如 Maya），期望所述应用程序将用于高性能 PC 上。此通常产生高得多的性能，及更昂贵且便携性较差、最少共同点的要求。因此，所述应用程序具有比通用 OS（或通用生产力应用程序，类似 Microsoft Office）有限得多的目标受众且通常以比通用 OS 软件或通用应用程序软件低得多的量出售。潜在的受众进一步受限制，因为预期的用户时常难以提前试用所述计算上密集的应用程序。举例而言，假设学生希望了解如何使用 Maya 或已经知道所述应用程序的潜在购买者在购买中希望在进行投资之前试用 Maya（此可能涉及也购买能够执行 Maya 的高端计算机）。当学生或潜在购买者可下载 Maya 的演示版本或得到 Maya 演示版本的物理媒体复本时，若其缺乏能够执行 Maya 的全部潜能（例如，处理复杂 3D 场景）的计算机，则其将不能够进行产品的全方位评估。此实质上限制所述高端应用程序的受众。这也使出售价格变高，因为开发成本通常经过比通用应用程序的购买次数小得多的购买次数而分摊。

[0031] 高价应用程序也对使用应用程序软件的盗版复本的个体及商业产生更多刺激。因此，高端应用程序软件遭受猖獗盗版，尽管该软件的出版商进行了大量努力来通过各种技术减轻该盗版。但是，甚至当使用盗版的高端应用程序时，用户也不可能排除投资昂贵的目前技术状态的 PC 来执行盗版复本的需要。因此，尽管用户可以用软件应用程序的实际零售价格的一小部分获得软件应用程序的使用，但盗版软件的用户仍需要购买或获得昂贵的 PC，以便完全利用该应用程序。

[0032] 此对于高性能盗版视频游戏的用户同样成立。尽管盗版者可以用游戏的实际价格的一小部分得到游戏，但其仍需要购买适当地玩游戏所需的昂贵计算硬件（例如，GPU- 增强型 PC，或类似 XBox 360 的高端视频游戏控制台）。假定视频游戏通常是消费者的娱乐，则用于高端视频游戏系统的额外成本可能是过于昂贵的。该情形在当前工人的平均年收入相当低（相对于美国的当前工人平均年收入）的国家（例如，中国）中更糟。这样，小得多的百分比的人口拥有高端视频游戏系统或高端 PC。在这些国家中，用户可支付费用以使用连接到互联网的计算机的“网吧”相当普遍。经常地，所述网吧具有不具有高性能特征（诸如，原本可使玩家能够玩计算上密集的视频游戏的 GPU）的较旧型号或低端 PC。这是在低端 PC 上执行的游戏成功的关键因素（诸如，Vivendi 的“魔兽世界”，其在中国高度成功，且通常是在中国的网吧中玩）。相比之下，计算上密集的游戏（如“第二人生”）更不可能在安装于中国网吧中的 PC 上玩。所述游戏实际上对于仅能够存取网吧中的低性能 PC 的用户来说是不可访问的。

[0033] 对于考虑购买视频游戏且首先愿意通过经由互联网将演示下载到其家庭而试用游戏的示范版本的用户也存在障碍。视频游戏演示常常为游戏的全能版本，其中一些特征停用，或对游戏播放的量施加限制。此可能涉及在可将游戏安装于PC或控制台上且在PC或控制台上执行之前下载数十亿字节的数据的长过程（可能几个小时）。在PC的状况下，其也可能涉及算出游戏需要哪些特殊驱动器（例如，DirectX或OpenGL驱动器），下载正确的版本，安装正确的版本，及接着确定PC是否能够播放该游戏。后者步骤可能涉及确定PC是否具有足够的处理（CPU及GPU）能力、足够的RAM及相容的OS（例如，一些游戏在Windows XP上执行而不在Vista上执行）。因此，在试图执行视频游戏演示的长过程之后，用户可能发现在给定用户的PC配置的情况下视频游戏演示不可能玩。更糟地，一旦用户已下载新驱动器以用于尝试该演示，这些驱动器版本就可能与用户在PC上习惯使用的其他游戏或应用程序不相容，因此，演示的安装可致使先前可操作的游戏或应用程序不能操作。这些障碍不仅使用户沮丧，而且其也对视频游戏软件出版商及视频游戏开发商销售其游戏产生障碍。

[0034] 导致不具经济效益的另一问题与以下事实有关：给定PC或游戏控制台通常被设计以适应对应用程序和/或游戏的特定程度的性能要求。举例而言，一些PC具有或多或少的RAM、较慢或较快的CPU及较慢或较快的GPU（若其具有GPU）。一些游戏或应用程序利用给定PC或控制台的全计算能力，而一些游戏或应用程序却不利用给定PC或控制台的全计算能力。若用户的游戏或应用程序的选择未达到本地PC或控制台的峰值性能能力，则用户可能由于未利用的特征而在PC或控制台上浪费了财力。在控制台的状况下，控制台制造商可能支付比资助控制台成本所要的多的成本。

[0035] 存在于视频游戏的销售及享受中的另一问题涉及在用户实施购买游戏之前允许用户观看他人玩游戏。存在用于记录视频游戏以在稍后时间重放的若干先前技术方法。举例而言，美国专利第5,558,339号教导了在“游戏播放”期间将游戏状态信息（包括游戏控制器动作）记录在视频游戏客户端计算机（由同一个或不同用户拥有）中。此状态信息可在稍后时间使用以在视频游戏客户端计算机（例如，PC或控制台）上重放一些或所有游戏动作。该方法的显著缺点在于：对于观看已记录的游戏的用户，用户必须具有能够播放该游戏的视频游戏客户端计算机且必须具有在该计算机上执行的视频游戏应用程序，以使得当重放被记录的游戏状态时游戏播放是完全相同的。除此之外，视频游戏应用程序必须是以在被记录的游戏与经回放的游戏之间不存在可能的执行差异的方式编写。

[0036] 举例而言，游戏图形大体在帧接帧基础上计算。对于许多游戏，取决于场景是否特别复杂或是否存在减缓执行的其他延迟（例如，在PC上，另一过程可能正在执行，以致从游戏应用程序夺走CPU周期），游戏逻辑有时可能花费比一帧时间短或比一帧时间长的时间来计算为下一个帧而显示的图形。在此种游戏中，以比一帧时间稍少的时间（例如，少几个CPU时钟周期）计算的“临限值”帧最终可出现。当使用完全相同的游戏状态信息再次计算该同一场景时，可能容易花费比一帧时间多几个CPU时钟周期的时间（例如，若内部CPU总线稍微与外部DRAM总线不同相，且即使不存在来自从游戏处理夺走数毫秒CPU时间的另一过程的大延迟，其也引入几个CPU周期时间的延迟）。因此，当回放游戏时，帧变成以两个帧时间计算而非以单个帧时间计算。一些行为基于游戏计算新帧的频率（例如，当游戏取样来自游戏控制器的输入时）。当播放游戏时，用于不同行为的时间参考中的该偏差不会

影响游戏播放，但其可导致所回放的游戏产生不同结果。举例而言，若篮球的轨道是以稳定的 60fps 速率来计算，但游戏控制器输入是基于计算的帧的速率来取样，则当记录游戏时，计算的帧的速率可能为 53fps，而当重放游戏时，计算的帧的速率可能为 52fps，此可使得篮球是否被阻止进入篮中存在差异，从而导致不同结果。因此，使用游戏状态记录视频游戏需要非常谨慎的游戏软件设计，以确保使用同一游戏状态信息重放产生完全相同的结果。

[0037] 用于记录视频游戏的另一先前技术方法是仅记录 PC 或视频游戏系统的视频输出（例如，到 VCR、DVD 记录器，或到 PC 上的视频捕获板）。接着可将视频回倒及重放，或替代地，将记录的视频上传到互联网（通常在将视频压缩之后）。该方法的不利之处在于：当回放 3D 游戏序列时，用户限于仅从观看点（序列从该观看点被记录）来观看序列。换言之，用户不可改变场景的观看点。

[0038] 另外，当经由互联网而使在家庭 PC 或游戏控制台上播放的记录的游戏序列的经压缩的视频为其他用户可用时，即使视频是实时压缩，也不可能实时地将经压缩的视频上传到互联网。其原因是因为世界上连接到互联网的许多家庭具有高度不对称的宽带连接（例如，DSL 及电缆调制解调器通常具有比上流带宽高得多的下流带宽）。被压缩的高分辨率视频序列常常具有比网络的上传带宽容量高的带宽，使得其不可能实时上传。因此，在播放游戏序列之后（可能几分钟或甚至几小时），在互联网上的另一用户能够观看该游戏之前，将存在显著延迟。尽管该延迟在特定情形下（例如，观看在先前时间出现的游戏玩家的成果）可容忍，但其消除了观看游戏现场直播（例如，由优胜玩家玩的篮球锦标赛）的能力或现场直播地播放游戏时的“即刻重放”能力。

[0039] 另一先前技术方法允许具有电视接收器的观看者观看视频游戏现场直播，但仅在电视制作人员的控制下。美国与其他国家中的一些电视频道提供视频游戏观看频道，其中电视观众能够在视频游戏频道上观看特定的视频游戏用户（例如，参加锦标赛烦人顶级玩家）。这通过将视频游戏系统 (PC 和 / 或控制台）的视频输出馈送至用于电视频道的视频分配及处理设备中来完成。这正如电视频道广播现场直播的篮球比赛时的情况，其中若干个相机从篮球场周围的不同角度提供现场直播的馈送。电视频道接着能够利用其视频 / 音频处理及效应设备来操作来自各种视频游戏系统的输出。举例而言，电视频道可在来自视频游戏的视频之上叠加指示不同玩家的状态的文字（正如其可在现场直播的篮球比赛期间叠加文字），且电视频道可加录来自评论员（其可论述在比赛期间出现的动作）的音频。另外，可将视频游戏输出与记录游戏的实际玩家的视频的相机（例如，显示玩家对游戏的情绪反应）组合。

[0040] 该方法的一个问题在于：必须实时地使所述现场直播的视频馈送为电视频道的视频分配及处理设备可用，以便使其具有现场直播的广播的刺激性。然而，如先前所述，当视频游戏系统从家庭执行时（尤其是当广播的一部分包括来自正捕获游戏玩家的真实世界视频的相机的现场直播的视频时），这常常不可能。另外，在锦标赛情形下，所关注的是家庭中游戏者可修改游戏及作弊，如先前所述。由于这些原因，电视频道上的所述视频游戏广播常常配置有聚集于公共位置处（例如，在电视演播室处或在竞技场中）的播放器及视频游戏系统，其中电视制作设备可接受来自多个视频游戏系统及潜在的现场直播的相机的视频馈送。

[0041] 尽管所述先前技术视频游戏电视频道可为电视观众提供非常刺激的演出（这是

与现场直播的运动事件同类（例如，与以“运动员”呈现的视频游戏玩家同类）的体验，不仅根据其在视频游戏世界中的动作，而且根据其在真实世界中的动作），但这些视频游戏系统常常限于玩家彼此身体上极接近的情形。此外，因为电视频道被广播，所以每个被广播的频道仅可显示由电视频道的制作人员选择的一个视频流。由于这些限制及广播时间、制作设备及制作人员的高成本，所述电视频道通常仅显示参加顶级锦标赛的顶级玩家。

[0042] 另外，向全部电视观众广播视频游戏的全屏幕图像的给定电视频道每次仅显示一个视频游戏。这严重地限制电视观看者的选择。举例而言，电视观看者可能对给定时间显示的游戏不感兴趣。另一观看者可能仅对观看并非由电视频道在给定时间放映的特定玩家的游戏播放感兴趣。在其他状况下，观看者可能仅对观看内行玩家如何处理游戏中的特定级别感兴趣。其他观看者可能希望控制观看点（视频游戏从该观看点来看），该观看点不同于由制作小组等选择的观看点。简言之，电视观看者在观看视频游戏中可能具有无数的偏好（即使若干个不同电视频道可用，电视网络的特定广播也不适应所述偏好）。由于所有上述原因，使得先前技术视频游戏电视频道在向电视观看者呈现视频游戏中具有显著限制。

[0043] 先前技术视频游戏系统及应用程序软件系统的另一缺点在于：他们很复杂，且通常遭受错误、崩溃和 / 或无意识且不需要的行为（统称“缺陷”）。尽管游戏及应用程序在发行之前通常经历除错及调谐过程（经常称为“软件质量保证”或 SQA），但几乎不变的是：一旦游戏或应用程序被发行到领域中的广大受众，缺陷就会突然出现。遗憾的是，软件开发商难以在发行之后识别及追踪到许多缺陷。软件开发商可能难以意识到缺陷。即使当其了解缺陷时，也可能仅存在其可用于识别是什么引起该缺陷的有限量的信息。举例而言，用户可打电话给游戏开发商的消费者服务热线且留下消息，该消息陈述：当玩游戏时，屏幕开始闪烁，接着变成固体蓝 (solid blue) 且 PC 冻结。其为 SQA 小组提供了在追踪缺陷中有用的非常少的信息。在线连接的一些游戏或应用程序在特定状况下有时可提供更多信息。举例而言，有时可使用”看门狗”过程来监视游戏或应用程序是否“崩溃”。看门狗过程可收集游戏或应用程序崩溃时关于游戏或应用程序过程的状态（例如，存储器堆栈使用状态、游戏或应用程序进展到的程度等）的统计，且接着经由互联网而将所述信息上传至 SQA 小组。但在复杂游戏或应用程序中，该信息可花费非常长的时间来解密，以便准确地确定在崩溃时用户正在进行什么。尽管如此，也不可能确定什么事件序列导致崩溃。

[0044] 与 PC 及游戏控制台相关联的又一问题在于：其经受使消费者极不便利的服务问题。服务问题也影响 PC 或游戏控制台的制造商，因为其通常需要发送特殊盒子以安全地装运破损的 PC 或控制台，且因而招致修理的成本（若 PC 或控制台处于保修期内）。游戏或应用程序软件出版商也可受处于修理状态中的 PC 和 / 或控制台引起的销售损失（或在线服务使用）影响。

[0045] 图 1 说明诸如 Sony Playstation®3、Microsoft Xbox 360®、Nintendo Wii™、以 Windows 为基础的个人计算机或 Apple Macintosh 的先前技术视频游戏系统。所述系统中的每一者包括用于执行程序码的中央处理单元 (CPU)（通常为用于执行高级图形操作的图形处理单元 (GPU)），及用于与外部设备及用户通信的多个形式的输入 / 输出 (I/O)。为简单起见，将所述组件显示为组合在一起为单个单元 100。图 1 的先前技术视频游戏系统也显示为包括光学媒体驱动器 104（例如，DVD-ROM 驱动器）；用于储存视频游戏程序代码及数据的硬盘驱动器 103；用于播放多人游戏、用于下载游戏、补丁、演示或其他媒体的网络连接

105；用于储存当前正由CPU/GPU 100执行的程序码的随机存取存储器(RAM)101；用于在游戏播放期间接收来自用户的输入命令的游戏控制器106；及显示设备102(例如，SDTV/HDTV或计算机监视器)。

[0046] 图1中所显示的先前技术系统受到若干限制。首先，与RAM101的存取速度相比较，光学驱动器104及硬碟机103往往具有慢得多的存取速度。当直接通过RAM 101工作时，由于RAM 101通常具有高得多的带宽且不会受到磁盘机构相对长的搜寻延迟的事实，CPU/GPU 100在实践中可处理比直接从硬盘驱动器103或光学驱动器104读出程序代码及数据时可能的每秒多边形数多得多的每秒多边形数。但仅有限量的RAM提供于这些先前技术系统中(例如，256-512兆字节)。因此，常常需要“正在载入...”序列，其中RAM 101被周期性地填充有用于视频游戏的下一个场景的数据。

[0047] 一些系统试图同时地重迭程序代码的载入与游戏播放，但这仅可在存在已知序列的事件时进行(例如，若正沿道路驾驶车，则可在驾驶车的同时载入路旁的正接近的建筑物的几何形状)。对于复杂和/或快速场景改变，此类型的重迭通常不起作用。举例而言，在用户处于战役进行之中且在那时刻的视图内RAM 101完全被填满表示对象的数据的状况下，若用户将视图快速地向左移动以观看当前未载入在RAM 101中的对象，则将导致动作的不连续性，因为不存在足够的时间来将新对象自硬盘驱动器103或光学媒体104载入到RAM 101中。

[0048] 图1的系统的另一问题是由于硬盘驱动器103及光学媒体104的储存容量的限制引起。尽管磁盘储存设备可被制造成有相对较大的储存容量(例如，500亿字节或500亿字节以上)，但其仍不提供用于在当前视频游戏中所遇到的特定情况的足够储存容量。举例而言，如先前所述，英式足球视频游戏可允许用户在全世界的许多小组、玩家及运动场当中选择。对于每个小组、每个玩家及每个运动场，需要大量纹理映射及环境映射来特征化世界上的3D表面(例如，每个小组具有唯一运动衫，每一者需要唯一纹理映射)。

[0049] 用于解决上述后者问题的一个技术是：对于游戏，一旦用户选择了纹理及环境映射，就预先计算纹理及环境映射。此可涉及许多计算上密集的过程，包括解压缩图像、3D映射、加阴影、组织数据结构等。因此，当视频游戏执行这些计算时，对于用户可能存在延迟。减少此延迟的一个方法原则上为：最初开发游戏时执行所有这些计算-包括小组、玩家名册及运动场的每个排列。游戏的发行版本因而将包括储存在光学媒体104上或互联网上的一个或多个服务器上的所有所述经预先处理的数据，当用户作出选择时，仅经由互联网将用于给定小组、玩家名册、运动场选择的选定的预先处理的数据下载到硬盘驱动器103。然而，作为实际问题，游戏播放中可能的每个排列的该预先载入的数据可能轻易地为几兆兆字节(terabyte)的数据，其远超过现今的光学媒体设备的容量。此外，用于给定小组、玩家名册、运动场选择的数据可能轻易地为几亿字节的数据或几亿字节以上的数据。在家庭网络连接的情况下(例如，10Mbps)，经由网络连接105下载该数据将比在本地计算数据花费更长时间。

[0050] 因此，图1中所显示的先前技术游戏架构使用户在复杂游戏的较大场景转变之间经受显著延迟。

[0051] 诸如图1中所显示的先前技术方法的先前技术方法的另一问题在于：这些年来，视频游戏倾向于变得更高级且需要更多CPU/GPU处理能力。因此，即使采用无限量的RAM，

视频游戏硬件要求也超过所述系统中可用的处理能力的峰值水平。因此，需要用户每隔几年升级游戏硬件以保持同步（或以较低质量水准玩较新游戏）。比以往更高级的视频游戏的趋势的后果为：用于家庭用途的玩视频游戏的机器通常不具经济效益，因为其成本通常由其可支持的最高性能游戏的要求来确定。举例而言，可能使用 XBox360 来玩类似“战争机器 (Gears of War)”的游戏，该游戏要求高性能的 CPU、GPU 及几亿字节的 RAM，或者可能使用 XBox 360 来玩“吃豆 (Pac Man)”，其为来自 20 世纪 70 年代的游戏，其仅需要几千字节的 RAM 及非常低性能的 CPU。实际上，XBox 360 具有同时主机代管许多同时的“吃豆”游戏的足够计算能力。

[0052] 在一周的大多数小时中，通常关闭视频游戏机。根据 2006 年 7 月 Nielsen（尼尔森）娱乐对 13 岁及 13 岁以上的活跃游戏者的研究，平均起来，活跃游戏者一周中花费十四个小时或一周中的全部小时的仅 12％来玩控制台视频游戏。这意味着平均视频游戏控制台在 88％的时间内闲置，这是昂贵资源的无效率使用。假定视频游戏控制台常常是由制造商来资助以降低购买价格（期望该资助将通过来自未来视频游戏软件购买的版税来赚回），则这特别有意义。

[0053] 视频游戏控制台也造成与几乎任何消费者电子设备相关的成本。举例而言，需要将系统的电子设备及机构容纳于外壳中。制造商需要提供服务保证。出售该系统的零售商需要收取关于系统的销售和 / 或关于视频游戏软件的销售的利润。所有这些因素添加视频游戏控制台的成本，该成本必须由制造商来资助、传递至消费者，或者由制造商与消费者两者来资助。

[0054] 另外，盗版是视频游戏工业的较大问题。实际上每个较大视频游戏系统上所利用的安全机构这些年来已“破裂”，导致视频游戏的未经授权的复制。举例而言，Xbox 360 安全系统在 2006 年 7 月破裂且用户现在能够在线下载非法复本。可下载的游戏（例如，用于 PC 或 Mac 的游戏）特别容易经受盗版。在世界的特定区域（其中盗版管制不强）中，实质上不存在独立视频游戏软件的可行市场，因为用户可与合法复本一般容易地以成本的非常小一部分购买盗版复本。而且，在世界的许多地方，游戏控制台的成本占收入的高百分比，以致即使盗版受控制，也很少有人可买得起目前技术状态的游戏系统。

[0055] 另外，已使用的游戏的市场减少了视频游戏业的收入。当用户变得对游戏厌倦时，其可将游戏出售给将游戏转售给其他用户的店铺。这种未经授权但普遍的实践显著减少了游戏出版商的收入。类似地，当每隔几年存在平台转变时，通常出现大约 50％的销售减少。这是因为：当用户知道即将发行较新版本的平台时，用户停止购买用于较旧平台的游戏（例如，当即将发行 Playstation 3 时，用户停止购买 Playstation 2 游戏）。组合起来，销售的损失及与新平台相关的增加的开发成本可对游戏开发商的收益性有非常显著的不利影响。

[0056] 新游戏控制台也非常昂贵。Xbox 360、Nintendo Wii 及 SonyPlaystation 3 均以数百美元零售。高能力的个人计算机游戏系统可花费高达 $8000。这表示用户的显著投资，具体来说，考虑到硬件在几年后变陈旧及许多系统是为孩子而购买的事实。

[0057] 上述问题的一个方法是在线游戏，其中将游戏程序代码及数据主机代管于服务器上且按要求将其传送至客户端机器，经压缩的视频及音频经由数字宽带网络而流动。一些公司（诸如，芬兰的 G-Cluster(G- 群集公司），其现在为日本的 SOFTBANK Broadmedia（软

银宽媒）的子公司）当前在线提供所述服务。类似游戏服务变得在本地网络（诸如，旅馆内及由 DSL 及电缆电视提供者提供的那些网络）中可用。这些系统的较大缺点是延时的问题，也即，信号行进到游戏服务器及从游戏服务器行进所花费的时间，游戏服务器通常定位在运营商的“前端”中。快速动作视频游戏（也称为“极速（twitch）”视频游戏）在用户通过游戏控制器执行动作的时间与更新显示屏幕以显示用户动作的结果的时间之间需要非常低的延时。需要低延时，以使得用户感觉到游戏“即刻地”响应。可视游戏的类型及用户的熟练程度而以不同延时间隔来满足用户。举例而言，对于缓慢的非正式游戏（类似西洋双陆棋）或慢动作角色扮演游戏而言，100 毫秒的延时可能是可容忍的，但在快动作游戏中，超过 70 毫秒或 80 毫秒的延时可引起用户在游戏中更拙劣地表现，且因此不可接受。举例而言，在需要快反应时间的游戏中，当延时自 50 毫秒增加至 100 毫秒时，存在准确度的锐降。

[0058] 当游戏或应用服务器安装在附近的受控网络环境或至用户的网络路径可预测和 / 或可容忍带宽峰值的网络环境中时，在最大延时以及延时的一致性方面，控制延时容易得多（例如，因此用户经由网络观察到来自数字视频流动的稳定运动）。该程度的控制可在以下达成：在电缆 TV 网络前端到电缆 TV 用户的家庭之间，或自 DSL 中央办公室至 DSL 用户的家庭，或在来自服务器或用户的商业办公室区域网络 (LAN) 环境中。而且，有可能获得商业之间的具有得到保证的带宽及延时的特定分级的点到点私用连接。但在将游戏主机代管于连接到通用互联网的服务器中心中且接着经由宽带连接而使经压缩的视频流动 (stream) 到用户的游戏或应用系统中，许多因素造成延时，导致先前技术系统的部署中的严重限制。

[0059] 在典型的连接宽带的家庭中，用户可具有用于宽带服务的 DSL 或电缆调制解调器。所述宽带服务通常造成用户的家庭与通用互联网之间的多达 25 毫秒的来回行程延时（且有时更多）。另外，存在由于经由互联网将数据路由到服务器中心而造成的来回行程延时。经由互联网的延时基于给出数据的路线及数据被路由时数据所造成的延迟而改变。除路由延迟之外，还由于光穿过使大多数互联网互连的光纤的速度而造成来回行程延时。举例而言，对于每 1000 英里，由于光穿过光纤的速度及其他耗用而造成约 22 毫秒的来回行程延时。

[0060] 额外延时可由于经由互联网流动的数据的数据速率而造成。举例而言，若用户具有以“6Mbps DSL 服务”出售的 DSL 服务，则在实践中，用户将很可能最多得到小于 5Mbps 的下行输送量，且将可能周期性地看见由于各种因素（诸如，峰值载入时间期间在数字用户线接入复用器 (DSLAM) 处的拥挤）产生的连接降级。若经由相邻者循环的本地共用同轴电缆中存在拥挤或电缆调制解调器系统网络中的其他地方存在拥挤，则类似问题可出现，从而将用于以“6Mbps 电缆调制解调器服务”出售的连接的电缆调制解调器的数据速率减小至远小于该数据速率。若使 4Mbps 的稳定速率下的数据分组以用户数据报协议 (UDP) 格式单向地从服务器中心经由所述连接而流动，若一切都适当地工作，则数据分组将通过而不造成额外延时，但若存在拥挤（或其他妨碍）且仅 3.5Mbps 可用于使数据流动到用户，则在典型情形下，包将被丢弃，导致丢失数据，或者分组将在拥挤点处排队直至它们可被发送为止，从而引入了额外延时。不同拥挤点具有用于保存被延迟的分组的不同队列容量，因此在一些状况下，立即将不可成功解决拥挤的分组丢弃。在其他状况下，将几百万比特的数据排队且最终将其发送。但是，在几乎所有状况下，拥挤点处的排队具有容量限制，且一旦超

过该限制，队列将溢出且分组将被丢弃。因此，为了避免造成额外延时（或更糟地，分组丢失），必须避免超过从游戏或应用服务器到用户的数据速率容量。

[0061] 还由于在服务器中压缩视频及在客户端设备中解压缩视频所需的时间而造成延时。当在服务器上执行的视频游戏正在计算待显示的下一个帧时，进一步造成延时。当前可用的视频压缩算法受到高数据速率或高延时。举例而言，运动 JPEG 为仅帧内有损的压缩算法，该压缩算法特征为低延时。视频的每个帧独立于视频的每个其他帧而压缩。当客户端设备接收经压缩的运动 JPEG 视频的一个帧时，其可立即解压缩该帧且显示该帧，从而导致非常低的延时。但因为每个帧分开进行压缩，所以算法不能够利用连续帧之间的类似性，且因此仅帧内视频压缩算法受到非常高的数据速率。举例而言，60fps（每秒帧数）640×480 运动 JPEG 视频可能需要 40Mbps（每秒百万比特）或 40Mbps（每秒百万比特）以上的数据。用于所述低分辨率视频窗的所述高数据速率在许多宽带应用程序中将是过于昂贵的（且对于大多数消费者的基于互联网的应用程序的确如此）。另外，因为每个帧经独立压缩，所以可能由于有损压缩而产生的帧中的假影可能出现于连续帧中的不同位置处。这可导致当解压缩视频时，在观看者看来为移动的视觉假影。

[0062] 其他压缩算法（诸如，来自 Microsoft 公司的 MPEG2、H.264 或 VC9）当用于先前技术的配置中时，可实现高压缩比率，但以高延时为代价。所述算法利用帧间压缩以及帧内压缩。周期性地，所述算法执行帧的仅帧内压缩。这种帧被称为关键帧（通常称作“I”帧）。接着，该算法通常将 I 帧与先前帧与相继帧两者相比较。并非独立地压缩先前帧及相继帧，而是算法确定图像从 I 帧到先前帧及相继帧有什么改变，且接着将该改变储存为：“B”帧（对于 I 帧之前的改变）及“P”帧（对于 I 帧之后的改变）。这导致比仅帧内压缩低得多的数据速率。但是，其通常以较高延时为代价。I 帧通常比 B 帧或 P 帧大得多（常常大 10 倍），且因此，以给定数据速率传输成比例地花费较长的时间。

[0063] 考虑（例如）一个 i 情形：其中 I 帧为 B 帧及 P 帧的大小的 10 倍，且对于每个单个 I 帧内，存在 29 个 B 帧 +30 个 P 帧＝ 59 个帧间，或对于每个“帧群”(GOP) 总共 60 个帧。因此，在 60fps 下，每秒存在 1 个 60 帧 GOP。假设传输信道具有 2Mbps 的最大数据速率。为了在信道中达成最高质量的视频，压缩算法将产生 2Mbps 数据流，且给定上述比率，这将产生每帧内 2 百万比特 (Mb)/(59+10) ＝ 30,394 个比特及每 I 帧 303,935 个比特。当通过解压缩算法接收经压缩的视频流时，为了稳定地播放视频，需要以规则间隔（例如，60fps）解压缩及显示每个帧。为了实现该结果，若任何帧受到传输延时，则需要将所有帧延迟至少该延时，因此最糟状况的帧延时将限定用于每个视频帧的延时。因为 I 帧最大，所以 I 帧引入最长传输延时，且整个 I 帧将必须在可解压缩及显示 I 帧（或取决于 I 帧的任何帧间）之前接收。假定信道数据速率为 2Mbps，则传输 I 帧将花费 303,935/2Mb ＝ 145 毫秒。

[0064] 使用传输信道的带宽的大百分比的帧间视频压缩系统（如上所述）将由于 I 帧相对于帧的平均大小的大的大小而经受长延时。或者，换言之，当先前技术帧间压缩算法达成比仅帧内压缩算法低的平均每帧数据速率（例如，2Mbps 对 40Mbps）时，其由于大 I 帧而仍遭受高的峰值每帧数据速率（例如，303,935*60 ＝ 18.2Mbps）。但请记住：上述分析假定 P 帧及 B 帧均比 I 帧小得多。尽管这大体成立，但对于具有与先前帧、高运动或场景改变不相关的高图像复杂度的帧，这不成立。在所述情形下，P 帧或 B 帧可变得与 I 帧一般大（若 P 帧或 B 帧变得比 I 帧大，则尖端压缩算法通常将“强制”I 帧且用 I 帧替换 P 帧或 B 帧）。因

此，I 帧大小的数据速率峰值可在任何时刻出现于数字视频流中。因此，对于经压缩的视频，当平均视频数据速率接近传输信道的数据速率容量时（经常为该状况，给定对于视频的高数据速率要求），来自 I 帧或大的 P 帧或 B 帧的高峰值数据速率导致高帧延时。

[0065] 当然，上述论述仅特征化由 GOP 中的大的 B 帧、P 帧或 I 帧产生的压缩算法延时。若使用 B 帧，则延时将更高。原因是因为在可显示 B 帧之前，必须接收 B 帧之后的所有 B 帧及 I 帧。因此，在诸如 BBBBBIPPPPPBBBBBIPPPPP 的图片群 (GOP) 序列中，其中在每个 I 帧之前存在 5 个 B 帧，只有在接收到随后的 B 帧及 I 帧之后才可由视频解压缩器显示第一 B 帧。因此，若使视频以 60fps（也即，16.67 毫秒 / 帧）流动，则在可解压缩第一 B 帧之前，不管信道带宽如何快，接收五个 B 帧及 I 帧将花费 16.67*6 = 100 毫秒，且这是仅 5 个 B 帧的情况。具有 30 个 B 帧的经压缩的视频序列相当普遍。此外，在如 2Mbps 的低信道带宽下，由于 I 帧的大小而引起的延时影响很大程度上增加到由于等待 B 帧到达而产生的延时影响。因此，在 2Mbps 信道上，在大量 B 帧的情况下，使用先前技术视频压缩技术超过 500 毫秒或 500 毫秒以上的延时相当容易。若不使用 B 帧（对于给定质量水准，以较低压缩比率为代价），则不招致 B 帧延时，但仍招致上文所描述的由于峰值帧大小而引起的延时。

[0066] 问题恰恰由于许多视频游戏的性质而加重。利用上文所描述的 GOP 结构的视频压缩算法很大程度上被最佳化以用于连同要用于被动观看的现场直播的视频或电影材料一起使用。通常，相机（真实相机，或者计算机产生的动画的状况下的虚拟相机）及场景相对稳定，仅因为若相机或场景太颠簸地来回移动，则视频或电影材料 (a) 通常观看起来令人不愉快，且 (b) 若其正被观看，当相机突然来回颠簸时，观看者通常不能够紧密地跟随该动作（例如，若相机在拍摄吹灭生日蛋糕上的蜡烛的孩子时被扰动且突然在蛋糕之间来回颠簸，则观看者通常集中于孩子及蛋糕上，而不理会相机突然移动时的简短中断）。在视频会谈或视频电话会议的状况下，可将相机固持于固定位置中且根本不移动，从而导致根本非常少的数据峰值。但 3D 高动作视频游戏通过恒定运动来被特征化（例如，考虑 3D 竞赛，其中整个帧在竞赛的持续时间中处于快速运动中，或者考虑第一人称射击游戏，其中虚拟相机恒定地颠簸地来回移动）。所述视频游戏可产生具有大的及频繁的峰值的帧序列，其中用户可能需要清楚地看见在该突然运动期间发生了什么。因此，在 3D 高动作视频游戏中，压缩假影远不可容忍。因此，许多视频游戏的视频输出（由于其性质）产生具有非常高且频繁的峰值的经压缩的视频流。

[0067] 假定快动作视频游戏的用户对于高延时具有小的容忍度，且给定所有上述延时原因，至今存在对于使视频在互联网上流动的服务器主机代管的视频游戏的限制。另外，若需要高度互动性的应用程序被主机代管于通用互联网上且使视频流动，则所述应用程序的用户遭受类似限制。所述服务需要网络配置，其中主机代管服务器直接设置于前端（在电缆宽带的状况下）或中央办公室（在数字用户线 (DSL) 的状况下）中，或商业背景中的 LAN 内（或特别分级的私用连接上），以便控制自客户端设备至服务器的路线及距离以最小化延时且可适应峰值而不造成延时。LAN（通常额定在 100Mbps-1Gbps）及具有足够带宽的租用线路通常可支持峰值带宽要求（例如，18Mbps 峰值带宽为 100Mbps LAN 容量的一小部分）。

[0068] 若进行特殊适应，则峰值带宽要求也可由住宅宽带基础架构来适应。举例而言，在电缆 TV 系统上，可为数字视频通信给出专用带宽，该专用带宽可处理诸如大 I 帧的峰值。此外，在 DSL 系统上，可供应较高速度的 DSL 调制解调器（考虑高峰值），或可供应可处理较高

数据速率的特别分级的连接。但是，附接至通用互联网的传统电缆调制解调器及 DSL 基础架构对于用于压缩的视频的峰值带宽要求而言远不能容忍。因此，在线服务（将视频游戏或应用程序主机代管于距客户端设备长距离的服务器中心中，且接着经由传统的住宅宽带连接经由互联网而使经压缩的视频输出流动）遭受显著的延时及峰值带宽要求－尤其对于需要非常低的延时的游戏及应用程序（例如，第一人称射击游戏及其他多用户、互动式动作游戏，或需要快响应时间的应用程序）。

附图说明

[0069] 根据下面的详细描述并根据附图可以更完整地理解本公开，然而，所述附图并不用来将公开的主题限制到所示特定实施方式中，而只是用作解释和理解。

[0070] 图 1 示出了先前技术视频游戏系统的架构。

[0071] 图 2a 至图 2b 示出了根据一个实施例的高级系统架构。

[0072] 图 3 示出了用于客户端与服务器之间的通信的实际的、额定的及所需的数据速率。

[0073] 图 4a 示出了根据一个实施例而使用的主机服务及客户端。

[0074] 图 4b 示出了与客户端与主机服务之间的通信相关的例示性延时。

[0075] 图 4c 示出了根据一个实施例的客户端设备。

[0076] 图 4d 示出了根据另一个实施例的客户端设备。

[0077] 图 4e 示出了图 4c 中的客户端设备的实例方块图。

[0078] 图 4f 示出了图 4d 中的客户端设备的实例方块图。

[0079] 图 5 示出了可根据一个实施例而使用的视频压缩的一个实例。

[0080] 图 6a 示出了可在另一个实施例中使用的视频压缩的一个实例。

[0081] 图 6b 示出了与传输低复杂度、低动作视频序列相关的数据速率中的峰值。

[0082] 图 6c 示出了与传输高复杂度、高动作视频序列相关的数据速率中的峰值。

[0083] 图 7a 至图 7b 示出了在一个实施例中使用的实例视频压缩技术。

[0084] 图 8 示出了在一个实施例中使用的额外实例视频压缩技术。

[0085] 图 9a 至图 9c 示出了在本发明一个实施例中使用的帧速率处理技术。

[0086] 图 10a 至图 10b 示出了将图像图像块有效地封装于分组内的一个实施例。

[0087] 图 11a 至图 11d 示出了使用前向纠错技术的实施例。

[0088] 图 12 示出了使用多核心处理单元来进行压缩的一个实施例。

[0089] 图 13a 至图 13b 示出了根据各种实施例的主机服务之间的地理定位及通信。

[0090] 图 14 示出了与客户端与主机服务之间的通信相关的示例性延时。

[0091] 图 15 示出了实例主机服务服务器中心架构。

[0092] 图 16 示出了包括多个现场直播的视频窗的用户接口的一个实施例的实例屏幕拍摄。

[0093] 图 17 示出了在选择特定视频窗之后的图 16 的用户接口。

[0094] 图 18 示出了在将特定视频窗放大至全屏幕大小之后的图 17 的用户接口。

[0095] 图 19 示出了叠加在多人游戏的屏幕上的实例合作用户视频数据。

[0096] 图 20 示出了用于主机服务上的游戏玩家的实例用户页面。

[0097] 图 21 示出了实例 3D 互动式广告。

[0098] 图 22 示出了用于自现场直播的表演的表面捕获产生具有纹理表面的照片般逼真的图像的实例步骤序列。

[0099] 图 23 示出了允许选择线性媒体内容的实例用户接口页面。

[0100] 图 24 为示出了在现场直播网页之前消逝的时间量与连接速度的曲线图。

[0101] 图 25a-b 示出了本发明的采用从客户端设备至主机服务的反馈信道的实施方式。

[0102] 图 26a-b 示出了一实施例，其中基于最后已知的已成功接收的图像块 / 帧来对图像块 / 帧进行编码。

[0103] 图 27a-b 示出了一实施例，其中游戏或应用程序的状态被从第一主机服务或服务器运送 (ported) 至第二主机服务或服务器。

[0104] 图 28 示出了一实施例，其中通过使用差异数据来运送游戏或应用程序的状态。

[0105] 图 29 示出了本发明一实施例，该实施例在客户端设备上采用临时解码器。

[0106] 图 30 示出了根据本发明一实施例的如何在“R 帧”之间散布“I 图像块”。

[0107] 图 31a-h 示出了本发明的生成直播流和 / 或一个或多个 HQ 流的实施例。

具体实施方式

[0108] 在以下描述中列举了特定细节（诸如，设备类型、系统配置、通信方法等），以便提供对本公开的彻底理解。然而，一般本领域的技术人员应了解，实践所描述的所述实施例可能不需要这些特定细节。

[0109] 图 2a 至图 2b 提供两个实施例的高级架构，其中视频游戏及软件应用程序由主机服务 210 主机代管且在订阅服务下由用户场所 211（注意，“用户场所”意思是用户所定位的无论何处的位置，若使用移动设备则包括室外）处的客户端设备 205 经由互联网 206（或其他公众网络或私用网络）来存取。客户端设备 205 可为具有至互联网的有线或无线连接、具有内部或外部显示设备 222 的通用计算机（诸如，以 Microsoft Windows 或 Linux 为基础的 PC 或 Apple 公司的 Macintosh 计算机），或者其可为将视频及音频输出到监视器或电视机 222 的诸如机顶盒的专用客户端设备（具有至互联网的有线或无线连接），或者其可为推测起来具有至互联网的无线连接的行动设备。

[0110] 所述设备中的任一者可具有其自身的用户输入设备（例如，键盘、按钮、触摸屏幕、追踪板 (track pad) 或惯性感测棒 (inertial-sensing wand)、视频捕获相机和 / 或运动追踪相机等），或者其可使用通过有线或无线连接的外部输入设备 221（例如，键盘、鼠标、游戏控制器、惯性感测棒、视频捕获相机和 / 或运动追踪相机等）。如下文更详细描述，主机服务 210 包括各种性能水平的服务器（包括具有高能力 CPU/GPU 处理能力的那些服务器）。在播放游戏或使用主机服务 210 上的应用程序期间，家庭或办公室客户端设备 205 接收来自用户的键盘和 / 或控制器输入，且接着其将控制器输入经由互联网 206 传输至主机服务 210，主机服务 210 作为响应来执行游戏程序代码并产生用于游戏或应用程序软件的视频输出（视频图像序列）的相继帧（例如，若用户按压将会指引屏幕上的人物向右移动的按钮，则游戏程序接着将产生显示人物向右移动的视频图像序列）。接着使用低延时视频压缩器压缩该视频图像序列，且主机服务 210 接着经由互联网 206 而传输低延时视频流。家庭或办公室客户端设备接着解码经压缩的视频流并将经解压缩的视频图像再现于监

视器或 TV 上。因此，显著地减少客户端设备 205 的计算及图形硬件要求。客户端 205 仅需要具有用于将键盘 / 控制器输入转发到互联网 206 且解码并解压缩从互联网 206 所接收的经压缩的视频流的处理能力，实际上现今任何个人计算机均能够在其 CPU 上以软件来进行这些（例如，以约 2GHz 执行的 Intel 公司双核 CPU 能够解压缩使用诸如 H.264 及 Windows 媒体 VC9 的压缩器编码的 720p HDTV）。此外，在任何客户端设备的状况下，专用芯片也可以比通用 CPU 低得多的成本及比通用 CPU 少得多的电力消耗（诸如，现代 PC 所需的）来实时地执行用于所述标准的视频解压缩。值得注意地，为了执行转递控制器输入及解压缩视频的功能，家庭客户端设备 205 不需要任何专门化的图形处理单元 (GPU)、光学驱动器或硬盘驱动器（诸如，图 1 中所显示的先前技术视频游戏系统）。

[0111] 随着游戏及应用程序软件变得更复杂及更具照片般逼真感，其将需要较高性能的 CPU、GPU、较多 RAM，及较大且较快的磁盘驱动器，且可使主机服务 210 处的计算能力不断地升级，但终端用户将不需要使家庭或办公室客户端平台 205 升级，因为将通过给定视频解压缩算法而使家庭或办公室客户端平台 205 的处理要求对于显示分辨率及帧速率保持恒定。因此，图 2a 至图 2b 中所说明的系统中不存在现今所见的硬件限制及相容性问题。

[0112] 另外，因为游戏及应用程序软件仅在主机服务 210 中的服务器中执行，所以在用户的家庭或办公室（除非另外有条件，否则如本文中所使用的“办公室”将包括任何非住宅背景，包括（例如）教室）中决不存在游戏或应用程序软件的复本（光学媒体的形式，或者为下载的软件）。这显著减轻游戏或应用程序软件被非法复制（盗版）的可能性，以及减轻可由游戏或应用程序软件使用的有价值的数据库被盗版的可能性。实际上，若需要专门化的服务器（例如，需要非常昂贵的、大的或有噪音的设备）来播放对于家庭或办公室使用不可行的游戏或应用程序软件，则即使获得游戏或应用程序软件的盗版复本，其也将不可在家庭或办公室中操作。

[0113] 在一个实施例中，主机服务 210 向设计视频游戏的游戏或应用程序软件开发商（其大体指代软件开发公司、游戏或电影工作室，或游戏或应用程序软件出版商）220 提供软件开发工具，以使得其可设计能够在主机服务 210 上执行的游戏。所述工具允许开发商利用主机服务的特征（所述特征通常在独立 PC 或游戏控制台中将不可用）（例如，快速存取复杂几何形状的非常大的数据库（除非另外有条件，否则“几何形状”将在本文中用于指代限定 3D 数据集的多边形、纹理、索具、照明、行为及其他组件及参数））。

[0114] 在该架构下，不同商业模型是可能的。在一个模型下，主机服务 210 从终端用户收取订阅费用且向开发商 220 支付版税，如图 2a 中所显示。在替代实施中（图 2b 中所显示），开发商 220 直接从用户收取订阅费用且向主机服务 210 支付用于主机代管游戏或应用程序内容的费用。这些基本原理不限于用于提供在线游戏或应用程序主机代管的任何特定商业模型。

[0115] 经压缩的视频特性

[0116] 如先前所论述，在线提供视频游戏服务或应用程序软件服务的一个显著问题在于延时。70 毫秒 -80 毫秒的延时（自输入设备被用户致动 (actuate) 的时刻到在显示设备上显示响应时的时刻）为用于需要快响应时间的游戏及应用程序的上限。然而，由于大量实际及物理约束而使得这在图 2a 及图 2b 中所显示的架构的情况下非常难以达成。

[0117] 如图 3 中所指示，当用户订阅互联网服务时，连接通常额定为到用户的家庭或办

公室的标定最大数据速率301。取决于提供者的策略及路由设备能力，该最大数据速率可或多或少被严格地强制执行，但通常由于许多不同原因中的一者而使得实际可用数据速率较低。举例而言，可能在DSL中央办公室处或在本地电缆调制解调器回路上存在过多网络通信通信，或可能在电缆线上存在噪音，从而引起丢弃的分组，或提供者可能建立每用户每月最大数目的比特。当前，用于电缆及DSL服务的最大下行数据速率通常在数百千比特/秒(Kbps)到30Mbps的范围内。蜂窝式服务通常限于数百Kbps的下行数据。然而，宽带服务的速度及订阅宽带服务的用户的数目将随着时间而急剧增加。当前，一些分析者估计33%的美国宽带用户具有2Mbps或2Mbps以上的下行数据速率。举例而言，一些分析者预测：至2010年止，超过85%的美国宽带用户将具有2Mbps或2Mbps以上的数据速率。

[0118] 如图3中所指示，实际可用最大数据速率302可随着时间而波动。因此，在低延时、在线游戏或应用程序软件情况下，有时难以预测用于特定视频流的实际可用数据速率。若对于特定量的场景复杂度及运动在给定数目的每秒帧数(fps)下以给定分辨率(例如，640×480@60fps)维持给定质量水平所需的数据速率303升高高于实际可用最大数据速率302(如通过图3中的峰值指示)，则可出现若干问题。举例而言，一些互联网服务将仅丢弃分组，从而导致用户的视频屏幕上的丢失的数据及失真的/丢失的图像。其他服务将暂时缓冲(也即，排队)额外分组且以可用数据速率将所述分组提供到客户端，从而导致延时的增加-对于许多视频游戏及应用程序而言为不可接受的结果。最后，一些互联网服务提供者将数据速率的增加视为恶意攻击(诸如，否认服务攻击(由计算机黑客用以使网络连接停用的公知技术))，且将在特定时间周期中切断用户的互联网连接。因此，本文中所描述的实施例设法确保用于视频游戏的所需数据速率不会超过最大可用数据速率。

[0119] 主机服务架构

[0120] 图4a说明根据一个实施例的主机服务210的架构。主机服务210可位于单个服务器中心中，或者可跨越多个服务器中心而分散(以为具有比其他者低延时的至特定服务器中心的路径的用户提供低延时连接，以在用户之间提供负载平衡，且在一或多个服务器中心出故障的状况下提供冗余)。主机服务210最终可包括成千上万个或甚至数百万个服务器402，从而服务非常大的用户基础(user base)。主机服务控制系统401提供对主机服务210的总体控制，且指引路由器、服务器、视频压缩系统、计费及帐务系统等。在一个实施例中，主机服务控制系统401实施在基于Linux的分散式处理系统上，该处理系统绑定到用于储存用于用户信息、服务器信息及系统统计数据的数据库的RAID阵列。在上述描述中，除非归因于其他特定系统，否则由主机服务210实施的各种动作由主机服务控制系统401来起始及控制。

[0121] 主机服务210包括许多服务器402，诸如当前可从Intel(因特尔公司)、IBM(美国国际商用机器公司)及Hewlett Packard(惠普公司)及其他者得到的所述服务器。或者，可将服务器402装配成定制组件配置，或者最终可将服务器402整合以便将整个服务器实施为单个芯片。尽管此图为说明起见而显示少数服务器402，但在实际部署中，可能存在少至一个服务器402或多达数百万个或数百万个以上服务器402的服务器。服务器402均可以相同方式配置(作为一些配置参数的实例，具有相同CPU类型及性能；具有或不具有GPU，且若具有GPU，则具有相同GPU类型及性能；具有相同数目的CPU及GPU；具有相同量及相同类型/速度的RAM；及具有相同RAM配置)，或服务器402的各种子集可具有相同配置

（例如，25％的服务器可以一个特定方式配置，50％的服务器以不同方式配置，且 25％的服务器以又一个方式配置），或每个服务器 402 可不同。

[0122] 在一个实施例中，服务器 402 无磁盘，也即，并非具有其自身的本地大容量储存器（其为光学或磁性储存器，或者基于半导体的储存器，诸如闪存或服务类似功能的其他大容量储存装置），每一个服务器经由快速底板或网络连接而存取共用的大容量储存器。在一个实施例中，该快速连接为连接到独立冗余磁盘阵列 (RAID)405 系列的储存区域网络 (SAN)403，在使用超高速以太网实施的设备之间具有连接。如本领域的技术人员已知的，SAN 403 可用于将许多 RAID 阵列 405 组合在一起，从而导致极高的带宽 - 接近或可能超过可自用于当前游戏控制台及 PC 中的 RAM 得到的带宽。此外，尽管基于诸如磁性媒体的旋转媒体的 RAID 阵列经常具有显著的搜寻时间存取延时，但基于半导体储存器的 RAID 阵列可实施为具有低得多的存取延时。在另一配置中，一些或所有服务器 402 在本地提供一些或所有其自身的大容量储存器。举例而言，服务器 402 可将频繁存取的信息（诸如，其操作系统及视频游戏或应用程序的复本）储存在基于低延时本地闪存的储存器上，但其可利用 SAN 来存取基于旋转媒体的具有较高搜寻延时的 RAID 阵列 405，以较不频繁地存取几何形状或游戏状态信息的大数据库。

[0123] 另外，在一个实施例中，主机服务 210 使用下文详细描述的低延时视频压缩逻辑 404。视频压缩逻辑 404 可以软件、硬件或其任何组合来实施（下文描述其特定实施例）。视频压缩逻辑 404 包括用于压缩音频以及视觉材料的逻辑。

[0124] 在操作中，当经由键盘、鼠标、游戏控制器或其他输入设备 421 而玩视频游戏或使用用户场所 211 处的应用程序时，客户端 415 上的控制信号逻辑 413 将表示由用户促使的按钮按压（及其他类型的用户输入）的控制信号 406a-b（通常为 UDP 分组的形式）传输到主机服务 210。将来自给定用户的控制信号路由到适当服务器（或若多个服务器响应于用户的输入设备，则路由至多个服务器）402。如图 4a 中所说明，可经由 SAN 而将控制信号 406a 路由至服务器 402。可替换地或另外，可经由主机服务网络（例如，基于以太网的区域网络）而将控制信号 406b 直接路由至服务器 402。不管控制信号 406a-b 是如何被传输，该或所述服务器均响应于控制信号 406a-b 而执行游戏或应用程序软件。尽管图 4a 中未说明，但各种网络连接组件（诸如，防火墙和 / 或网关）可处理主机服务 210 的边缘处（例如，主机服务 210 与互联网 410 之间）和 / 或用户场所 211 的边缘处（互联网 410 与家庭或办公室客户端 415 之间）的传入及传出的通信。所执行的游戏或应用程序软件的图形及音频输出（也即，新的视频图像序列）提供至低延时视频压缩逻辑 404，低延时视频压缩逻辑 404 根据低延时视频压缩技术（诸如，本文中所描述的所述技术）而压缩视频图像序列且经由互联网 410（或，如下文所描述，经由绕过通用互联网的最佳高速网络服务）而将经压缩的视频流（通常具有经压缩或未经压缩的音频）传输回至客户端 415。接着，客户端 415 上的低延时视频解压缩逻辑 412 解压缩视频及音频流并再现经解压缩的视频流，且通常在显示设备 422 上播放经解压缩的音频流。或者，可在与显示设备 422 分开的扬声器上播放音频或根本不播放音频。注意，尽管输入设备 421 及显示设备 422 在图 2a 及图 2b 中显示为独立式设备，但其可集成在诸如便携式计算机或行动设备的客户端设备内。

[0125] 家庭或办公室客户端 415（先前在图 2a 及图 2b 中描述为家庭或办公室客户端 205）可为非常低廉且低能力的设备，其具有非常有限的计算或图形性能且可能具有非常

有限的本地大容量储存器或不具有本地大容量储存器。相比之下，耦合至 SAN 403 及多个 RAID 405 的每一个服务器 402 可为格外高性能的计算系统，且实际上，若多个服务器以并列处理配置合作地使用，则几乎不存在对可承受的计算量及图形处理能力的限制。此外，由于低延时视频压缩 404 及低延时视频解压缩 412（由用户感知地），所以将服务器 402 的计算能力提供给用户。当用户按压输入设备 421 上的按钮时，显示器 422 上的图像被响应于按钮按压而更新（在感知上无有意义的延迟），好像游戏或应用程序软件在本地执行。因此，对于为非常低性能的计算机或只是实施低延时视频解压缩及控制信号逻辑 413 的低廉芯片的家庭或办公室客户端 415，自看来在本地可用的远程位置有效地为用户提供任意计算能力。此为用户给出用于玩最高级、处理器密集的（通常为新的）视频游戏及最高性能的应用程序的能力。

[0126] 图 4c 显示非常基础且低廉的家庭或办公室客户端设备 465。该设备为根据图 4a 及图 4b 的家庭或办公室客户端 415 的一个实施例。其大约 2 英寸长。其具有与具有以太网供电 (PoE) 的以太网电缆相接口的以太网插孔 462，该设备从以太网插孔 462 得到其电力及其到互联网的连接性。该设备能够在支持网络地址转换 (NAT) 的网络内执行 NAT。在办公室环境中，许多新的以太网交换器具有 PoE 且将 PoE 直接带到办公室中的以太网插孔。在此种情形下，所需的为从壁式插孔到客户端 465 的以太网电缆。若可用的以太网连接不运输电力（例如，在具有 DSL 或电缆调制解调器但无 PoE 的家庭中），则存在可用的低廉的壁式“砖块 (brick)”（也即，电源），其将接受无电力的以太网电缆且输出具有 PoE 的以太网。

[0127] 客户端 465 含有耦合至蓝牙无线接口的控制信号逻辑 413（图 4a），该蓝牙无线接口与诸如键盘、鼠标、游戏控制器和 / 或麦克风和 / 或耳机的蓝牙输入设备 479 相接口。而且，客户端 465 的一个实施例在与显示设备 468 耦合的情况下能够以 120fps 输出视频，显示设备 468 能够支持 120fps 视频且向一对遮光眼镜 466 发信号（通常经由红外）以对于每个相继帧交替地遮蔽一只眼接着遮蔽另一只眼。用户所感觉的效果为“跳出”显示屏幕的立体 3D 图像。支持该操作的一种该显示设备 468 为 Samsung HL-T5076S。因为用于每一只眼的视频流是单独的，所以在两个独立视频流由主机服务 210 压缩的一个实施例中，帧在时间上交错，且帧在客户端 465 内以两个独立解压缩过程来解压缩。

[0128] 客户端 465 也含有低延时视频解压缩逻辑 412，其解压缩传入的视频及音频且经由 HDMI（高清晰度多媒体接口）、连接器 463 输出，HDMI（高清晰度多媒体接口）、连接器 463 插入到 SDTV（标准清晰度电视）或 HDTV（高清晰度电视）468 中，从而向 TV 提供视频及音频，或插入到支持 HDMI 的监视器 468 中。若用户的监视器 468 不支持 HDMI，则可使用 HDMI 至 DVI（数字视觉接口），但音频将丢失。在 HDMI 标准下，显示能力（例如，所支持的分辨率，帧速率）464 从显示设备 468 表达，且接着经由互联网连接 462 将该信息传回到主机服务 210，因此主机服务 210 可使经压缩的视频以适合于该显示设备的格式流动。

[0129] 图 4d 显示家庭或办公室客户端设备 475，除了客户端设备 475 具有更多外部接口之外，其与图 4c 中所显示的家庭或办公室客户端设备 465 相同。而且，客户端 475 可接受 PoE 来供电，或者其可占用插入墙壁中的外部电源适配器（未图示）。视频相机 477 使用客户端 475 USB 输入将经压缩的视频提供到客户端 475，经压缩的视频由客户端 475 上传到主机服务 210 以用于下文所描述的用途。将利用下文所描述的压缩技术将低延时压缩器创建

到相机 477 中。

[0130] 除具有用于其互联网连接的以太网连接器之外，客户端 475 还具有到互联网的 802.11g 无线接口。两种接口均能够在支持 NAT 的网络内使用 NAT。

[0131] 而且，除具有用于输出视频及音频的 HDMI 连接器之外，客户端 475 还具有双链接 DVI-I 连接器，双链接 DVI-I 连接器包括模拟输出端（且具有将提供 VGA 输出的标准适配器电缆）。其还具有用于复合视频及 S 视频的模拟输出端。

[0132] 对于音频，客户端 475 具有左 / 右模拟立体 RCA 插孔，且对于数字音频输出，其具有 TOSLINK（光纤）输出端。

[0133] 除了到输入设备 479 的蓝牙无线接口之外，其还具有用于接口到输入设备的 USB 插孔。

[0134] 图 4e 显示客户端 465 的内部架构的一个实施例。该图中所显示的所有设备或一些设备可实施在场可程序化逻辑阵列、定制 ASIC 中或若干个离散设备（定制设计或者现成的）中。

[0135] 具有 PoE 的以太网 497 附接到以太网接口 481。电力 499 才具有 PoE 的以太网 497 得到且连接到客户端 465 中的其余设备。总线 480 为用于设备之间的通信的公同总线。

[0136] 执行来自闪存 476 的小客户端控制应用程序的控制 CPU 483（几乎任何小 CPU 都是适当的，诸如具有嵌入式 RAM 的 100MHz 下的 MIPSR4000 系列 CPU）实施用于网络（也即，以太网接口）的协议栈且还与主机服务 210 通信，并配置客户端 465 中的所有设备。其还处理与输入设备 469 的接口并将分组（必要时，连同受前向纠错保护的用户控制器数据一起）发送回至主机服务 210。而且，控制 CPU 483 监视分组通信（例如，分组是丢失还是延迟，以及其到达的时间戳）。将此信息发送回至主机服务 210，以使得其可恒定地监视网络连接且相应地调整其发送的内容。最初在制造时为闪存 476 载入用于控制 CPU 483 的控制程序以及对于特定客户端 465 单元而言唯一的序号。此序号允许主机服务 210 唯一地识别客户端 465 单元。

[0137] 蓝牙接口 484 经由其天线（在客户端 465 内部）无线地通信至输入设备 469。

[0138] 视频解压缩器 486 为经配置以实施本文中所描述的视频解压缩的低延时视频解压缩器。大量视频解压缩设备存在，或者为现成产品，或者作为具有可整合在 FPGA 或定制 ASIC 中的设计的知识产权 (IP)。一个提供用于 H.264 解码器的 IP 的公司为澳大利亚新南威尔士州 (NSW Australia) 的 Ocean Logic of Manly。使用 IP 的优点在于：本文中所使用的压缩技术与压缩标准不相符。一些标准解压缩器足够灵活以经配置以适应本文中的压缩技术，但一些标准解压缩器可能并非如此。但是，在 IP 的情况下，在视需要而重新设计解压缩器中存在完全灵活性。

[0139] 视频解压缩器的输出端耦合至视频输出子系统 487，视频输出子系统 487 将视频耦合至 HDMI 接口 490 的视频输出端。

[0140] 音频解压缩子系统 488 或者使用可用的标准音频解压缩器来实施，或者其可实施为 IP，或者可在可（例如）实施 Vorbis 音频解压缩器（可在 Vorbis.com 上找到）的控制处理器 483 内实施音频解压缩。

[0141] 实施音频解压缩的设备耦合到音频输出子系统 489，音频输出子系统 489 将音频耦合至 HDMI 接口 490 的音频输出端。

[0142] 图 4f 显示客户端 475 的内部架构的一个实施例。如可见，除额外接口及来自插入墙壁中的电源适配器的可选外部 DC 电力（且若如此使用，则可选外部 DC 电力替换将来自以太网 PoE 497 的电力）之外，该架构与客户端 465 的架构相同。下文中将不重复与客户端 465 共同的功能性，但将额外功能性描述如下。

[0143] CPU 483 与额外设备通信且配置额外设备。

[0144] WiFi 子系统 482 经由其天线提供无线互联网存取，作为对以太网 497 的替代。WiFi 子系统可购自多家制造商，包括美国加州的圣克拉拉的 Atheros Communications（阿特赫鲁斯通信公司）。

[0145] USB 子系统 485 提供对用于有线 USB 输入设备 479 的蓝牙通信的替代。USB 子系统相当标准且可容易地用于 FPGA 及 ASIC，且经常创建到执行如视频解压缩的其他功能的现成设备中。

[0146] 与客户端 465 内的视频输出相比较，视频输出子系统 487 产生较宽范围的视频输出。除提供 HDMI 490 视频输出之外，其提供 DVI-I 491、S- 视频 492 及合成视频 493。而且，当 DVI-I 491 接口用于数字视频时，将显示能力 464 自显示设备传回至控制 CPU 483，以使得其可向主机服务 210 通知显示设备 478 的能力。由视频输出子系统 487 提供的所有接口均为相当标准的接口且容易以许多形式可用。

[0147] 音频输出子系统 489 经由数字接口 494(S/PDIF 和 / 或 Toslink) 数字地输出音频且经由立体模拟接口 495 输出模拟形式的音频。

[0148] 来回行程延时分析

[0149] 当然，为了实现前一段的利益，用户使用输入设备 421 的动作与在显示设备 420 上看见该动作的后果之间的来回行程延时应不大于 70 毫秒 -80 毫秒。此延时必须考虑在自用户场所 211 中的输入设备 421 到主机服务 210 及再次返回到用户场所 211 至显示设备 422 的路径中的所有因素。图 4b 说明各种组件及网络（信号必须经由该组件或网络行进），且该组件及网络上方的为时间线，该时间线列出实际实施中可预期的例示性延时。注意，图 4b 经简化以便仅显示重要路径路由。下文描述用于系统的其他特征的数据的其他路由。双头箭头（例如，箭头 453）指示来回行程延时且单头箭头（例如，箭头 457）指示单向延时，且“～”表示近似量测。应指出，将存在所列的延时不可达成的真实世界情形，但在大量状况下，在美国，使用至用户场所 211 的 DSL 及电缆调制解调器连接，该延时可在下一段中所描述的情形中达成。而且，注意，尽管至互联网的蜂窝式无线连接性的确将在所显示的系统中工作，但大多数当前美国蜂窝式数据系统（诸如，EVDO）招致非常高的延时且将不能够达成图 4b 中所显示的延时。然而，这些基本原理可在可能能够实施该水平的延时的未来蜂窝式技术上实施。进一步地，存在这样的游戏及应用程序实例（例如，不要求快速用户反应时间的游戏，诸如象棋），其中通过当前 US 蜂窝数据系统所产生的延时虽然对于用户而言是显而易见的，但对于该游戏及应用程序而言是可接受的。

[0150] 自用户场所 211 处的输入设备 421 开始，一旦用户致动输入设备 421，就将用户控制信号发送至客户端 415（其可为诸如机顶盒的独立设备，或其可为在诸如 PC 或行动设备的另一设备中执行的软件或硬件），且将其分组化（在一个实施例中以 UDP 格式）并为分组给出目的地地址以到达主机服务 210。分组将也含有用于指示控制信号来自哪个用户的信息。接着经由防火墙 / 路由器 /NAT（网络地址转换）设备 443 将控制信号分组转发到 WAN

接口 442。WAN 接口 442 为由用户的 ISP（互联网服务提供者）提供到用户场所 211 的接口设备。WAN 接口 442 可以是电缆或 DSL 数据机、WiMax 收发器、光纤收发器、蜂窝式数据接口、电力线互联网协议接口 (Internet Protocol-over-powerline interface)，或到互联网的许多接口中的任何其他接口。另外，可将防火墙 / 路由器 /NAT 设备 443（及（可能地）WAN 接口 442）整合到客户端 415 中。该一个实例将为移动电话，其包括用于实施家庭或办公室客户端 415 的功能性的软件，以及用于经由某标准（例如，802.11g）而无线地路由及连接到互联网的装置。

[0151] WAN 接口 442 接着将控制信号路由至本文中所称的用于用户的互联网服务提供者 (ISP) 的“存在点”441，WAN 接口 442 为提供连接至用户场所 211 的 WAN 输送器与通用互联网或私用网络之间的接口的设施。存在点的特性将视所提供的互联网服务的性质而改变。对于 DSL，其通常是 DSLAM 位于的电话公司中央办公室。对于电缆调制解调器，其通常是电缆多系统运营商 (MSO) 前端。对于蜂窝式系统，其通常是与蜂窝式塔相关的控制室。但无论存在点的性质怎样，其均将接着将控制信号分组路由至通用互联网 410。接着经由将最可能系光纤收发器接口的接口将控制信号分组路由至 WAN 接口 444 至主机服务 210。WAN 444 将接着将控制信号分组路由至路由逻辑 409（其可以许多不同方式来实施，包括以太网交换器及路由服务器），路由逻辑 409 估计用户的地址且将控制信号路由至用于给定用户的正确的服务器 402。

[0152] 服务器 402 接着将所述控制信号视为在服务器 402 上执行的游戏或应用程序软件的输入且使用所述控制信号来处理游戏或应用程序的下一个帧。一旦产生下一个帧，就将视频及音频自服务器 402 输出至视频压缩器 404。可经由各种装置将视频及音频自服务器 402 输出至压缩器 404。首先，可将压缩器 404 创建到服务器 402 中，因此可在服务器 402 内在本地实施压缩。或者，可经由到网络（其或者是服务器 402 与视频压缩器 404 之间的私用网络，或者是经由诸如 SAN 403 的共用网络的网络）的网络连接（诸如，以太网连接）以分组化的形式输出视频和 / 或音频。或者，可经由视频输出连接器（诸如，DVI 或 VGA 连接器）将视频自服务器 402 输出，且接着由视频压缩器 404 来捕获。而且，可将音频自服务器 402 输出为数字音频（例如，经由 TOSLINK 或 S/PDIF 连接器）或模拟音频，模拟音频由视频压缩器 404 内的音频压缩逻辑来数字化并编码。

[0153] 一旦视频压缩器 404 已捕获来自服务器 402 的视频帧及在该帧时间期间所产生的音频，则视频压缩器将使用下文所描述的技术压缩视频及音频。一旦视频及音频被压缩，就通过一地址将其分组化以将其发送回至用户的客户端 415，且将其路由至 WAN 接口 444，WAN 接口 444 接着经由通用互联网 410 路由视频及音频分组，通用互联网 410 接着将视频及音频分组路由至用户的 ISP 的存在点 441，存在点 441 将视频及音频分组路由至用户场所处的 WAN 接口 442，WAN 接口 442 将视频及音频分组路由至防火墙 / 路由器 /NAT 设备 443，防火墙 / 路由器 /NAT 设备 443 接着将视频及音频分组路由至客户端 415。

[0154] 客户端 415 解压缩视频及音频，且接着在显示设备 422（或客户端的内置显示设备）上显示视频并将音频发送到显示设备 422 或到单独的放大器 / 扬声器或到创建到客户端中的放大器 / 扬声器。

[0155] 为使用户感觉到刚才所描述的整个过程在感知上没有滞后，来回行程延迟需要小于 70 毫秒或 80 毫秒。所描述的来回行程路径中的一些延时延迟受主机服务 210 和 / 或用

户的控制，而其他的延时延迟不受主机服务 210 和 / 或用户的控制。尽管如此，基于大量真实世界情况的分析及测试，以下为近似量测。

[0156] 用于发送控制信号的单向传输时间 451 通常小于 1 毫秒，经由用户场所的来回行程路由 452 通常使用以太网上的易得的消费者级防火墙 / 路由器 /NAT 交换器而在约 1 毫秒内完成。用户 ISP 广泛地改变其来回行程延迟 453，但在 DSL 及电缆调制解调器提供者的情况下，通常看见其在 10 毫秒与 25 毫秒之间。通用互联网 410 上的来回行程延时可视频务被如何路由及路在线是否存在任何故障（且该问题在下文加以论述）而极大地改变，但通常通用互联网提供相当最佳的路由且延时很大程度上由光穿过光纤的速度（给定到目的地的距离）来确定。如下文进一步论述，已确定 1000 英里作为期望将主机服务 210 远离用户场所 211 置放的大致最远距离。在 1000 英里处（来回行程 2000 英里），用于信号经由互联网的实际传输时间为约 22 毫秒。至主机服务 210 的 WAN 接口 444 通常为具有可忽略的延时的商业级光纤高速接口。因此，通用互联网延时 454 通常在 1 毫秒与 10 毫秒之间。经由主机服务 210 的单向路由 455 延时可在小于 1 毫秒内达成。服务器 402 通常将在小于一帧时间（其在 60fps 下为 16.7 毫秒）的时间中计算用于游戏或应用程序的新帧，因此 16 毫秒为将使用的合理的最大单向延时 456。在本文中所描述的视频压缩及音频压缩算法的最佳硬件实施中，压缩 457 可在 1 毫秒内完成。在次佳版本中，压缩可花费多达 6 毫秒（当然，更欠佳的版本可花费更长时间，但所述实施将影响来回行程的总延时且将需要其他延时较短（例如，可减小经由通用互联网的可允许的距离）以维持 70 毫秒 -80 毫秒延时目标）。已经考虑互联网 454、用户 ISP 453 及用户场所路由 452 的来回行程延时，因此剩余的为视频解压缩 458 延时，视频解压缩 458 延时取决于视频解压缩 458 是实施于专用硬件中还是实施于客户端设备 415（诸如，PC 或行动设备）上的软件中，可视显示器的大小及解压缩 CPU 的性能而改变。通常，解压缩 458 花费 1 毫秒与 8 毫秒之间的时间。

[0157] 因此，可通过将在实践中所见的所有最糟状况的延时加在一起来确定图 4a 中所显示的系统的用户可预期将经历的最糟状况的来回行程延时。他们是 :1+1+25+22+1+16+6+8 = 80 毫秒。此外，实际上，在实践中（具有下文所论述的防止误解的说明），此大致为使用图 4a 中所显示的系统的原型版本（在美国使用现成的 Windows PC 作为客户端设备及家庭 DSL 及电缆调制解调器连接）所见的来回行程延时。当然，优于最糟状况的情况可导致短得多的延时，但不可依赖其来开发广泛使用的商业服务。

[0158] 为了经由通用互联网达成图 4b 中所列出的延时，需要客户端 415 中的视频压缩器 404 及视频解压缩器 412（来自图 4a）产生具有非常特定的特性的分组流，以使得经由自主机服务 210 至显示设备 422 的整个路径所产生的分组序列不经受延迟或过多分组丢失，且详言之，始终如一地落在经由用户的互联网连接（经由 WAN 接口 442 及防火墙 / 路由器 / NAT 433）而可用于用户的带宽的约束内。另外，视频压缩器必须产生足够强健的分组流，以使得其可容忍在正常互联网及网络传输中出现的不可避免的分组丢失及分组重排序。

[0159] 低延时视频压缩

[0160] 为了完成上述目标，一个实施例采用新的视频压缩方法，该方法降低用于传输视频的延时及峰值带宽要求。在描述该实施例之前，将关于图 5 及图 6a 至图 6b 提供对当前视频压缩技术的分析。当然，若用户具备足以处理所述技术所需的数据速率的带宽，则该技术可根据基本原理来使用。注意，本文中不解决音频压缩，而是陈述音频压缩与视频压缩同

时且同步地来实施。满足用于该系统的要求的先前技术音频压缩技术存在。

[0161] 图 5 说明用于压缩视频的一个特定先前技术，其中由压缩逻辑 520 使用特定压缩算法来压缩每个个别视频帧 501-503 以产生一系列经压缩的帧 511-513。该技术的一个实施例是“运动 JPEG”，其中根据联合图像专家群 (JPEG) 压缩算法基于离散余弦变换 (DCT) 来压缩每一帧。可使用各种不同类型的压缩算法，然而，仍遵守该基本原理（例如，以小波为基础的压缩算法，诸如 JPEG-2000）。

[0162] 此类型压缩的一个问题在于：其减小了每个帧的数据速率，但其不利用连续帧之间的类似性来减小总视频流的数据速率。举例而言，如图 5 中所说明，假定 640×480×24 比特 / 像素 = 640*480*24/8/1024 = 900 千字节 / 帧 (KB/ 帧）的帧速率，则对于给定质量的图像，运动 JPEG 可能仅将该流压缩 1/10，从而产生 90KB/ 帧的数据流。在 60 帧 / 秒下，此将需要 90KB*8 比特 *60 帧 / 秒 = 42.2Mbps 的信道带宽，其对于美国现今几乎所有的家庭互联网连接而言将为极高的带宽，且对于许多办公室互联网连接而言为过高的带宽。实际上，假定其在此种高带宽的情况下要求恒定数据流，且其将仅服务一个用户，则即使在办公室 LAN 环境中，其也将消耗 100Mbps 以太网 LAN 的带宽的大百分比及支持 LAN 的负担沉重的以太网交换器。因此，当与其他压缩技术（诸如下文所描述的那些技术）相比较时，用于运动视频的压缩无效率。此外，使用有损压缩算法的单个帧压缩算法（如 JPEG 及 JPEG-2000）产生在静止图像中可能不引人注意的压缩假影（例如，场景中的密集树叶内的假影可能不呈现为假影，因为眼并不确切地知道密集树叶应如何呈现）。但是，一旦场景在运动中，假影就可能突出，因为眼侦测到自帧至帧而改变的假影，尽管假影在场景的区域（在该区域中，假影在静止图像中可能不引人注意）中。此导致帧序列中的“背景噪音”的感知，该“背景噪音”的外观与边缘模拟 TV 接收期间可见的“雪花”杂音类似。当然，该类型的压缩仍可用于本文中所描述的特定实施例中，但一般而言，为了避免场景中的背景噪音，对于给定感知质量，需要高数据速率（也即，低压缩比率）。

[0163] 其他类型的压缩（诸如，H.264，或 Windows 媒体 VC9、MPEG2 及 MPEG4）在压缩视频流中均更有效，因为其利用连续帧之间的类似性。这些技术均依赖于用于压缩视频的相同的一般技术。因此，尽管将描述 H.264 标准，但相同的一般原理适用于各种其他压缩算法。大量 H.264 压缩器及解压缩器可用，包括用于压缩 H.264 的 x264 开放源软件库及用于解压缩 H.264 的 FFmpeg（一种视频和音频流方案）开放源软件库。

[0164] 图 6a 及图 6b 说明例示性先前技术压缩技术，其中由压缩逻辑 620 将一系列未经压缩的视频帧 501-503、559-561 压缩成一系列“I 帧”611、671；“P 帧”612-613；及“B 帧”670。图 6a 中的垂直轴大体表示经编码的帧中的每一者的所得大小（尽管所述帧未按比例进行绘制）。如上所述，本领域的技术人员良好地理解使用 I 帧、B 帧及 P 帧的视频写码。简言之，I 帧 611 为完全未压缩的帧 501 的基于 DCT 的压缩（类似于如上所述的经压缩的 JPEG 图像）。P 帧 612-613 的大小通常显著小于 I 帧 611 的大小，因为其利用先前 I 帧或 P 帧中的数据；也即，其含有指示先前 I 帧或 P 帧之间的改变的数据。除 B 帧使用随后参考帧中的帧以及（可能地）之前参考帧中的帧之外，B 帧 670 类似于 P 帧。

[0165] 对于以下论述，将假定所要的帧速率为 60 帧 / 秒，每个 I 帧为约 160Kb，平均 P 帧及 B 帧为 16Kb 且每隔一秒产生一新的 I 帧。在该组参数下，平均数据速率将为：160Kb+16Kb*59 = 1.1Mbps。该数据速率适当地落在用于到家庭及办公室的许多当前宽带

互联网连接的最大数据速率内。该技术也倾向于避免来自仅帧内编码的背景噪音问题，因为P帧及B帧追踪帧之间的差异，因此压缩假影倾向于不自帧至帧而呈现及消失，从而减少上文所描述的背景噪音问题。

[0166] 上述类型的压缩的一个问题在于：尽管平均数据速率相对低（例如，1.1Mbps），但单个I帧可能花费若干个帧时间来传输。举例而言，使用先前技术，2.2Mbps网络连接（例如，DSL或电缆调制解调器，其具有来自图3a的最大可用数据速率302的2.2Mbps峰值）通常将足够使视频以1.1Mbps流动，每60个帧一个160Kbps的I帧。这将通过使解压缩器在解压缩视频之前将1秒视频排入队列来完成。在1秒内，将传输1.1Mb的数据，其将通过2.2Mbps最大可用数据速率来容易地适应，即使假定可用数据速率可能周期性地下降多达50％也如此。遗憾地，此先前技术方法将由于接收器处的1秒视频缓冲而导致视频的1秒延时。此种延迟对于许多先前技术应用程序（例如，线性视频的回放）而言足够，但对于不可容忍大于70毫秒-80毫秒的延时的快动作视频游戏而言其为极长的延时。

[0167] 若进行尝试来消除1秒视频缓冲，其仍将不会导致用于快动作视频游戏的足够延时减少。举例而言，如先前所述，B帧的使用将需要接收I帧的前的所有B帧以及I帧。若假定在P帧与B帧之间大致分裂59个非I帧，则将存在至少29个B帧且可显示任何B帧之前所接收的I帧。因此，不管信道的可用带宽如何，均需要29+1 = 30个帧每一者1/60秒持续时间的延迟，或500毫秒的延时。显而易见，该时间极长。

[0168] 因此，另一方法将是消除B帧且仅使用I帧及P帧。（其中一个后果是：对于给定质量水平，数据速率将增加，但出于此实例中的一致性起见，继续假定每个I帧的大小为160Kb且平均P帧的大小为16Kb，且因此数据速率仍为1.1Mbps）。该方法消除了由B帧引入的不可避免的延时，因为每个P帧的解码仅依赖于先前所接收的帧。该方法仍存在的问题在于：I帧比平均P帧大得多，以致在低带宽信道上（如大多数家庭中及许多办公室中典型的），I帧的传输添加实质延时。此在图6b中加以说明。视频流数据速率624低于可用最大数据速率621（除对于I帧之外），其中I帧所需的峰值数据速率623远超过可用最大数据速率622（且甚至超过额定最大数据速率621）。P帧所需的数据速率小于可用最大数据速率。即使在2.2Mbps的可用最大数据速率峰值稳定地保持在其2.2Mbps峰值速率，也将花费160Kb/2.2Mb = 71毫秒来传输I帧，且若可用最大数据速率622下降50％（1.1Mbps），则将花费142毫秒来传输I帧。因此，传输I帧中的延时将落在71毫秒与142毫秒之间的某处。该延时添加到图4b中所识别的延时（所述延时在最糟状况下总计达70毫秒），因此，这将导致从用户致动输入设备421的时刻直到图像呈现于显示设备422上的为141-222毫秒的总来回行程延时，其极高。且若可用最大数据速率下降至低于2.2Mbps，则延时将进一步增加。

[0169] 还注意到，以远超过可用数据速率622的峰值数据速率623使ISP“堵塞”通常存在严重后果。不同ISP中的设备将不同地表现，但当以比可用数据速率622高得多的数据速率接收分组时，以下行为在DSL及电缆调制解调器ISP当中相当普遍：(a)通过将分组排入队列而使分组延迟（引入延时），(b)丢弃一些或所有分组，(c)将连接停用一时间周期（最可能因为ISP担忧其为恶意攻击，诸如“否认服务”攻击）。因此，以全数据速率传输分组流（具有诸如图6b中所显示的所述特性的特性）并非为可行的选项。可在主机服务210处将峰值623排入队列且以低于可用最大数据速率的数据速率进行发送，从而引入前一段

中所描述的不可接受的延时。

[0170] 另外，图6b中所显示的视频流数据速率序列624为非常“驯服的(tame)”视频流数据速率序列，且将是由于压缩来自视频序列的视频而预期产生的该种数据速率序列，该视频序列并不改变很大且具有非常少的运动（例如，如在视频电话会议中将是普遍的，在视频电话会议中，相机处于固定位置中且具有非常少的运动，且场景（例如，就座的人谈话）中的对象显示较少运动）。

[0171] 图6c中所显示的视频流数据速率序列634为从具有多得多的动作的视频（诸如，可能在电影或视频游戏中或在某应用程序软件中产生）预期可见的典型序列。注意，除I帧峰值633之外，还存在相当大且在许多场合上超过可用最大数据速率的P帧峰值（诸如，635及636）。尽管所述P帧峰值并不如I帧峰值一般相当大，但其仍极大以致不能由信道以全数据速率来运输，且如同I帧峰值一样，P帧峰值必须缓慢地传输（借此增加延时）。

[0172] 在高带宽信道（例如，100Mbps LAN，或高带宽100Mbps私用连接）上，网络将能够容忍诸如I帧峰值633或P帧峰值636的大峰值，但原则上，可维持低延时。但是，所述网络经常在许多用户当中共用（例如，在办公室环境中），且该“有峰”数据将影响LAN的性能，尤其在网络通信经路由至私用共用连接（例如，从远程数据中心到办公室）的情况下。首先，记住此实例是60fps下640×480像素的相对低分辨率的视频流的实例。60fps下的1920×1080的HDTV流容易由现代计算机及显示器来处理，且60fps下的2560×1440分辨率显示器日益可用（例如，Apple公司的30″显示器）。使用H.264压缩，60fps下的1920×1080的高动作视频序列可能需要4.5Mbps以获得合理质量水平。若假定I帧峰值为标定数据速率的10倍，则其将产生45Mbps峰值，以及较小但仍相当大的P帧峰值。若若干个用户正在同一100Mbps网络（例如，办公室与数据中心之间的私用网络连接）上接收视频流，则容易看见来自若干用户的视频流的峰值可如何碰巧对准，从而淹没网络的带宽，且可能淹没网络上支持用户的交换器的底板的带宽。即使在超高速以太网的状况下，若足够的用户具有同时对准的足够峰值，则其可淹没网络或网络交换器。此外，一旦2560×1440分辨率视频变得更常见，平均视频流数据速率就可能为9.5Mbps，从而或许产生95Mbps峰值数据速率。不用说，数据中心与办公室之间的100Mbps连接（其在现今为格外快的连接）将完全被来自单一用户的峰值通信击溃。因此，即使LAN及私用网络连接对有峰流动视频可具有更高容忍度，具有高峰值的流动视频也是不需要的且可能需要办公室的IT部门的特殊计划及适应。

[0173] 当然，对于标准线性视频应用程序，该问题并不是问题，因为数据速率在传输点“经平滑化”且用于每一帧的数据低于最大可用数据速率622，且在解压缩I帧、P帧及B帧序列之前，客户端中的缓冲器储存I帧、P帧及B帧序列。因此，网络上的数据速率保持接近于视频流的平均数据速率。很遗憾地，这引入延时，即使不使用B帧，对于诸如需要快响应时间的视频游戏及应用程序的低延时应用程序而言，这也是不可接受的。

[0174] 用于减轻具有高峰值的视频流的一个先前技术解决方法是使用常常被称作“恒定比特速率”(CBR)编码的技术。尽管术语CBR看来似乎暗示将所有帧压缩以具有相同比特速率（也即，大小），但其经常提及的为压缩范例，在压缩范例中，允许跨越特定数目的帧（在我们的状况下，1个帧）的最大比特速率。举例而言，在图6c的状况下，若对编码施加CBR约束（其将比特速率限于（例如）额定最大数据速率621的70％），则压缩算法将限制所

述帧中的每一者的压缩，以使得通常将使用额定最大数据速率 621 的 70％以上来压缩的任何帧将以较少比特来压缩。这个结果是：将使通常将需要更多比特来维持给定质量水平的帧“极度缺乏”比特且所述帧的图像质量将比不需要比额定最大数据速率 621 的 70％多的比特的其他帧的图像质量糟。此方法对于特定类型的经压缩视频（其中 (a) 预期较少运动或场景改变且 (b) 用户可接受周期性的质量降级）可产生可接受的结果。适合 CBR 的应用的良好实例为视频电话会议，因为存在较少峰值，且在质量暂时降级的情况下（例如，若使相机扫视，从而导致显著场景运动及大峰值，则在扫视期间可能不存在足够的比特用于高质量图像压缩，其将导致降级的图像质量），大多数用户可接受。很遗憾地，CBR 并非良好地适合具有高复杂度的场景或大量运动和 / 或需要合理恒定的质量水平的许多其他应用。

[0175] 在一个实施例中所使用的低延时压缩逻辑 404 使用若干不同技术来解决流动低延时经压缩视频同时维持高质量的许多问题。首先，低延时压缩逻辑 404 仅产生 I 帧及 P 帧，从而缓解等待若干个帧时间来解码每个 B 帧的需要。另外，如图 7a 中所说明，在一个实施例中，低延时压缩逻辑 404 将每个未经压缩的帧 701-760 再分成一系列“图像块 (tile)”且将每个图像块个别地编码为 I 帧或 P 帧。在本文中将该群经压缩的 I 帧及 P 帧称作“R 帧”711-770。在图 7a 中所显示的特定实例中，将每个未经压缩的帧再分成 4×4 矩阵的 16 个图像块。然而，所述基本原理不限于任何特定再分机制。

[0176] 在一实施例中，低延时压缩逻辑 404 将视频帧划分成许多图像块，且将来自每个帧的一个图像块编码（也即，压缩）为 I 帧（也即，将该图像块压缩，好像其为全图像的大小的 1/16 的单独视频帧，且用于此“迷你型”帧的压缩为 I 帧压缩）并将剩余图像块编码为 P 帧（也即，用于每个“迷你型”1/16 帧的压缩为 P 帧压缩）。经压缩为 I 帧的图像块及经压缩为 P 帧的图像块将分别被称作“I 图像块”及“P 图像块”。随着每个相继视频帧而改变待编码为 I 图像块的图像块。因此，在给定帧时间中，视频帧中的所述图像块中仅一个图像块为 I 图像块，且所述图像块中的剩余者为 P 图像块。举例而言，在图 7a 中，未经压缩的帧 701 的图像块 0 经编码为 I 图像块 I0 且剩余的 1-15 图像块经编码为 P 图像块 (P1 至 P15) 以产生 R 帧 711。在下一个未经压缩的视频帧 702 中，未经压缩的帧 701 的图像块 1 经编码为 I 图像块 I1 且剩余的图像块 0 及 2 至 15 经编码为 P 图像块 (P0，及 P2 至 P15) 以产生 R 帧 712。因此，用于图像块的 I 图像块及 P 图像块在相继帧上逐渐地在时间上交错。该过程继续，直至产生 R 图像块 770（矩阵中最末图像块经编码为 I 图像块（也即，I15））为止。该过程接着重新开始，从而产生诸如帧 711（也即，对于图像块 0，编码 I 图像块）等的另一个 R 帧。尽管图 7a 中未说明，但在一个实施例中，R 帧的视频序列的第一 R 帧仅含有 I 图像块（也即，以使得随后的 P 帧具有参考图像数据（自其开始计算运动））。或者，在一实施例中，启动序列使用与正常相同的 I 图像块型样，但不包括用于尚未连同 I 图像块一起编码的所述图像块的 P 图像块。换言之，在第一 I 图像块到达之前不连同任何数据一起编码特定图像块，从而避免图 9a 中的视频流数据速率 934 中的启动峰值，其在下文进一步详细说明。此外，如下所述，各种不同大小及形状可用于所述图像块同时仍遵守所述基本原理。

[0177] 在客户端 415 上执行的视频解压缩逻辑 412 解压缩每个图像块，好像其为小 I 帧及 P 帧的单独视频序列，且接着将每个图像块再现给驱动显示设备 422 的帧缓冲器。举例而言，使用来自 R 帧 711 至 770 的 I0 及 P0 来解压缩并再现视频图像的图像块 0。类似地，使用来自 R 帧 711 至 770 的 I1 及 P1 来重建图像块 1，等等。如上所述，I 帧及 P 帧的解压

缩是这项技术中众所熟知的，且 I 图像块及 P 图像块的解压缩可通过使视频解压缩器的多个执行个体在客户端 415 上执行来完成。尽管倍增过程看来似乎增加客户端 415 上的计算负担，但实际上其不会增加客户端 415 上的计算负担，因为图像块本身成比例地较小（相对于额外处理的数目而言），因此所显示的像素的数目相同，好像存在一个处理且使用传统的全大小的 I 帧及 P 帧。

[0178] 该 R 帧技术显著减轻通常与 I 帧相关联的带宽峰值（图 6b 及图 6c 中所说明），因为任何给定帧主要是由通常比 I 帧小的 P 帧构成。举例而言，再次假定典型 I 帧为 160Kb，则图 7a 中所说明的帧中的每一者的 I 图像块将为该量的大致 1/16 或 10Kb。类似地，假定典型 P 帧为 16Kb，则用于图 7a 中所说明的图像块中的每一者的 P 帧可为大致 1Kb。最终结果为约 10Kb+15*1Kb ＝ 25Kb 的 R 帧。因此，每一 60 帧序列将是 25Kb*60 ＝ 1.5Mbps。因此，在 60 帧 / 秒下，这将需要能够维持 1.5Mbps 的带宽的信道，但由于 I 图像块是贯穿 60 帧间隔而分散而使得具有低得多的峰值。

[0179] 注意，在先前实例中，在用于 I 帧及 P 帧的相同假定数据速率情况下，平均数据速率为 1.1Mbps。这是因为在先前实例中，每隔 60 个帧时间仅引入新 I 帧，而在此实例中，构成 I 帧的 16 个图像块在 16 个帧时间中循环，且因此，每隔 16 个帧时间引入 I 帧的均等物，从而导致稍高的平均数据速率。尽管如此，但在实践中，引入更频繁的 I 帧并不会线性地增加数据速率。这是由于以下事实 :P 帧（或 P 图像块）主要编码自先前帧至下一个帧的差异。因此，若先前帧与下一个帧相当类似，则 P 帧将非常小，若先前帧与下一个帧相当不同，则 P 帧将非常大。但因为 P 帧很大程度上是自先前帧导出，而非自实际帧导出，所以所得的经编码帧可能含有比具有足够数目的比特的 I 帧多的错误（例如，视觉假影）。此外，当一个 P 帧跟随另一 P 帧时，可出现错误累加（当存在长 P 帧序列时，变得更糟）。现在，尖端的视频压缩器将侦测到图像的质量在一序列 P 帧的后降级的事实，且必要时，其将更多比特分配给随后的 P 帧以提高质量，或若其为最有效的动作过程，则用 I 帧替换 P 帧。因此，当使用长 P 帧序列（例如，59 个 P 帧，如上文先前实例中）时，特定言的当场景具有大量复杂度和 / 或运动时，通常，对于 P 帧而言需要更多比特（因为其变得距 I 帧更远）。

[0180] 或者，自相对观看点看 P 帧，紧密地跟随 I 帧的 P 帧倾向于需要比距 I 帧更远的 P 帧少的比特。因此，在图 7a 中所显示的实例中，没有 P 帧比距 I 帧隔开 15 个帧更远（在 I 帧之前），而在先前实例中，P 帧可能从 I 帧隔开 59 个帧。因此，在更频繁的 I 帧情况下，P 帧较小。当然，确切相对大小将基于视频流的性质而改变，但在图 7a 的实例中，若 I 图像块为 10Kb，则 P 图像块的大小平均可为仅 0.75kb，从而导致 10Kb+15*0.75Kb ＝ 21.25Kb，或在 60 帧 / 秒下，数据速率将为 21.25Kb*60 ＝ 1.3Mbps，或比 1.1Mbps 下的具有 I 帧随后跟着 59 个 P 帧的流的数据速率高约 16％。再一次，用于视频压缩的所述两种方法之间的相对结果将视视频序列而改变，但通常，我们凭经验发现，对于给定质量水平，使用 R 帧比使用 I/P 帧序列需要多约 20％的比特。但是，当然，R 帧急剧地减少峰值，这使视频序列在远小于 I/P 帧序列的延时下可用。

[0181] 可视视频序列的性质、信道的可靠性及可用数据速率而以多种不同方式来配置 R 帧。在替代实施例中，在 4×4 配置中使用不同于 16 的数目的图像块。举例而言，可在 2×1 或 1×2 配置中使用 2 个图像块，可在 2×2、4×1 或 1×4 配置中使用 4 个图像块，可在 3×2、2×3、6×1 或 1×6 配置中使用 6 个图像块或可在 4×2（如图 7b 中所显示）、2×4、8×1 或

1×8 配置中使用 8 个图像块。注意，图像块不需要为方形，视频帧也不必为方形，或甚至矩形。可将图像块分解成最佳地适合所使用的视频流及应用程序的无论什么形状。

[0182] 在另一实施例中，I 图像块及 P 图像块的循环不锁定到图像块的数目。举例而言，在 8 图像块 4×2 配置中，仍可如图 7b 中所说明而使用 16 循环序列。顺序的未经压缩的帧 721、722、723 各自经划分成 8 个图像块 0-7，且每个图像块被个别压缩。自 R 帧 731，仅图像块 0 被压缩为 I 图像块，且剩余图像块被压缩为 P 图像块。对于随后的 R 帧 732，所有 8 个图像块被压缩为 P 图像块，且接着对于随后的 R 帧 733，图像块 1 被压缩为 I 图像块且其他图像块均被压缩为 P 图像块。此外，如此对于 16 个帧继续进行排序，仅每隔一帧产生一个 I 图像块，因此在第 15 个帧时间期间（图 7b 中未图示）及在第 16 个帧时间期间产生用于图像块 7 的最末 I 图像块（使用所有的 P 图像块压缩 R 帧 780）。接着，序列再次以图像块 0 被压缩为 I 图像块且其他图像块被压缩为 P 图像块开始。如在先前实施例中，整个视频序列的第一帧通常将均为 I 图像块，以提供用于从该点向前的 P 图像块的参考。I 图像块及 P 图像块的循环甚至不需要为图像块的数目的偶倍数。举例而言，在 8 个图像块的情况下，在使用另一 I 图像块之前，具有一个 I 图像块的每一帧之后可为所有都为 P 图像块的 2 个帧。在又一实施例中，若（例如）已知屏幕的特定区域具有更多运动（需要更频繁的 I 图像块），而其他区域更为静态（例如，显示游戏的分数）（需要较不频繁的 I 图像块），则与其他图像块相比，可更经常地将特定图像块连同 I 图像块一起进行排序。此外，尽管在图 7a- 图 7b 中说明每个帧具有单个 I 图像块，但可在单个帧中编码多个 I 图像块（取决于传输信道的带宽）。相反地，特定帧或帧序列可在不具有 I 图像块（也即，仅 P 图像块）的情况下传输。

[0183] 前一段的方法适当起作用的原因在于：尽管不具有跨越每个单个帧而分散的 I 图像块看来似乎导致较大峰值，但系统的行为并不如此简单。因为每个图像块是与其他图像块分开进行压缩，所以当图像块变小时，每个图像块的编码可变得较不有效，因为给定图像块的压缩器不能够利用来自其他图像块的类似图像特征及类似运动。因此，与将屏幕划分成 8 个图像块相比较，将屏幕划分成 16 个图像块通常将导致较不有效的编码。但是，若将屏幕划分成 8 个图像块且其引起每隔 8 个帧（而非每隔 16 个帧）引入一个完全 I 帧的数据，则其导致总体上高得多的数据速率。因此，通过每隔 16 个帧（而非每隔 8 个帧）引入一个完全 I 帧，减小了总数据速率。而且，通过使用 8 个较大图像块（而非 16 个较小图像块），减小了总数据速率，其也将由较大图像块引起的数据峰值减轻至某种程度。

[0184] 在另一实施例中，图 7a 及图 7b 中的低延时视频压缩逻辑 404 通过基于待压缩的视频序列的已知特性而通过设定预先配置或者基于每个图像块中的图像质量的正在进行的分析而自动地控制至 R 帧中的各图像块的比特的分配。举例而言，在一些竞赛视频游戏中，玩家的汽车（其为场景中相对无运动的）的前方占据屏幕的下半部的大部分，而屏幕的上半部完全被填满正接近的道路、建筑物及风景，其几乎总是在运动中。若压缩逻辑 404 将相等数目的比特分配给每个图像块，则图 7b 中的未经压缩的帧 721 中的屏幕的下半部上的图像块（图像块 4-7）通常将以比图 7b 中的未经压缩的帧 721 中的屏幕的上半部中的图像块（图像块 0-3）高的质量而压缩。若已知该特定游戏或游戏的此特定场景具有所述特性，则主机服务 210 的运营商可配置压缩逻辑 404 以将更多比特分配给屏幕的顶部的图像块（与分配给屏幕的底部处的图像块的比特相比）。或者，压缩逻辑 404 可在压缩帧之

后估计图像块的压缩质量（使用许多压缩质量度量中的一者或多者，诸如峰值信号噪音比(PSNR)），且若确定在特定时间窗上，特定图像块始终如一地产生较佳质量结果，则其逐渐地将更多比特分配给产生较低质量结果的图像块，直至各种图像块达到类似水平的质量为止。在替代实施例中，压缩器逻辑 404 分配比特以在特定图像块或图像块群中达成较高质量。举例而言，其可提供较佳的总体感知外观，以在屏幕的中心具有比边缘处高的质量。

[0185] 在一个实施例中，为了改良视频流的特定区域的分辨率，视频压缩逻辑 404 使用较小图像块来编码视频流的具有相对多的场景复杂度和 / 或运动的区域（与视频流的具有相对少的场景复杂度和 / 或运动的区域相比）。举例而言，如图 8 中所说明，在一个 R 帧 811（可能随后跟着具有相同图像块大小的一系列 R 帧（未图示））的一个区域中的移动人物 805 的周围使用较小图像块。接着，当人物 805 移动至图像的新区域时，在另一 R 帧 812 内的此新区域的周围使用较小图像块，如所说明。如上所述，各种不同大小及形状可用作“图像块”同时仍遵守所述基本原理。

[0186] 尽管上文所描述的循环 I/P 图像块实质上减小视频流的数据速率中的峰值，但其并不完全消除峰值，尤其在快速改变或高度复杂的视频图像（诸如在电影、视频游戏及某一应用程序软件下出现）的状况下。举例而言，在突然场景转变期间，复杂帧可能随后跟着完全不同的另一复杂帧。即使若干个 I 图像块可领先于场景转变仅几个帧时间，其在该情形下也没有帮助，因为新帧的材料与先前 I 图像块无关。在这种情形下（及在即使并非一切都改变，大量图像也改变的其他情形下），视频压缩器 404 将确定将许多（若并非所有）P 图像块更有效地写码为 I 图像块，且所导致的是所述帧的数据速率中的非常大的峰值。

[0187] 如先前所论述，其仅为对于大多数消费者级互联网连接（及许多办公室连接）的状况，其仅不可“堵塞”超过图 6c 中显示为 622 的可用最大数据速率以及额定最大数据速率 621 的数据。注意，额定最大数据速率 621（例如，“6Mbps DSL”）实质上为对于考虑购买互联网连接的用户的销售数字，但通常其不保证性能水平。出于该应用的目的，其是不相关的，因为我们仅关注经由连接使视频流动时的可用最大数据速率 622。因此，在图 9a 及图 9c 中，当描述对峰值问题的解决方法时，自曲线图省略额定最大数据速率，且仅显示可用最大数据速率 922。视频流数据速率不得超过可用最大数据速率 922。

[0188] 为了解决此问题，视频压缩器 404 进行的第一件事是确定峰值数据速率 941，其为信道能够稳定地处理的数据速率。该速率可通过许多技术来确定。一种该技术是将越加变高的数据速率测试流自主机服务 210 逐渐发送至客户端 415（图 4a 及图 4b 中），且使客户端将关于分组丢失及延时的水平的反馈提供至主机服务。当分组丢失和 / 或延时开始显示尖锐增加时，其为达到可用最大数据速率 922 的指示。之后，主机服务 210 可逐渐地减小测试流的数据速率，直至客户端 415 报告在合理的时间周期中已接收到测试流（分组丢失处于可接受水平，且延时接近最小）为止。这确定峰值最大数据速率 941，其接着将用作用于流动视频的峰值数据速率。随着时间的推移，峰值数据速率 941 将波动（例如，若家庭中的另一用户开始严重地使用互联网连接），且客户端 415 将需要恒定地监视峰值数据速率 941 以观看分组丢失或延时是否增加（指示可用最大数据速率 922 下降至低于先前所确定的峰值数据速率 941），且若如此，则峰值数据速率 941。类似地，若随着时间的推移，客户端 415 发现分组丢失及延时保持在最佳水平，则其可请求视频压缩器缓慢地增加数据速率以观看可用最大数据速率是否增加（例如，若家庭中的另一用户已停止对互联网连接的严重使用），

且再次等待直至分组丢失和 / 或较高延时指示已超过可用最大数据速率 922 为止，且可再次发现用于峰值数据速率 941 的较低水平，但该较低水平可能高于测试增加的数据速率之前的水平。因此，可通过使用该技术（及类似其的其他技术）而发现峰值数据速率 941，且视需要而周期性地进行调整。峰值数据速率 941 确定可由视频压缩器 404 使用以使视频流动至用户的最大数据速率。用于确定峰值数据速率的逻辑可在用户场所 211 处和 / 或在主机服务 210 上加以实施。在用户场所 211 处，客户端设备 415 执行计算以确定峰值数据速率且将此信息传输回至主机服务 210 ；在主机服务 210 处，主机服务处的服务器 402 执行计算以基于自客户端 415 所接收的统计数据（例如，峰值丢失、延时、最大数据速率等）而确定峰值数据速率。

[0189] 图 9a 显示具有实质场景复杂度和 / 或运动的实例视频流数据速率 934，其是使用先前所描述且在图 7a、图 7b 及图 8 中加以说明的循环 I/P 图像块压缩技术而产生。视频压缩器 404 经配置而以低于峰值数据速率 941 的平均数据速率输出经压缩的视频，且注意，大部分时间，视频流数据速率保持低于峰值数据速率 941。数据速率 934 与图 6c 中所显示的视频流数据速率 634（其是使用 I/P/B 或 I/P 帧而产生）的比较显示循环 I/P 图像块压缩产生平滑得多的数据速率。但在帧 2 倍峰值 952（其接近 2 倍峰值数据速率 942）及帧 4 倍峰值 954（其接近 4 倍峰值数据速率 944）下，数据速率仍超过峰值数据速率 941，这是不可接受的。在实践中，即使对于来自快速改变的视频游戏的高动作视频，超过峰值数据速率 941 的峰值也在小于 2％的帧中出现，超过 2 倍峰值数据速率 942 的峰值很少出现，且超过 3 倍峰值数据速率 943 的峰值难得出现。但是，当其确实出现时（例如，在场景转变期间），其所需的数据速率必须产生良好质量的视频图像。

[0190] 解决此问题的一个方式是简单地配置视频压缩器 404 以使得其最大数据速率输出为峰值数据速率 941。遗憾地，峰值帧期间的所得视频输出质量不良，因为压缩算法“极度缺乏”比特。所导致的为当存在突然转变或快速运动时出现压缩假影，且及时地，用户开始认识到 ：当存在突然改变或快速运动时假影总是突然出现，且其可变得相当讨厌。

[0191] 尽管人的视觉系统对在突然改变或快速运动期间出现的视觉假影相当敏感，但对在所述情形下侦测到帧速率的减小并不是非常敏感。事实上，当所述突然改变出现时，看来似乎人的视觉系统专注于追踪所述改变，且若帧速率暂时从 60fps 下降到 30fps 且接着立即返回到 60fps，则人的视觉系统不会注意到。此外，在非常急剧的转变（如突然场景改变）的状况下，若帧速率下降到 20fps 或甚至 15fps 且接着立即返回到 60fps，则人的视觉系统不会注意到。只要帧速率减小仅偶尔出现，对于人观察者而言，看来似乎视频以 60fps 不断地执行。

[0192] 通过图 9b 中所说明的技术来利用人的视觉系统的此特性。服务器 402（来自图 4a 及图 4b）以稳定帧速率（在一个实施例中，在 60fps 下）产生未经压缩的视频输出流。时间线显示每个 1/60 秒每个帧 961-970 输出。自帧 961 开始，将每个未经压缩的视频帧输出至低延时视频压缩器 404，低延时视频压缩器 404 在小于一帧时间的时间中压缩所述帧，产生用于第一帧的经压缩的帧 1 981。经产生用于经压缩的帧 1 981 的数据可视如先前所描述的许多因素而较大或较小。若数据足够小以致可以峰值数据速率 941 在一帧时间 (1/60 秒）或小于一帧时间内将其传输至客户端 415，则在传输时间 (xmit 时间）991（指示传输时间的持续时间的箭头的长度）期间将其传输。在下一个帧时间中，服务器 402 产生未经压

缩的帧 2 962，将其压缩成经压缩的帧 2 982，且在小于一帧时间的传输时间 992 期间以峰值数据速率 941 将其传输至客户端 415。

[0193] 接着，在下一个帧时间中，服务器 402 产生未经压缩的帧 3 963。当由视频压缩器 404 来压缩未经压缩的帧 3 963 时，所得的经压缩的帧 3 983 为比可以峰值数据速率 941 在一帧时间中传输的数据多的数据。因此，在传输时间（2 倍峰值）993 期间将其传输，其占据所有帧时间及下一个帧时间的一部分。现在，在下一个帧时间期间，服务器 402 产生另一未经压缩的帧 4 964 且将其输出至视频压缩器 404，但数据被忽略且通过 974 来说明。这是因为视频压缩器 404 经配置以忽略在其仍传输先前经压缩的帧时到达的其他未经压缩的视频帧。当然，客户端 415 的视频解压缩器将未能接收到帧 4，但其简单地继续在显示设备 422 上显示帧 3 历时 2 个帧时间（也即，暂时将帧速率自 60fps 减小至 30fps）。

[0194] 对于下一个帧 5，服务器 402 输出未经压缩的帧 5 965，将其压缩成经压缩的帧 5 985 且在传输时间 995 期间在 1 帧内将其传输。客户端 415 的视频解压缩器解压缩帧 5 并将其显示于显示设备 422 上。接着，服务器 402 输出未经压缩的帧 6 966，视频压缩器 404 将其压缩成经压缩的帧 6 986，但此时所得的数据非常大。在传输时间（4 倍峰值）996 期间以峰值数据速率 941 传输经压缩的帧，但花费几乎 4 个帧时间来传输帧。在接下来的 3 个帧时间期间，视频压缩器 404 忽略来自服务器 402 的 3 个帧，且客户端 415 的解压缩器将帧 6 稳定地保持在显示设备 422 上历时 4 个帧时间（也即，暂时将帧速率自 60fps 减小至 15fps）。接着最后，服务器 402 输出帧 10 970，视频压缩器 404 将其压缩成经压缩的帧 10 987，且在传输时间 997 期间将其传输，且客户端 415 的解压缩器解压缩帧 10 并将其显示于显示设备 422 上且再一次视频以 60fps 重新开始。

[0195] 注意，尽管视频压缩器 404 丢弃了来自由服务器 402 产生的视频流的视频帧，但其不会丢弃音频数据（不管音频是以什么形式来的），且当丢弃视频帧时视频压缩器 404 继续压缩音频数据并将其传输至客户端 415，客户端 415 继续解压缩音频数据并将音频提供至由用户使用以回放音频的无论什么设备。因此在丢弃帧的周期期间，音频继续而不减弱。与经压缩的视频相比，经压缩的音频消耗相对小百分比的带宽，且因此不会对总数据速率有较大影响。尽管在数据速率图中的任一者中都未说明，但峰值数据速率 941 内总是存在经保留用于经压缩音频流的数据速率容量。

[0196] 选择刚刚在图 9b 中所描述的实例来说明在数据速率峰值期间帧速率如何下降，但未说明的是当使用先前所描述的循环 I/P 图像块技术时，所述数据速率峰值及随的发生的丢弃的帧很少，即使在高场景复杂度 / 高动作序列（诸如在视频游戏、电影及某个应用程序软件中出现的那些）期间也如此。因此，减小的帧速率罕有且暂时，且人的视觉系统不会侦测到它们。

[0197] 若将刚刚所描述的帧速率减小机制应用于图 9a 中所说明的视频流数据速率，则在图 9c 中说明所得的视频流数据速率。在此实例中，2 倍峰值 952 已减小至平坦化的 2 倍峰值 953，且 4 倍峰值 955 已减小至平坦化的 4 倍峰值 955，且整个视频流数据速率 934 保持处于或低于峰值数据速率 941。

[0198] 因此，使用上文所描述的技术，可经由通用互联网及消费者级互联网连接而以低延时来传输高动作视频流。另外，在 LAN（例如，100Mbs 以太网或 802.11g 无线网络）上或私用网络（例如，数据中心与办公室之间的 100Mbps 连接）上的办公室环境中，可在无峰值情

况下传输高动作视频流，以使得多个用户（例如，以 4.5Mbps 传输 60fps 下的 1920×1080）可使用 LAN 或共用私用数据连接，而不使重迭峰值淹没 (overwhelm) 网络或网络交换器底板。

[0199] 数据速率调整

[0200] 在一个实施例中，主机服务 210 最初评估信道的可用最大数据速率 622 及延时以确定用于视频流的适当数据速率且接着响应于此而动态地调整数据速率。为了调整数据速率，主机服务 210 可（例如）修改待发送至客户端 415 的视频流的图像分辨率和 / 或每秒帧数。而且，主机服务可调整经压缩视频的质量水平。当改变视频流的分辨率时（例如，自 1280×720 分辨率至 640×360），客户端 415 上的视频解压缩逻辑 412 可将图像按比例增加以在显示屏幕上维持相同图像大小。

[0201] 在一个实施例中，在信道完全掉线 (drop out) 的情形下，主机服务 210 将游戏暂停。在多人游戏的状况下，主机服务向其他用户报告该用户游戏已掉线和 / 或针对其他用户暂停所述游戏。

[0202] 丢弃或延迟的分组

[0203] 在一个实施例中，若数据由于图 4a 或图 4b 中的视频压缩器 404 与客户端 415 之间的分组丢失而丢失，或由于到达得过晚以致不能解压缩及满足经解压缩帧的延时要求的分组被无次序地接收而丢失，则视频解压缩逻辑 412 能够减轻视觉假影。在流动 I/P 帧实施中，若存在丢失 / 延迟的分组，则整个屏幕受影响，从而可能引起屏幕完全冻结一段时间周期或显示其他屏幕宽视觉假影。举例而言，若丢失 / 延迟的分组引起 I 帧的丢失，则在接收新的 I 帧之前，解压缩器将缺乏用于跟随的所有 P 帧的参考。若丢失 P 帧，则其将影响跟随的用于整个屏幕的 P 帧。视 I 帧出现之前有多久，这将具有较长或较短的视觉影响。使用如图 7a 及图 7b 中所显示的交错 I/P 图像块，丢失 / 延迟的分组不太可能影响整个屏幕，因为其仅影响受影响的分组中所含有的图像块。若每个图像块的数据是在个别分组内发送，则若分组丢失，则其仅影响一个图像块。当然，视觉假影的持续时间将取决于 I 图像块分组是否丢失及在 P 图像块丢失的情况下在 I 图像块出现之前将花费多少个帧。但是，假定屏幕上的不同图像块是通过 I 帧非常频繁地（可能每个帧）更新，则即使屏幕上的一图像块受影响，其他图像块也可能不受影响。另外，若某一事件引起若干分组同时丢失（例如，邻接 DSL 线的电力中的暂时中断数据流的尖峰信号），则一些图像块将比其他图像块受到更大影响，但因为一些图像块将通过新的 I 图像块迅速地更新，所以其仅暂时受影响。而且，在流动 I/P 帧实施的情况下，不仅 I 帧为最关键帧，而且 I 帧极大，因此若存在引起丢弃 / 延迟的分组的事件，则与小得多的 I 图像块相比，I 帧受影响存在较高机率（也即，若 I 帧的任何部分丢失，则根本不可能可解压缩 I 帧）。由于所有所述原因，与 I/P 帧的情况相比，当分组被丢弃 / 延迟时，使用 I/P 图像块导致小得多的视觉假影。

[0204] 一个实施例试图通过将经压缩的图像块智能地封装于 TCP（传输控制协议）分组或 UDP（用户数据报协议）分组内而减少丢失分组的效应。举例而言，在一个实施例中，只要可能，即将图像块与分组边界对准。图 10a 说明可如何在不实施此特征的情况下将图像块封装于一系列分组 1001-1005 内。具体言之，在图 10a 中，图像块越过分组边界且经无效率地封装以致单一分组的丢失导致多个帧的丢失。举例而言，若分组 1003 或 1004 丢失，则丢失三个图像块，导致视觉假影。

[0205] 相比之下，图 10b 说明用于将图像块智能地封装于分组内以减少分组丢失的效应的图像块封装逻辑 1010。首先，图像块封装逻辑 1010 将图像块与分组边界对准。因此，图像块 T1、T3、T4、T7 及 T2 分别与分组 1001-1005 的边界对准。图像块封装逻辑也试图以可能的更有效的方式将图像块组合于分组内，而不越过分组边界。基于图像块中的每一者的大小，将图像块 T1 与 T6 组合于一个分组 1001 中；将 T3 与 T5 组合于一个分组 1002 中；将图像块 T4 与 T8 组合于一个分组 1003 中；将图像块 T8 添加至分组 1004；且将图像块 T2 添加至分组 1005。因此，在此方案下，单个分组丢失将导致不多于 2 个图像块（而非如图 10a 中所说明的 3 个图像块）的丢失。

[0206] 图 10b 中所显示的实施例的一个额外益处在于：图像块是以其在图像内被显示的不同次序进行传输。若邻近分组由于干扰传输的同一事件（其将影响屏幕上彼此不接近的区域）而丢失，则此方式在显示器上产生较不引人注意的假影。

[0207] 一个实施例使用前向纠错 (FEC) 技术来保护视频流的特定部分以使其免受信道错误的影响。如此项技术中已知，诸如里德 - 所罗门及 Viterbi（维特比）的 FEC 技术产生纠错数据信息并将其附加至经由通信信道而传输的数据。若错误在基本数据（例如，I 帧）中出现，则 FEC 可用于校正该错误。

[0208] FEC码增加传输的数据速率，因此理想地，其仅在最需要时使用。若数据正被发送，且其将不导致非常引人注意的视觉假影，则可较佳不使用 FEC 码来保护数据。举例而言，紧接于丢失的 I 图像块之前的 P 图像块将仅在屏幕上产生 1/60 秒的视觉假影（也即，屏幕上的图像块将不被更新）。此种视觉假影几乎不能被人眼侦测到。随着 P 图像块自 I 图像块进一步向后，丢失 P 图像块越加变得更引人注意。举例而言，若图像块循环型样为在 I 图像块再次可用之前 I 图像块随后跟着 15 个 P 图像块，则若紧接于 I 图像块之后的 P 图像块丢失，则其导致该图像块显示不正确的图像历时 15 个帧时间（在 60fps 下，这将为 250 毫秒）。人眼将容易侦测到 250 毫秒的流的中断。因此，P 图像块距新的 I 图像块越向后（也即，P 图像块跟随 I 图像块越接近），则假影越引人注意。如先前所论述，尽管如此，但一般而言，P 图像块跟随 I 图像块越接近，用于该 P 图像块的数据越小。因此，跟随 I 图像块的 P 图像块不仅对于保护以免丢失而言更关键，而且其大小较小。此外，一般而言，需要保护的数据越小，保护其所需的 FEC 码越小。

[0209] 因此，如图 11a 中所说明，在一个实施例中，由于 I 图像块在视频流中的重要性，仅 I 图像块具备 FEC 码。因此，FEC 1101 含有用于 I 图像块 1100 的纠错码且 FEC 1104 含有用于 I 图像块 1103 的纠错码。在此实施例中，对于 P 图像块不产生 FEC。

[0210] 在图 11b 中所说明的一个实施例中，对于在丢失时最可能引起视觉假影的 P 图像块也产生 FEC 码。在此实施例中，FEC 1105 提供用于前 3 个 P 图像块但不用于跟随的 P 图像块的纠错码。在另一实施例中，对于数据大小最小的 P 图像块产生 FEC 码（其将倾向于自选在 I 图像块的后最早出现的 P 图像块，其对于保护最为关键）。

[0211] 在另一实施例中，并非将 FEC 码连同图像块一起发送，而是将图像块传输两次，每次在不同分组中传输。若一分组丢失 / 延迟，则使用另一分组。

[0212] 在图 11c 中所显示的一个实施例中，产生分别用于与视频同时自主机服务传输的音频分组 1110 及 1112 的 FEC 码 1111 及 1113。维持视频流中的音频的完整性特别重要，因为失真的音频（例如，滴答声或嘶嘶声）将导致特别不合需要的用户体验。FEC 码帮助确保

音频内容在客户端计算机 415 处无失真地再现。

[0213] 在另一实施例中，并非将 FEC 码连同音频数据一起发送，而是将音频数据传输两次，每次在不同分组中传输。若一个分组丢失 / 延迟，则使用另一分组。

[0214] 另外，在图 11d 中所说明的一个实施例中，FEC 码 1121 及 1123 分别用于自客户端 415 上行传输到主机服务 210 的用户输入命令（例如，按钮按压）1120 及 1122。这是重要的，因为在视频游戏或应用程序中漏掉按钮按压或鼠标运动可能导致不合需要的用户体验。

[0215] 在另一实施例中，并非将 FEC 码连同用户输入命令数据一起发送，而是将用户输入命令数据传输两次，每次在不同分组中传输。若一个分组丢失 / 延迟，则使用另一分组。

[0216] 在一个实施例中，主机服务 210 评估与客户端 415 的通信信道的质量，以确定是否使用 FEC，且若使用，则确定应对视频、音频及用户命令的何部分应用 FEC。评估信道的“质量”可包括如上所述的诸如估计分组丢失、延时等的功能。若信道特别不可靠，则主机服务 210 可对所有 I 图像块、P 图像块、音频及用户命令应用 FEC。相比之下，若信道可靠，则主机服务 210 可仅对音频及用户命令应用 FEC，或可不对音频或视频应用 FEC，或可根本不使用 FEC。可使用 FEC 的应用的各种其他排列，同时仍遵守所述基本原理。在一个实施例中，主机服务 210 不断地监视信道的状况且相应地改变 FEC 策略。

[0217] 在另一实施例中，参看图 4a 及图 4b，当分组丢失 / 延迟，从而导致图像块数据的丢失时，或若可能由于特别糟的分组丢失而使得 FEC 不能够校正丢失的图像块数据，客户端 415 评估在将接收新的 I 图像块之前剩余多少个帧且将其与自客户端 415 至主机服务 210 的来回行程延时相比较。若来回行程延时小于新的 I 图像块应到达之前的帧的数目，则客户端 415 向主机服务 210 发送消息，请求新的 I 图像块。将此消息路由至视频压缩器 404，且其并非产生用于数据已丢失的图像块的 P 图像块，而是产生 I 图像块。假定图 4a 及图 4b 中所显示的系统经设计以提供通常小于 80 毫秒的来回行程延时，则这导致图像块被校正于 80 毫秒内（在 60fps 下，帧具有 16.67 毫秒的持续时间，因此在全帧时间中，80 毫秒延时将导致 83.33 毫秒内的经校正的图像块，83.33 毫秒为 5 个帧时间，其为引人注意的中断，但远不及（例如）对于 15 个帧 250 毫秒中断引人注意）。当压缩器 404 脱离其通常的循环次序而产生此种 I 图像块时，若 I 图像块将引起所述帧的带宽超过可用带宽，则压缩器 404 将延迟其他图像块的循环，以使得其他图像块在所述帧时间期间接收 P 图像块（即使在所述帧期间一个图像块通常将应为 I 图像块），且接着通常的循环将自下一个帧开始继续，且通常将已接收到先前帧中的 I 图像块的图像块将接收 I 图像块。尽管此动作暂时延迟 R 帧循环的阶段，但其通常将在视觉上不引人注意。

[0218] 视频及音频压缩器 / 解压缩器实施

[0219] 图 12 说明一个特定实施例，其中使用多核和 / 或多处理器 1200 来并行地压缩 8 个图像块。在一实施例中，使用在 2.66GHz 或更高下执行的双核处理器、四核 Xeon（至强）CPU 计算机系统，每个核心作为独立过程实施开源 x264 H.264 压缩器。然而，可使用各种其他硬件 / 软件配置，同时仍遵守所述基本原理。举例而言，CPU 核心中的每一者可通过以 FPGA 实施的 H.264 压缩器来替换。在图 12 中所显示的实例中，核心 1201-1208 用于作为八个独立线绪来同时处理 I 图像块及 P 图像块。如此项技术中众所周知的，当前多核及多处理器计算机系统与诸如 Microsoft Windows XP 专业版 (64 比特版或者 32 比特版）及 Linux 的

多线绪处理操作系统整合时，其固有地能够进行多线绪处理。

[0220] 在图 12 中所说明的实施例中，因为该 8 个核心中的每一者仅负责一个图像块，所以其很大程度上独立于其他核心而操作，每一者执行 x264 的单独实例化。使用以 PCI Express x1 为基础的 DVI 捕获卡（诸如，来自 Netherlands 的 Microtronix of Oosterhout 的 Sendero 视频成像 IP 开发板）来捕获 640×480、800×600 或 1280×720 分辨率下的未经压缩的视频，且卡上的 FPGA 使用直接存储器存取 (DMA) 来将所捕获的视频经由 DVI 总线传送至系统 RAM 中。将所述图像块配置成 4×2 配置 1205（尽管其说明为方形图像块，但在该实施例中，其具有 160×240 分辨率）。x264 的每个实例化被配置成压缩该 8 个 160×240 图像块中的一者，且其经同步化以使得在初始 I 图像块压缩的后每一核心进入一循环，每一帧与另一帧不同相，以压缩一 I 图像块继的以七个 P 图像块，如图 12 中所说明。

[0221] 在每一帧时间，使用先前所描述的技术将所得的经压缩图像块组合成分组流，且接着将经压缩图像块传输至目的地客户端 415。

[0222] 尽管图 12 中未说明，但若组合的 8 个图像块的数据速率超过指定峰值数据速率 941，则所有 8 个 x264 过程将暂时中止历时达必要的帧时间，直至已传输用于组合的 8 个图像块的数据为止。

[0223] 在一个实施例中，将客户端 415 实施为执行 FFmpeg 的 8 个实例化的 PC 上的软件。接收过程接收 8 个图像块，且将每一图像块路由至 FFmpeg 实例化，FFmpeg 实例化解压缩图像块并将其再现至显示设备 422 上的适当图像块位置。

[0224] 客户端 415 接收来自 PC 的输入设备驱动器的键盘、鼠标或游戏控制器输入并将其传输到服务器 402。服务器 402 接着应用所接收的输入设备数据并将其应用于在服务器 402 上执行的游戏或应用程序，服务器 402 为使用 Intel 2.16GHz 双核 CPU 执行 Windows 的 PC。服务器 402 接着产生新帧并经由其 DVI 输出端将新帧自以主机板为基础的图形系统或者经由 NVIDIA 8800GTX PCI Express 卡的 DVI 输出端输出。

[0225] 同时，服务器 402 经由其数字音频输出端（例如，S/PDIF）输出由游戏或应用程序产生的音频，该数字音频输出端耦合至实施视频压缩的以双四核 Xeon 为基础的 PC 上的数字音频输入端。Vorbis 开源音频压缩器用于使用可用于处理线绪的无论什么核心来与视频同时地压缩音频。在一个实施例中，完成压缩其图像块的核心首先执行音频压缩。接着将经压缩的音频连同经压缩的视频一起传输，并在客户端 415 上使用 Vorbis 音频解压缩器来解压缩经压缩的音频。

[0226] 主机服务服务器中心分配

[0227] 经由玻璃（诸如，光纤）的光以光在真空中的速度的某一部分行进，且因此可确定光在光纤中的确切传播速度。但是，在实践中，考虑用于路由延迟、传输无效率及其他耗用的时间，我们观察到互联网上的最佳延时反映较接近光速的 50％的传输速度。因此，最佳 1000 英里来回行程延时为约 22 毫秒，且最佳 3000 英里来回行程延时为约 64 毫秒。因此，一美国海岸上的单个服务器将距离过远以致不能以所要的延时伺服另一海岸上的客户端（其可能达 3000 英里远）。然而，如图 13a 中所说明，若主机服务 210 服务器中心 1300 定位于美国的中心（例如，堪萨斯州、内布拉斯加州等），以致至美国大陆中的任何点的距离为约 1500 英里或 1500 英里以下，来回行程互联网延时可低至 32 毫秒。参看图 4b，注意：尽管用户 ISP 453 所允许的最糟状况延时为 25 毫秒，但通常，在 DSL 及电缆调制解调器系统的

情况下我们观察到较接近 10-15 毫秒的延时。而且，图 4b 假定自用户场所 211 至主机代管中心 210 的最大距离为 1000 英里。因此，在所使用的典型的 15 毫秒的用户 ISP 来回行程延时及对于 32 毫秒的来回行程延时的 1500 英里的最大互联网距离的情况下，自用户致动输入设备 421 的时刻至在显示设备 422 上看见响应的总来回行程延时为 1+1+15+32+1+16+6+8 ＝ 80 毫秒。因此，通常可在 1500 英里的互联网距离上达成 80 毫秒响应时间。这将允许美国大陆中具有足够短的用户 ISP 延时 453 的任何用户场所存取在中心定位的单个服务器中心。

[0228] 在图 13b 中所说明的另一实施例中，主机服务 210 服务器中心 HS1-HS6 战略上定位于美国（或其他地理区域）的周围，特定较大的主机服务服务器中心接近高人口中心而定位（例如，HS2 及 HS5）。在一个实施例中，服务器中心 HS1-HS6 经由网络 1301 交换信息，网络 1301 可为互联网或私用网络或两者的组合。在多个服务器中心的情况下，可以较低延时向具有高用户 ISP 延时 453 的用户提供服务。

[0229] 尽管互联网上的距离的确为对经由互联网的来回行程延时有影响的因素，但有时很大程度上与延时无关的其他因素也起作用。有时经由互联网将分组流路由至距离远的位置且再次返回，从而导致来自长循环的延时。有时在路径上存在不适当操作的路由设备，从而导致传输的延迟。有时存在使路径超载的通信，其引入延迟。此外，有时，根本是存在防止用户的 ISP 路由至给定目的地的故障。因此，尽管通用互联网通常以相当可靠且最佳的路由及延时来提供从一点到另一点的连接，该相当可靠且最佳的路线及延时很大程度上是通过距离来确定（尤其是在导致路由至用户的本地区域的外部的长距离连接的情况下），但该可靠性及延时得不到任何保证且常常不可自用户的场所至通用互联网上的给定目的地而达成。

[0230] 在一个实施例中，当用户客户端 415 最初连接到主机服务 210 以玩视频游戏或使用应用程序时，客户端在启动时与可用的主机服务服务器中心 HS1-HS6 中的每一者通信（例如，使用上文所描述的技术）。若延时对于特定连接而言足够低，则使用该连接。在一个实施例中，客户端与所有主机服务服务器中心或主机服务服务器中心的一个子集通信，选择具有最低延时连接的主机服务服务器中心。客户端可选择具有最低延时连接的服务中心，或服务器中心可识别具有最低延时连接的服务器中心并将此信息（例如，以互联网地址的形式）提供给客户端。

[0231] 若特定主机服务服务器中心超载和 / 或用户的游戏或应用程序可容忍至另一、较少载入的主机服务服务器中心的延时，则可将客户端 415 重定向至另一主机服务服务器中心。在此种情形下，将使用户正执行的游戏或应用程序在用户的超载服务器中心处的服务器 402 上暂停，且将游戏或应用程序状态数据传送至另一主机服务服务器中心处的服务器 402。接着将重新开始该游戏或应用程序。在一个实施例中，主机服务 210 将等待直至游戏或应用程序达到自然暂停点（例如，游戏中的级别之间，或者在用户在应用程序中起始“保存”操作之后）才进行传送。在又一实施例中，主机服务 210 将等待直至用户活动停止历时指定时间周期（例如，1 分钟）为止且接着将在这时起始传送。

[0232] 如上所述，在一个实施例中，主机服务 210 订阅图 14 的互联网旁路服务 440 以试图将得到保证的延时提供给其客户端。如本文中所使用的互联网旁路服务是提供自互联网上的一点至另一点的具有得到保证的特性（例如，延时、数据速率等）的私用网络路线的服

务。举例而言，若主机服务 210 正使用在圣弗朗西斯科提供的 AT&T 的 DSL 服务接收来自用户的大量通信（而非路由至以 AT&T 的圣弗朗西斯科为基地的中央办公室），则主机服务 210 将在以圣弗朗西斯科为基地的中央办公室与用于主机服务 210 的服务器中心中的一或多者之间租用来自服务提供者（可能为 AT&T 本身或另一提供者）的高容量私用数据连接。接着，若自所有主机服务服务器中心 HS1-HS6 经由通用互联网至圣弗朗西斯科中使用 AT&T DSL 的用户的路线导致过高延时，则可改为使用私用数据连接。尽管私用数据连接通常比经由通用互联网的路线更昂贵，但只要其保持主机服务 210 的一小百分比连接至用户，总成本影响就低，且用户将体验到更一贯的服务体验。

[0233] 在电力故障的情况下，服务器中心常常具有两个备用电力层。第一层通常为来自电池（或来自替代的立即可用的能量源，诸如保持运转且附接至发电机的飞轮）的备用电力，其在电力干线出故障时立即提供电力且保持服务器中心运转。若电力故障为暂时的，且电力干线迅速返回（例如，在一分钟内），则电池所需的是保持服务器中心运转。但若电力故障历时较长的时间周期，则通常启动发电机（例如，柴油机供电）来取代电池且发电机只要具有燃料即可运转。所述发电机极昂贵，因为其必须能够产生多达服务器中心通常自电力干线所得到的电力。

[0234] 在一个实施例中，主机服务 HS1-HS5 中的每一者彼此共用用户数据，以便在一个服务器中心具有电力故障时，其可将在进行中的游戏及应用程序暂停，且接着将游戏或应用程序状态数据自每个服务器 402 传送至其他服务器中心处的服务器 402，且接着将通知每一用户的客户端 415 以指导其传达至新的服务器 402。假定所述情形偶尔出现，则将用户转移至不能够提供最佳延时的主机服务服务器中心（也即，用户将仅必须容忍较高延时历时电力故障的持续时间）可为可接受的，其将允许用于转移用户的宽得多的范围的选项。举例而言，给定跨越美国的时区差，则东海岸上的用户在 11:30PM 可能将要睡眠，而西海岸上的用户在 8:30PM 正开始在视频游戏使用上达到峰值。若那时西海岸上的主机服务服务器中心中存在电力故障，则其他主机服务服务器中心处可能不存在用于处理所有用户的足够的西海岸服务器 402。在此种情形下，可将一些用户转移至东海岸上具有可用服务器 402 的主机服务服务器中心，且对于用户而言的唯一后果将是较高延时。一旦将用户自失去电力的服务器中心转移，服务器中心接着就可开始其服务器及设备的有序切断，以便在电池（或其他立即电力备用）耗尽之前切断所有设备。以此方式，可避免用于服务器中心的发电机的成本。

[0235] 在一个实施例中，在主机服务 210 的严重载入的时间期间（或者由于峰值用户载入，或者因为一个或多个服务器中心出故障），基于用户正使用的游戏或应用程序的延时要求将用户转移至其他服务器中心。因此，将为使用需要低延时的游戏或应用程序的用户给出对存在有限供应的可用低延时服务器连接的优选。

[0236] 主机服务特征

[0237] 图 15 说明在以下特征描述中利用的用于主机服务 210 的服务器中心的组件的实施例。如同图 2a 中所说明的主机服务 210 一样，除非另外有条件，否则此服务器中心的组件由主机服务 210 控制系统 401 来控制及协调。

[0238] 将来自用户客户端 415 的入埠互联网通信 1501 指引至入埠路由 1502。通常，入埠互联网通信 1501 将经由至互联网的高速光纤连接而进入服务器中心，但具有足够带宽、可

靠性及低延时的任何网络连接装置将是足够的。入埠路由 1502 是网络（该网络可实施为以太网、光纤信道网络，或经由任何其他输送装置）交换器及支持所述交换器的路由服务器的系统，其取得到达的分组且将每个分组路由到适当应用程序 / 游戏服务器 1521-1525。在一实施例中，传送至特定应用程序 / 游戏服务器的分组表示自客户端所接收的数据的一子集和 / 或可由数据中心内的其他组件（例如，网络连接组件，诸如网关及路由器）来转译 / 改变。在一些状况下，（例如）若游戏或应用程序同时并行地在多个服务器上执行，则每次将分组路由至一个以上服务器 1521-1525。RAID 阵列 1511-1512 连接至入埠路由网络 1502，以使得应用程序 / 游戏服务器 1521-1525 可读取 RAID 阵列 1511-1512 及写入 RAID 阵列 1511-1512。另外，RAID 阵列 1515（其可实施为多个 RAID 阵列）也连接至入埠路由 1502，且来自 RAID 阵列 1515 的数据可自应用程序 / 游戏服务器 1521-1525 来读取。入埠路由 1502 可在多种先前技术网络架构（包括树结构的交换器，入埠互联网通信 1501 在其根部）中实施；在互连所有各种设备的网状结构中实施；或作为互连的子网络序列（互通设备当中的集中通信与其他设备当中的集中通信隔离）来实施。一类型的网络配置为 SAN（存储区域网络），其尽管通常用于储存设备，但其也可用于设备之间的通用高速数据传送。又，应用程序 / 游戏服务器 1521-1525 可各自具有至入埠路由 1502 的多个网络连接。举例而言，服务器 1521-1525 可具有至附接至 RAID 阵列 1511-1512 的子网络的网络连接及至附接至其他设备的子网络的另一网络连接。

[0239] 应用程序 / 游戏服务器 1521-1525 可经相同地、有些不同地或全部不同地来配置，如先前关于图 4a 中所说明的实施例中的服务器 402 所描述的。在一实施例中，每一用户当使用主机服务时通常为至少一应用程序 / 游戏服务器 1521-1525。出于说明的简单起见，将假定给定用户正使用应用程序 / 游戏服务器 1521，但多个服务器可由一用户使用，且多个用户可共用单一应用程序 / 游戏服务器 1521-1525。自客户端 415（如先前所描述）所发送的用户的控制输入经接收为入埠互联网通信 1501，且经由入埠路由 1502 而路由至应用程序 / 游戏服务器 1521。应用程序 / 游戏服务器 1521 使用用户的控制输入作为至在服务器上执行的游戏或应用程序的控制输入，且计算视频及与其相关联的音频的下一个帧。应用程序 / 游戏服务器 1521 接着将未经压缩的视频 / 音频 1529 输出至共用视频压缩 1530。应用程序 / 游戏服务器可经由任何装置（包括一或多个超高速以太网连接）而输出未经压缩的视频，但在一实施例中，视频是经由 DVI（交互式数字视频系统）连接而输出，且音频及其他压缩及通信信道状态信息系经由通用串列总线 (USB) 连接而输出。

[0240] 共用视频压缩 1530 压缩来自应用程序 / 游戏服务器 1521-1525 的未经压缩的视频及音频。该压缩可完全以硬件或以执行软件的硬件来实施。可存在用于每个应用程序 / 游戏服务器 1521-1525 的专用压缩器，或若压缩器足够快，则可使用给定压缩器来压缩来自一个以上应用程序 / 游戏服务器 1521-1525 的视频 / 音频。举例而言，在 60fps 下，视频帧时间为 16.67 毫秒。若压缩器能够在 1 毫秒内压缩 1 帧，则该压缩器可用于通过取得来自一个接一个的服务器的输入而压缩来自多达 16 个应用程序 / 游戏服务器 1521-1525 的视频 / 音频，该压缩器保存每个视频 / 音频压缩过程的状态且当其在来自服务器的视频 / 音频流当中循环时切换背景。这导致压缩硬件的实质成本节省。因为不同服务器将在不同时间完成帧，所以在一个实施例中，压缩器资源是处于具有用于储存每个压缩过程的状态的共用储存装置（例如，RAM，闪存）的共用集区 1530 中，且当服务器 1521-1525 帧完整且

准备被压缩时，控制装置确定那时哪个压缩资源可用，为该压缩资源提供服务器的压缩过程的状态及待压缩的未经压缩的视频 / 音频的帧。

[0241] 注意，用于每个服务器的压缩过程的状态的一部分包括关于压缩本身的信息，诸如先前帧的经解压缩的帧缓冲数据（其可用作用于 P 图像块的参考）、视频输出的分辨率；压缩的质量；图像块结构；每图像块的比特的分配；压缩质量、音频格式（例如，立体声、环绕音效、Dolby® AC-3（杜比®）AC-3）。但是压缩过程状态也包括关于以下的通信信道状态信息：峰值数据速率 941，及先前帧（如图 9b 中所说明）当前是否正被输出（且因此应忽略当前帧），及潜在地是否存在应在压缩中考虑的（诸如，过多分组丢失）影响压缩决策（例如，在 I 图像块的频率方面，等）的信道特性。因为峰值数据速率 941 或其他信道特性随着时间而改变，如由支持每个用户监视从客户端 415 发送的数据的应用程序 / 游戏服务器 1521-1525 所确定的，所以应用程序 / 游戏服务器 1521-1525 将相关信息发送至共用硬件压缩 1530。

[0242] 共用硬件压缩 1530 也使用诸如先前所描述的所述装置的装置将经压缩的视频 / 音频分组化，且在适当时，应用 FEC 码，复制特定数据，或采取其他步骤，以便充分地确保视频 / 音频数据流由客户端 415 接收且以可行的高质量及可靠性解压缩的能力。

[0243] 一些应用程序（诸如，下文所描述的应用程序）需要给定应用程序 / 游戏服务器 1521-1525 的视频 / 音频输出同时在多个分辨率下（或以其他多个格式）可用。若应用程序 / 游戏服务器 1521-1525 如此通知共用硬件压缩 1530 资源，则该应用程序 / 游戏服务器 1521-1525 的未经压缩的视频音频 1529 将被以不同格式、不同分辨率和 / 或在不同分组 / 纠错结构中同时压缩。在一些状况下，一些压缩资源可在压缩同一视频 / 音频的多个压缩过程当中共用（例如，在许多压缩算法中，存在借以在应用压缩的前将图像按比例调整成多个大小的步骤。若需要输出不同大小的图像，则该步骤可用于同时服务若干个压缩过程）。在其他状况下，对于每个格式将需要单独的压缩资源。在任何状况下，将用于给定应用程序 / 游戏服务器 1521-1525（一或多个）所需的所有各种分辨率及格式的经压缩的视频 / 音频 1539 同时输出到出埠路由 1540。在一个实施例中，经压缩的视频 / 音频 1539 的输出是处于 UDP 格式，因此其为单向分组流。

[0244] 出埠路由网络 1540 包含一系列路由服务器及交换器，该系列路由服务器及交换器将每个经压缩的视频 / 音频流经由出埠互联网通信 1599 接口（其通常将连接到至互联网的光纤接口）而指引到有意的用户或其他目的地和 / 或返回至延迟缓冲器 1515，和 / 或返回至入埠路由 1502，和 / 或经由私用网络（未图示）而输出以供进行视频分配。注意（如下所述）：出埠路由 1540 可将给定视频 / 音频流同时输出至多个目的地。在一实施例中，这是使用互联网协议 (IP) 多播来实施，其中广播意欲同时流动到多个目的地的给定 UDP 流，且该广播由出埠路由 1540 中的路由服务器及交换器来重复。广播的该多个目的地可经由互联网而至多个用户的客户端 415、经由入埠路由 1502 而到多个应用程序 / 游戏服务器 1521-1525，和 / 或到一个或多个延迟缓冲器 1515。因此，将给定服务器 1521-1522 的输出压缩成一个或多个格式，且将每个经压缩的流指引到一个或多个目的地。

[0245] 另外，在另一实施例中，若多个应用程序 / 游戏服务器 1521-1525 同时由一个用户使用（例如，在用于产生具有复杂场景的 3D 输出的并行处理配置中）且每个服务器产生所得图像的部分，则可由共用硬件压缩 1530 将多个服务器 1521-1525 的视频输出组合成

一组合帧，且自该点向前如上所述处理该组合帧，好像其来自单个应用程序 / 游戏服务器 1521-1525。

[0246] 注意，在一个实施例中，将由应用程序 / 游戏服务器 1521-1525 产生的所有视频的复本（至少以由用户观看的视频的分辨率或更高分辨率）记录于延迟缓冲器 1515 中历时至少某一数目的分钟（在一个实施例中为 15 分钟）。这允许每个用户“回放”来自每个会话的视频，以便核查先前工作或业绩（在游戏的状况下）。因此，在一个实施例中，将路由至用户客户端 415 的每个经压缩视频 / 音频输出 1539 流也多播至延迟缓冲器 1515。当将视频 / 音频储存于延迟缓冲器 1515 上时，延迟缓冲器 1515 上的目录提供应用程序 / 游戏服务器 1521-1525（其为延迟的视频 / 音频的来源）的网络地址与延迟缓冲器 1515 上可发现延迟的视频 / 音频的位置之间的交叉参考。

[0247] 现场直播的、可即刻观看的、可即刻播放的游戏

[0248] 应用程序 / 游戏服务器 1521-1525 不仅可用于执行用户的给定应用程序或视频游戏，而且其可用于建立用于支持经由主机服务 210 的导航及其他特征的主机服务 210 的用户接口应用程序。一种该用户接口应用程序的屏幕拍摄显示在图 16 中（“游戏取景器 (Game Finder)”屏幕）。该特定用户接口屏幕允许用户观看由其他用户现场玩的（或延迟的）15 个游戏。“缩略图”视频窗中的每一者（诸如，1600）为在运动中的现场直播的视频窗，其显示来自一个用户的游戏的一个视频。缩略图中所显示的视图可为用户正看的同一视图，或其可为延迟的视图（例如，若用户正玩搏斗游戏，则用户可能不希望其他用户看见其隐藏在哪里且其可选择将其游戏播放的任何视图延迟一时间周期（例如，10 分钟））。视图也可为不同于任何用户的视图的游戏的相机视图。通过菜单选择（此说明中未图示），用户可基于多种标准来选择要同时观看的游戏的选择。作为例示性选择的小取样，用户可选择游戏的随机选择（诸如图 16 中所显示的游戏）、所有一个类别的游戏（均由不同玩家来玩）、仅游戏的顶级玩家、游戏中的给定级别的玩家，或较低级玩家（例如，若玩家正学习基础）、是“搭档”（或为竞争者）的玩家、具有最多数目观看者的游戏等。

[0249] 注意，通常，每个用户将决定来自其游戏或应用程序的视频是否可由他人观看，且若如此，则决定该视频可由哪些他人观看及何时可由他人观看，决定该视频是否只可在具有延迟的情况下观看。

[0250] 产生图 16 中所显示的用户接口屏幕的应用程序 / 游戏服务器 1521-1525 通过向每个用户（该应用程序 / 游戏服务器 1521-1525 正请求来自该用户的游戏）的应用程序 / 游戏服务器 1521-1525 发送消息而获取该 15 个视频 / 音频馈送。该消息经由入埠路由 1502 或另一网络来发送。该消息将包括被请求的视频 / 音频的大小及格式，且将识别观看用户接口屏幕的用户。给定用户可选择选择“盗版”模式且不准许任何其他用户观看其游戏的视频 / 音频（自其观看点或者自另一观看点），或如先前的段中所描述，用户可选择允许观看来自其游戏的视频 / 音频，但延迟所观看的视频 / 音频。用户应用程序 / 游戏服务器 1521-1525（其接收并接受允许其视频 / 音频被观看的请求）将因此向请求服务器确认，且其也将通知共用硬件压缩 1530 需要产生被请求格式或屏幕大小（假定格式及屏幕大小不同于已经产生的格式及屏幕大小）的额外经压缩视频流，且其也将指示经压缩视频的目的地（也即，请求服务器）。若被请求的视频 / 音频仅被延迟，则请求应用程序 / 游戏服务器 1521-1525 将被这样通知，且其将通过查找延迟缓冲器 1515 上的目录中的视频 / 音频的

位置及为延迟的视频 / 音频的来源的应用程序 / 游戏服务器 1521-1525 的网络地址而自延迟缓冲器 1515 获取延迟的视频 / 音频。一旦所有该请求被产生并处理，则将高达 15 个现场直播的缩略图大小的视频流从出埠路由 1540 路由到入埠路由 1502、到产生用户接口屏幕的应用程序 / 游戏服务器 1521-1525，且将由该服务器来解压缩及显示。延迟的视频 / 音频流可能处于过大的屏幕大小，如果这样，则应用程序 / 游戏服务器 1521-1525 将解压缩所述流并将视频流按比例缩减至缩略图大小。在一个实施例中，将对音频 / 视频的请求发送到与图 4a 的主机服务控制系统类似的中央“管理”服务（图 15 中未显示）（且由中央“管理”服务来管理），中央“管理”服务接着将所述请求重定向到适当应用程序 / 游戏服务器 1521-1525。此外，在一个实施例中，可能不需要请求，因为缩略图被“推送”至允许其的那些用户的客户端。

[0251] 来自 15 个游戏的所有同时混合的音频可能产生刺耳的声音。用户可选择以此方式将所有声音混合在一起（可能就为得到由被观看的所有动作产生的“喧嚣”的感觉），或者用户可选择每次仅收听来自一个游戏的音频。单个游戏的选择是通过将黄色选择帧 1601（显示为图 16 的黑白再现中的黑色矩形轮廓线）移动到给定游戏来完成（黄色帧移动可通过使用键盘上的箭头键、通过移动鼠标、通过移动操纵杆或通过推动诸如移动电话的另一设备上的方向按钮来完成）。一旦选择了单个游戏，则只有来自该游戏的音频播放。而且，显示游戏信息 1602。在该游戏的状况下，例如，出版商标志（例如，“电子艺界公司 (Electronic Arts)”的“EA”）及游戏标志例如“极品飞车卡本峡谷”及橙色横条（在图 16 中被再现为具有垂直条纹的横条）在相对条件下指示在该特定时刻玩游戏或观看游戏的人的数目（在此状况下，许多，因此游戏为“热门”）。另外提供“状态”（即，统计），指示存在 145 个玩家正积极地玩极品飞车游戏的 80 个不同具体实例（也即，该游戏可通过个别玩家游戏或多人游戏来玩），且存在 680 个观看者（此用户是其中之一）。注意，该统计数据（及其他统计数据）由主机服务控制系统 401 来收集并储存于 RAID 阵列 1511-1512 上，以用于保持主机服务 210 操作的日志且用于适当地向用户计费并向提供内容的出版商支付费用。一些统计数据是由于由服务控制系统 401 进行的动作而记录，且一些统计数据是由个别应用程序 / 游戏服务器 1521-1525 报告给服务控制系统 401。举例而言，当游戏正被观看时（及当游戏被停止观看时），执行此游戏取景器应用程序的应用程序 / 游戏服务器 1521-1525 向主机服务控制系统 401 发送消息，以使得主机服务控制系统 401 可更新多少个游戏处于观看中的统计数据。一些统计数据可为用户接口应用程序（诸如，此游戏取景器应用程序）所用。

[0252] 若用户单击其输入设备上的启动按钮，则在继续播放现场直播视频的同时，其将看见黄色帧中的缩略图视频放大为全屏幕大小。该效果显示在图 17 中的过程中。注意，视频窗 1700 的大小增大。为了实施该效果，应用程序 / 游戏服务器 1521-1525 从运行选定的游戏的应用程序 / 游戏服务器 1521-1525 请求具有路由到其的游戏的全屏幕大小（以用户的显示设备 422 的分辨率）的视频流的复本。执行游戏的应用程序 / 游戏服务器 1521-1525 通知共用硬件压缩器 1530 不再需要游戏的缩略图大小的复本（除非另一应用程序 / 游戏服务器 1521-1525 需要这种缩略图），且接着其指引共用硬件压缩器 1530 将视频的全屏幕大小的复本发送至放大视频的应用程序 / 游戏服务器 1521-1525。玩该游戏的用户可或可不具有分辨率与将游戏放大的用户的所述显示设备的分辨率相同的显示设备 422。另外，游

戏的其他观看者可或可不具有分辨率与将游戏放大的用户相同的显示设备 422(且可具有不同的音频回放装置,例如,立体声或环绕音效)。因此,共用硬件压缩器 1530 确定是否已经产生满足请求视频 / 音频流的用户的要求的合适的经压缩视频 / 音频流,且若合适的经压缩视频 / 音频流确实存在,则共用硬件压缩器 1530 通知出埠路由 1540 将该流的复本路由至放大该视频的应用程序 / 游戏服务器 1521-1525,且若合适的经压缩视频 / 音频流不存在,则压缩视频的适合于所述用户的另一复本并指导出埠路由将该流发送回至入埠路由 1502 及放大该视频的应用程序 / 游戏服务器 1521-1525。现在接收选定视频的全屏幕版本的该服务器将解压缩该全屏幕版本并将其逐渐地按比例放大至全大小。

[0253] 图 18 说明在将游戏完全放大至全屏幕且以用户的显示设备 422 的全分辨率显示游戏之后屏幕看起来如何(如通过箭头 1800 指向的图像所指示的)。执行游戏取景器应用程序的应用程序 / 游戏服务器 1521-1525 向提供缩略图的其他应用程序 / 游戏服务器 1521-1525 发送消息以指示所述缩略图不再需要且向主机服务控制服务器 401 发送消息以指示不再观看其他游戏。此时,产生的唯一显示为屏幕顶部的叠加层 (overlay)1801,其将信息及菜单控制提供给用户。注意,随着该游戏进展,观众增长至 2,503 个观看者。在如此多的观看者的情况下,必然存在具有显示设备 422 的许多观看者,所述显示设备 422 具有相同或接近相同的分辨率(每个应用程序 / 游戏服务器 1521-1525 具有按比例调整视频以用于调整配合度的能力)。

[0254] 因为所显示的游戏为多人游戏,所以用户可决定在某时刻加入该游戏。由于多种原因,主机服务 210 可或可不允许用户加入该游戏。举例而言,用户可能必须支付玩游戏的费用而选择不支付,用户可能不具有足以加入所述特定游戏的足够等级(例如,对于其他玩家而言,其将不有竞争性),或者用户的互联网连接可能不具有足以允许用户玩的足够低的延时(例如,不存在用于观看游戏的延时约束,因此可在无延时关注的情况下观看被远距离地(实际上,在另一大陆上)玩的游戏,但对于待玩的游戏而言,延时必须足够低以使用户 (a) 享受该游戏,且 (b) 处于与可能具有较低延时连接的其他玩家相等的地位)。若准许用户玩,则为用户提供游戏取景器用户接口的应用程序 / 游戏服务器 1521-1525 将请求主机服务控制服务器 401 初始化(也即,定位并启动)被合适地配置以用于播放特定游戏的应用程序 / 游戏服务器 1521-1525 从而从 RAID 阵列 1511-1512 载入该游戏,且接着主机服务控制服务器 401 将指导入埠路由 1502 将来自用户的控制信号传送至现在对游戏进行主机代管的应用程序 / 游戏服务器且现在对游戏进行主机代管的应用程序 / 游戏服务器将指导共用硬件压缩 1530 自压缩来从主机代管游戏取景器应用程序的应用程序 / 游戏服务器的视频 / 音频切换至压缩来自现在对游戏进行主机代管的应用程序 / 游戏服务器的视频 / 音频。游戏取景器应用程序 / 游戏服务与对游戏进行主机代管的新的应用程序 / 游戏服务器的垂直同步并不同步,且因此在该两个同步之间可能存在时间差。因为共用视频压缩硬件 1530 将在应用程序 / 游戏服务器 1521-1525 完成视频帧之后即开始压缩视频,所以来自新服务器的第一帧可比旧服务器的全帧时间完成得早,来自新服务器的第一帧可能在先前经压缩的帧完成其传输之前(例如,考虑图 9b 的传输时间 992 :若未经压缩的帧 3 963 早一帧时间的一半地完成,则其将影响 (impinge) 传输时间 992)。在这种情形下,共用视频压缩硬件 1530 将忽略来自新服务器的第一帧(例如,如忽略 (974) 帧 4 964),且客户端 415 将来自旧服务器的最末帧保持额外一帧时间,且共用视频压缩硬件 1530 将开始压缩来

自对游戏进行主机代管的新应用程序 / 游戏服务器的下一帧时间视频。对于用户而言，在视觉上，从一个应用程序 / 游戏服务器至另一应用程序 / 游戏服务器的转变将是无缝的。主机服务控制服务器 401 接着将通知对游戏取景器进行主机代管的应用程序 / 游戏服务器 1521-1525 切换至闲置状态，直至再次需要其为止。

[0255] 用户接着能够玩该游戏。此外，例外的是游戏将在感知上即刻地播放（因为游戏已被以十亿比特 / 秒速度从 Raid 阵列 1511-1512 载入到应用程序 / 游戏服务器 1521-1525 上），且将通过理想的驱动器、暂存器配置（在 Windows 的状况下）将游戏连同被正确配置以用于该游戏的操作系统一起载入到准确适合于该游戏的服务器上，且没有可能与该游戏的操作竞争的其他应用程序在该服务器上执行。

[0256] 而且，随着用户在游戏中进展，游戏的片段中的每一者将以十亿比特 / 秒的速度（也即，在 8 秒内载入十亿字节）从 RAID 阵列 1511-1512 载入服务器中，且由于 RAID 阵列 1511-1512 的巨大储存容量（因为其为许多用户的共用资源，所以其可能非常大，但仍具成本效益），使得可预先计算几何形状设置或其他游戏片段设置并将其储存于 RAID 阵列 1511-1512 上且极快速地进行载入。此外，因为每个应用程序 / 游戏服务器 1521-1525 的硬件配置及计算能力是已知的，所以可预先计算像素及顶点着色。

[0257] 因此，游戏可几乎即刻启动，其将在理想环境中执行，且随后的片段将几乎即刻载入。

[0258] 但是，除这些优点之外，用户将能够观看他人玩游戏（经由先前所描述的游戏取景器，及其他装置），且两者均决定游戏是否有趣，如果这样，则自观看他人来学习技巧。此外，用户将能够即刻地演示该游戏，而不必等待大的下载和 / 或安装，且用户将能够即刻玩该游戏（可能在较小费用的试用基础上，或在较长期的基础上）。此外，用户将能够通过足够低延时的无线连接（虽然对于仅观看而言，延时不是一个问题）而在 Windows PC、Macintosh 上、在电视机上、在家里、在行进时且甚至在移动电话上玩该游戏。此外，这均可在并非曾经实体拥有游戏复本的情况下完成。

[0259] 如先前所叙述，用户可决定不允许其游戏播放可被他人观看，允许其游戏可在延迟之后观看，允许其游戏可被选定用户观看，或允许其游戏可被所有用户观看。不管怎样，在一个实施例中，将视频 / 音频储存于延迟缓冲器 1515 中历时 15 分钟，且用户将能够“回倒”并观看其先前的游戏播放，且将游戏暂停，将游戏缓慢地回放，将游戏快进等，正如其在观看具有数字视频记录器 (DVR) 的 TV 时所能够进行的。尽管在该实例中，用户在玩游戏，但若用户正使用应用程序，则相同“DVR”能力是可用的。这在核查先前工作中及在如下详述的其他应用中可是有用的。另外，若游戏经设计为具有基于利用游戏状态信息而回倒的能力，以便可改变相机视图等，则也将支持此“3D DVR”能力，但其将需要将游戏设计为支持“3D DVR”能力。使用延迟缓冲器 1515 的“DVR”能力将连同任何游戏或应用程序（当然，限于在使用游戏或应用程序时所产生的视频）一起起作用，但在具有 3D DVR 能力的游戏的状况下，用户可控制先前所播放的片段的 3D“飞越 (fly-through)”，且使延迟缓冲器 1515 记录所得视频并记录游戏片段的游戏状态。因此，将特定“飞越”记录为经压缩的视频，但因为也将记录游戏状态，所以不同的飞越将可能在游戏的同一片段的稍后日期。

[0260] 如下所述，主机服务 210 上的用户将各自具有一个用户页面，在该用户页面中，用户可公布关于其本身的信息及其他数据。用户将能够公布的事情之一为来自其已保存的游

戏播放的视频片段。举例而言，若用户已克服游戏中的特别困难的挑战，则用户可刚好“回倒”到其在游戏中获得其大成果的地点之前，且接着指导主机服务 210 将某一持续时间（例如，30 秒）的视频片段保存在用户的用户页面上以供其他用户观看。为了实施这些，用户正使用的应用程序 / 游戏服务器 1521-1525 仅要做的事情是将储存于延迟缓冲器 1515 中的视频回放至 RAID 阵列 1511-1512 且接着在用户的用户页面上给所述视频片段编索引。

[0261] 若游戏具有 3D DVR 的能力，如上所述，则也可由用户来记录 3D DVR 所需的游戏状态信息且使其为用户的用户页面可用。

[0262] 在游戏经设计为除具有活跃玩家外还具有“旁观者”（也即，能够在不参与的情况下在 3D 世界行进并观察到动作的用户）的情况下，则游戏取景器应用程序将使用户能够作为旁观者以及玩家加入游戏。从观看的实施点看，对于主机系统 210 而言，用户是旁观者而不是活跃玩家是不存在差异的。将游戏载入应用程序 / 游戏服务器 1521-1525 上且用户将控制该游戏（例如，控制观看世界的虚拟相机）。唯一的差异是用户的游戏体验。

[0263] 多个用户合作

[0264] 主机服务 210 的另一特征是多个用户在观看现场直播的视频的同时合作的能力（即使使用迥然不同的设备来观看也如此）。当玩游戏时及当使用应用程序时，这都有用。

[0265] 许多 PC 及移动电话装备有视频相机且具有进行实时视频压缩的能力（尤其当图像小时）。而且，小相机是可用的，可附接到电视，且以软件或使用用于压缩视频的许多硬件压缩设备中的一者来实施实时压缩并不困难。而且，许多 PC 及所有移动电话具有麦克风，且耳机在具有麦克风情况下可用。

[0266] 组合有本地视频 / 音频压缩能力（具体来说，使用本文中所描述的低延时视频压缩技术）的所述相机和 / 或麦克风将使用户能够将视频和 / 或音频连同输入设备控制数据一起从用户场所 211 传输到主机服务 210。当使用所述技术时，则可实现图 19 中所说明的能力：用户可使其视频及音频 1900 出现在另一用户的游戏或应用程序内的屏幕上。该实例为多人游戏，其中队友在赛车中合作。用户的视频 / 音频仅可被其队友选择性地观看 / 听到。此外，因为使用上文所描述的技术将有效地不存在延时，所以玩家将能够实时地彼此谈话或进行运动而没有可感知的延迟。

[0267] 该视频 / 音频整合是通过使来自用户的相机 / 麦克风的经压缩视频和 / 或音频作为入埠互联网通信 1501 到达而完成。接着，入埠路由 1502 将该视频和 / 或音频路由到被准许观看 / 听到视频和 / 或音频的应用程序 / 游戏服务器 1521-1525。接着，选择使用视频和 / 或音频的各别应用程序 / 游戏服务器 1521-1525 的用户解压缩视频和 / 或音频且视需要而将其整合以出现于游戏或应用程序内，诸如通过 1900 所说明的。

[0268] 图 19 的实例显示如何在游戏中使用该合作，但该合作可为用于应用程序的极其强大的工具。考虑一各情形：其中一大建筑物正由在芝加哥的建筑师为以纽约为基地的房地产开发商为纽约市设计，但该决策涉及在旅游中且碰巧在迈阿密机场的财务投资者，且需要进行关于建筑物的特定设计要素（在其如何搭配其附近的建筑物方面）的决策，以满足投资者与房地产开发商两者。假定建筑公司在芝加哥具有具有附接到 PC 的相机的高分辨率监视器，房地产开发商在纽约具有带相机的笔记本计算机，且投资者在迈阿密具有带相机的移动电话。建筑公司可使用主机服务 210 来对能够进行高度逼真 3D 再现的强大的建筑设计应用程序进行主机代管，且其可利用纽约市的建筑物的大数据库，以及正设计的

建筑物的数据库。建筑设计应用程序将在应用程序 / 游戏服务器 1521-1525 中的一者上（或若其需要大量计算能力，则在若干者上）执行。处于全异 (disparate) 位置处的 3 个用户中的每一者将连接至主机服务 210，且每一者将具有对建筑设计应用程序的视频输出的同时观看，但其将被针对每个用户具有的给定设备及网络连接特性而由共用硬件压缩 1530 适当地设定大小（例如，建筑公司可经由 20Mbps 商用互联网连接看见 2560×1440 60fps 显示，纽约的房地产开发商可经由其笔记本计算机上的 6Mbps DSL 连接看见 1280×720 60fps 图像，且投资者可经由其移动电话上的 250Kbps 蜂窝式数据连接看见 320×180 60fps 图像）。每一方将听到其他方的语音（将通过应用程序 / 游戏服务器 1521-1525 中的许多广泛可用的会议呼叫套装软件中的任一者来处理会议呼叫），且经由用户输入设备上的按钮的致动，用户将能够使用其本地相机使视频出现。随着会议进行，建筑师将能够通过极具照片般逼真感的 3D 再现显示当其使建筑物旋转且使其邻接该区域中的另一建筑物飞越 (fly-by) 时建筑物看起来像什么，且所有方均将在各方的显示设备的分辨率下可见到相同视频。由任何方使用的本地设备中的任一者均能够以该真实感处理 3D 动画不是问题，更不用说下载或甚至储存再现纽约市的周围建筑物所需的巨大数据库。自用户中的每一者的观点看，尽管距离很远，且尽管是全异本地设备，但其将简单地在难以置信的真实感程度下具有无缝体验。此外，当一方希望其面部被看来较佳地传达其情绪状态时，其可如此进行。另外，若房地产开发商或投资者希望控制建筑程序且使用其自身的输入设备（其为键盘、鼠标、小键盘或触摸屏幕），则其可如此，且其可以用无感知的延时来响应（假定其网络连接不具有不合理的延时）。举例而言，在移动电话的状况下，若移动电话连接至机场的 WiFi 网络，则其将具有非常低的延时。但若其使用美国现今可用的蜂窝式数据网络，则其很可能将遭受引人注意的滞后。但是，对于会议的大多数目的（其中投资者正观看建筑师控制建筑物飞越或正谈论视频电话会议），甚至蜂窝式延时也应是可接受的。

[0269] 最后，在合作性会议呼叫结束时，房地产开发商及投资者将进行其评论且从主机服务停播，建筑公司将能够“回倒”已记录在延迟缓冲器 1515 上的会议的视频且核查在会议期间进行的应用于建筑物的 3D 模型的评论、面部表情和 / 或动作。若存在其希望保存的特定片段，则可将视频 / 音频的所述片段自延迟缓冲器 1515 移动至 RAID 阵列 1511-1512 以用于档案储存及稍后回放。

[0270] 而且，从成本观点看，若建筑师仅需要使用纽约市的计算能力及大数据库历时 15 分钟的会议呼叫，则其仅需要支付所述资源被使用的时间的费用，而不必拥有高能力的工作台且不必购买大数据库的昂贵复本。

[0271] 视频丰富的社区服务

[0272] 主机服务 210 使得有空前机会来在互联网上建立视频丰富的社区服务。图 20 显示用于主机服务 210 上的游戏玩家的例示性用户页面。如同游戏取景器应用程序一样，用户页面为在应用程序 / 游戏服务器 1521-1525 中的一者上执行的应用程序。该页面上的所有缩略图及视频窗显示恒定地移动的视频（若片段短，则其循环）。

[0273] 使用视频相机或通过上传视频，用户（其用户名为“KILLHAZARD”）能够公布其本身的视频 2000（其他用户可观看该视频）。该视频储存于 RAID 阵列 1511-1512 上。而且，当其他用户来到 KILLHAZARD 的用户页面时，若 KILLHAZARD 此时正使用主机服务 210，则将显示其正进行的无论什么的现场直播的视频 2001（假定 KILLHAZARD 准许观看其用户页面

的用户观看该视频）。这将由对用户页面应用程序进行主机代管的应用程序 / 游戏服务器 1521-1525 从服务控制系统 401 请求 KILLHAZARD 是否为活跃的（且若如此，则请求其正使用的应用程序 / 游戏服务器 1521-1525）来完成。接着，使用由游戏取景器应用程序使用的相同方法，将合适分辨率及格式的经压缩的视频流发送至执行用户页面应用程序的应用程序 / 游戏服务器 1521-1525 且将其显示。若用户选择具有 KILLHAZARD 的现场直播的游戏播放的窗口且接着适当地单击其输入设备，则该窗口将放大（再次使用与游戏取景器应用程序相同的方法），且现场直播的视频将以观看用户的显示设备 422 的分辨率（适合于观看用户的互联网连接的特性）填充屏幕。

[0274] 这优于先前技术方法的关键优点是：观看用户页面的用户能够看见用户不拥有的现场直播地播放的游戏，且可不具有能够玩该游戏的本地计算机或游戏控制台。其为用户提供看用户页面中显示为“活动中”的用户玩游戏的极好机会，且这是了解观看用户可能希望尝试或较擅长的游戏的机会。

[0275] 来自 KILLHAZARD 的搭档 2002 的相机记录的或上传的视频剪辑也显示于用户页面上，且每个视频剪辑的下方为指示该搭档是否在线玩游戏的文字（例如，six_shot 正玩游戏“龙骑士 (Eragon)”（在此显示为 Game4）且 MrSnuggles99 离线等）。通过单击菜单项（未图示），搭档视频剪辑自显示已记录的或上传的视频切换至当前正玩主机服务 210 上的游戏的搭档在所述时刻在其游戏中正在进行的内容的现场直播的视频。因此，其变成为搭档分群的游戏取景器。若选择搭档的游戏且用户单击该游戏，则该游戏将放大至全屏幕，且用户将能够观看全屏幕现场直播地播放的游戏。

[0276] 再次，观看搭档的游戏的用户不拥有游戏的复本，也不拥有用于玩该游戏的本地计算 / 游戏控制台资源。游戏观看是有效瞬时的。

[0277] 如上文先前所描述，当用户玩主机服务 210 上的游戏时，用户能够“回倒”游戏且发现其希望保存的视频片段，且接着将该视频片段保存到其用户页面。这被称为“自赏剪辑 (Brag Clip)”。视频片段 2003 都是由 KILLHAZARD 从其所玩的先前游戏保存的自赏剪辑 2003。数字 2004 显示自赏剪辑已被观看多少次，及自赏剪辑何时被观看，用户具有对其评定等级的机会，且橙色（显示为黑色轮廓线）钥匙孔形状的图示 2005 的数目指示等级是多高。当用户观看用户页面时，自赏剪辑 2003 连同页面上的其余视频一起恒定地循环。若用户选择并单击自赏剪辑 2003 中的一者，则其放大以呈现自赏剪辑 2003，以及允许播放、暂停、回倒、快进、步进等该剪辑的 DVR 控制。

[0278] 自赏剪辑 2003 回放是通过应用程序 / 游戏服务器 1521-1525 载入用户记录自赏剪辑时储存于 RAID 阵列 1511-1512 上的经压缩视频片段且将其解压缩并将其回放来实施。

[0279] 自赏剪辑 2003 也可为来自支持 3D DVR 能力的游戏的“3DDVR”视频片段（也即，来自可被重放且允许用户改变相机观看点的游戏的游戏状态序列）。在此状况下，除用户在记录游戏片段时进行的特定“飞越”的经压缩视频记录之外，也储存游戏状态信息。当用户页面正被观看且所有缩略图及视频窗均恒定地循环时，3D DVR 自赏剪辑 2003 将使在用户记录游戏片段的“飞越”时记录为经压缩视频的自赏剪辑 2003 恒定地循环。但是，当用户选择 3D DVR 自赏剪辑 2003 并单击 3D DVR 自赏剪辑 2003 时，除允许播放经压缩视频自赏剪辑的 DVR 控制之外，用户将能够单击给出其用于游戏片段的 3D DVR 能力的按钮。其将能够独立地控制游戏片段期间的相机“飞越”，且若其希望（且拥有用户页面的用户允许如

此），则其将能够以经压缩的视频的形式记录替代性自赏剪辑“飞越”，替代性自赏剪辑“飞越”将接着可为用户页面的其他观看者所用（立即地，或者在用户页面的拥有者具有核查自赏剪辑的机会之后）。

[0280] 该 3D DVR 自赏剪辑 2003 能力是通过启动将要在另一应用程序 / 游戏服务器 1521-1525 上重放已记录的游戏状态信息的游戏来启用。因为游戏可被几乎瞬时地启动（如先前所描述），所以启动其（其播放限于由自赏剪辑片段记录的游戏状态）且接着允许用户在将经压缩视频记录到延迟缓冲器 1515 的同时用相机进行“飞越”并不困难。一旦用户完成进行“飞越”，则将游戏撤销启动。

[0281] 从用户的观点看，启动具有 3D DVR 自赏剪辑 2003 的“飞越”并不比控制线性自赏剪辑 2003 的 DVR 控制难。用户可不知道该游戏或甚至不知道如何玩该游戏。用户指示盯着看另一操作者记录的游戏片段期间的 3D 世界的虚拟相机操作者。

[0282] 用户将也能够将其自身的音频加录于自赏剪辑上（或者自麦克风记录或者上传）。以这种方式，可使用自赏剪辑来使用来自游戏的人物及动作产生定制动画。此动画制作技术通常被称为“游戏电影 (machinima)”。

[0283] 随着用户在游戏中进展，其将达成不同技能级别。所播放的游戏将成果报告给服务控制系统 401，且所述技能级别也将显示于用户页面上。

[0284] 互动式动画广告

[0285] 在线广告已自文字转变至静态图像、视频，且现在转变至互动式片段，通常使用如 Adobe Flash 的动画精简型客户端来实施。使用动画精简型客户端的原因在于：用户通常对于因向其推销产品或服务的特权而被延迟较无耐心。而且，精简型客户端在非常低性能的 PC 上执行，且因此广告商可具有高度信心：互动式广告将适当地工作。很遗憾地，诸如 Adobe Flash 的动画精简型客户端在互动性的程度及体验（以减少下载时间，且可用于几乎所有的用户设备，包括不具有 GPU 或高性能 CPU 的低性能 PC 及苹果电脑）的持续时间上受限制。

[0286] 图 21 说明一个互动式广告，其中用户将在汽车在陈列室中旋转时选择汽车的外部及内部色彩，同时实时射线追踪显示汽车看起来如何。接着用户选择角色来驾驶汽车，且接着用户可采用该汽车来用于在竞赛轨道上或者穿过诸如摩纳哥的外国场所驾驶。用户可选择较大引擎或较佳轮胎，且接着可看见改变的配置如何影响汽车加速或保持稳定的能力。

[0287] 当然，广告有效地的为尖端的 3D 视频游戏。但对于可在 PC 或视频游戏控制台上播放的这种广告，其将需要可能 100MB 下载，且在 PC 的状况下，其可能需要安装特殊驱动器，且可能在 PC 缺乏足够 CPU 或 GPU 计算能力时根本不执行。因此，所述广告在先前技术配置中不切实际。

[0288] 在主机服务 210 中，所述广告几乎即刻地投放，且较佳地执行，无论用户的客户端 415 能力如何。因此，其比精简型客户端互动式广告更迅速地投放，体验上更加丰富，且高度可靠。

[0289] 实时动画期间流动的几何形状

[0290] RAID 阵列 1511-1512 及入埠路由 1502 可提供如此快的数据速率且具有如此低的延时，以致有可能设计依赖于 RAID 阵列 1511-1512 及入埠路由 1502 来在实时动画（例如，

具有复杂数据库的飞越）期间于游戏播放的中间或应用程序中可靠地直接传送几何形状的视频游戏及应用程序。

[0291] 在先前技术的系统（诸如，图 1 中所显示的视频游戏系统）下，可用的大量储存设备（尤其是在实用的家庭设备中）极缓慢以致不能在游戏播放期间流动几何形状（除了所需的几何形状稍微可预测的情形之外）。举例而言，在存在指定道路的驾驶游戏中，可合理地适当预测用于进入视野内的建筑物的几何形状且大量储存设备可提前搜寻即将到来的几何形状所定位的位置。

[0292] 但在具有不可预测的改变的复杂场景中（例如，在周围具有复杂人物的战役场景中），若 PC 或视频游戏系统上的 RAM 完全被填满用于当前在视图中的对象的几何形状，且接着用户突然将其人物转向以观看其人物之后是什么，若未将几何形状预先载入 RAM 中，则可能在可显示几何形状之前存在延迟。

[0293] 在主机服务 210 中，RAID 阵列 1511-1512 可以超过超高速以太网速度的速度使数据流动，且在 SAN 网络下，有可能达到优于 10 个十亿比特以太网或优于其他网络技术的 100 亿比特 / 秒的速度。100 亿比特 / 秒将在小于一秒内载入十亿字节的数据。在 60fps 帧时间（16.67 毫秒）内，可载入约 170 百万比特 (21MB) 的数据。当然，甚至在 RAID 配置中，旋转媒体也仍将导致大于一帧时间的延时，但以闪存为基础的 RAID 储存器最终将与旋转媒体 RAID 阵列一般大且将不会招致该高延时。在一各实施例中，使用经由大量 RAM 写入的高速缓存来提供非常低延时的存取。

[0294] 因此，在足够高的网络速度，以及足够低延时的大量储存器下，可与 CPU 和 / 或 GPU 可处理 3D 数据一般快地将几何形状流动到应用程序 / 游戏服务器 1521-1525 中。因此，在先前所给出的实例中，其中用户突然将其人物转向且向后看，可在人物完成旋转之前载入其身后的所有人物的几何形状，且因此，对于用户而言，将看来似乎其处于与现场直播的动作一般真实的照片般逼真的世界中。

[0295] 如先前所论述，照片般逼真的计算机动画中的最后的边界中的一者是人脸，且由于人眼对于不完全性的敏感性，来自照片般逼真的面部的最轻微错误可导致来自观看者的负面反应。图 22 显示使用 Contour™ 真实性捕获技术（以下共同待审申请中的申请的主题：2004 年 9 月 15 日申请的第 10/942,609 号“Apparatus and method for capturing the motion of a performer”；2004 年 9 月 15 日申请的第 10/942,413 号“Apparatus and method for capturing the expression of a performer”；2005 年 2 月 25 日申请的第 11/066,954 号“Apparatus and method for improving marker identification within a motion capture system”；2005 年 3 月 10 日申请的第 11/077,628 号“Apparatus and method for performing motion capture using shutter synchronization”；2005 年 10 月 20 日申请的第 11/255,854 号“Apparatus and method for performing motion capture using a random pattern on capture surfaces”；2006 年 6 月 7 日申请的第 11/449,131 号“System and method for performing motion capture using phosphor application techniques”；2006 年 6 月 7 日申请的第 11/449,043 号“System and method for performing motion capture by strobing a fluorescent lamp”；2006 年 6 月 7 日申请的第 11/449,127 号“System and method for three dimensional capture of stop-motion animated characters”，上述申请中的每一者都被转让给本 CIP 申请的受让

人）捕获的现场直播的表演如何导致非常平滑的捕获表面，既而达成高多边形计数的追踪表面（也即，多边形运动精确地追随面部的运动）。最后，当将现场直播的表演的视频映射于追踪表面上以产生纹理表面时，产生照片般逼真的结果。

[0296] 尽管当前 GPU 技术能够再现追踪表面及纹理中的许多多边形且实时地照明该表面，但若多边形及纹理每一帧时间改变（其将产生最具照片般逼真感的结果），则其将迅速地消耗现代 PC 或视频游戏控制台的所有可用 RAM。

[0297] 使用上文所描述的流动几何形状技术，将几何形状不断地馈送至应用程序 / 游戏服务器 1521-1525 中以使得其可不断地动画制作照片般逼真的面部从而允许产生具有几乎不能区别于现场直播的动作面部的面部的视频游戏变得实际。

[0298] 线性内容与互动式特征的整合

[0299] 电影、电视节目及音频材料（统称“线性内容”）广泛地以许多形式可用于家庭及办公室用户。线性内容可在如 CD、DVD、及蓝光媒体的实体媒体上获取。其也可通过来自卫星及电缆 TV 广播的 DVR 来记录。此外，其可以经由卫星及电缆 TV 的即付即看 (PPV) 内容及以电缆 TV 上的视频点播 (VOD) 可用。

[0300] 日益增加的线性内容可经由互联网以下载的内容及流动内容可用。现今，确实不存在一个能体验与线性媒体相关的所有特征的位置。举例而言，DVD 及其他视频光学媒体通常具有在其他位置处不可用的互动式特征（如导演的评论、“花絮”短片等）。在线音乐站点具有通常在 CD 上不可用的封面艺术及歌曲信息，但并非所有 CD 在线可用。且与电视节目相关联的网站常常具有额外特征、博客及有时来自演员或创作人员的评论。

[0301] 另外，在许多电影或运动事件的情况下，通常有经常连同线性媒体一起发行（在电影的状况下）或（在运动的状况下）可以被紧密地联系到真实世界事件（例如，玩家的交易）的视频游戏。

[0302] 主机服务 210 非常适合于在将全异形式的相关内容连结在一起时传送线性内容。的确，传送电影不如传送高度互动式视频游戏有挑战，且主机服务 210 能够将线性内容传送至家庭或办公室中的多种设备，或传送至移动设备。图 23 显示用于主机服务 210 的例示性用户接口页面，其显示线性内容的选择。

[0303] 但是，不同于大多数线性内容传送系统，主机服务 210 还能够传送相关的互动式成份（例如，DVD 上的菜单及特征、HD-DVD 上的互动式叠加层，及网站上的 Adobe Flash 动画（如下文所述））。因此，客户端设备 415 限制不再引入哪些特征可用的限制。

[0304] 另外，主机代管系统 210 能够动态且实时地将线性内容与视频游戏内容连结在一起。举例而言，若用户正观看哈利波特电影中的 Quidditch（魁地奇）比赛，且决定其愿意尝试玩 Quidditch，则其可仅仅单击按钮且电影将暂停且其将被立即输送到哈利波特视频游戏的 Quidditch 片段。在玩 Quidditch 比赛的后，另一次单击按钮，且电影将即刻重新开始。

[0305] 在照片般逼真的图形及制作技术的情况下，其中摄影捕获的视频不能区别于现场直播的动作人物，当用户进行自现场直播的动作电影中的 Quidditch 游戏至主机服务上的视频游戏中的 Quidditch 游戏的转变时（如本文中所述），该两个场景实际上不能区别。此为线性内容与互动式（例如，视频游戏）内容两者的导演提供全新的创作选项，因为该两个世界之间的线变得不能区别。

[0306] 利用图 14 中所显示的主机服务架构，可将 3D 电影中的虚拟相机的控制提供给观看者。举例而言，在发生于列车内的场景中，将有可能允许观看者在故事进展时控制虚拟相机且环顾列车。此假定列车中的所有 3D 对象（“资产”）以及能够实时地再现所述场景以及原始电影的足够计算能力水平可用。

[0307] 且甚至对于非计算机产生的娱乐，存在可提供的非常刺激的互动式特征。举例而言，2005 电影“傲慢与偏见”具有装饰华丽的旧英国大厦中的许多场景。对于特定大厦场景，用户可将视频暂停且接着控制相机以巡视大厦，或可能的周围区域。为了实施这些，可携带具有鱼眼镜头的相机穿过大厦，当其追踪其位置时，非常类似于实施先前技术 Apple（苹果）公司的 QuickTime VR。各种帧接着将被转换，因此图像不失真，且接着其连同电影一起被储存于 RAID 阵列 1511-1512 上，且在用户选择继续虚拟巡视时被回放。

[0308] 在运动事件的情况下，可经由主机服务 210 来使现场直播的运动事件（诸如，篮球比赛）流动以供用户观看（如同其对于常见 TV 所想要的那样）。在用户观看特定播放之后，游戏的视频游戏（最终篮球玩家看起来与真实玩家一般照片般逼真）可赶上在同一位置中开始的玩家，且用户（可能各自控制一玩家）可重新玩以观看其是否可比所述玩家做得更佳。

[0309] 本文中所描述的主机服务 210 极其适合于支持此未来世界，因为其能够承受不切实际以致不能安装于家庭中或大多数办公室背景中的计算能力及大容量储存资源，而且其计算资源总是最新的（在可用的最新的计算硬件的情况下），但是在家庭背景中，将总是存在具有较旧代的 PC 及视频游戏的家庭。此外，在主机服务 210 中，用户被隐瞒所有此计算复杂度，因此，即使用户可能正使用非常尖端的系统，自用户的观点看，也如改变电视上的信道一般简单。另外，用户将能够存取所有计算能力及计算能力将自任何客户端 415 带来的体验。

[0310] 多人游戏

[0311] 对于游戏为多人游戏的程度，则游戏将能够不仅经由入埠路由 1502 网络传达到应用程序 / 游戏服务器 1521-1525，而且通过网络桥接器传达至具有不在主机服务 210 中执行的服务器或游戏机器的互联网（未图示）。当通过通用互联网上的计算机玩多人游戏时，则应用程序 / 游戏服务器 1521-1525 将具有极快访问互联网的益处（与游戏在家庭中的服务器上执行的情况相比），但其将受在较缓慢连接上玩游戏的其他计算机的能力限制，且也潜在地受互联网上的游戏服务器被设计以适应最少共同点 (common denominator)（所述游戏服务器是相对较慢的消费者互联网连接上的家庭计算机）的事实限制。

[0312] 但当完全在主机服务 210 服务器中心内玩多人游戏时，则可达到极大差异。对用于用户的游戏进行主机代管的每个应用程序 / 游戏服务器 1521-1525 将与其他应用程序 / 游戏服务器 1521-1525 以及用极高速度、极低延时连接性及巨大、非常快的储存阵列对针对多人游戏的中央控制进行主机代管的任何服务器互连。举例而言，若超高速以太网用于入埠路由 1502 网络，则应用程序 / 游戏服务器 1521-1525 将在彼此当中传达，且传达到以十亿比特 / 秒速度在潜在的仅 1 毫秒或 1 毫秒以下的延时下对针对多人游戏的中央控制进行主机代管的任何服务器。另外，RAID 阵列 1511-1512 将能够非常快速地响应且接着以十亿比特 / 秒速度传送数据。作为一个实例，若用户在外表及服装方面定制人物，以使得人物具有对于人物而言唯一的大量几何形状及行为，在限于在家庭中在 PC 或游戏控制台上执

行的游戏客户端的先前技术系统下，若所述人物将进入另一用户的视野中，则用户将必须等待直到长的缓慢下载完成为止，以便将所有几何形状及行为数据载入其计算机中。在主机服务 210 内，所述相同下载可优于以十亿比特 / 秒速度从 RAID 阵列 1511-1512 服务的超高速以太网。即使家庭用户具有 8Mbps 互联网连接（其根据现今的标准来看极快），超高速以太网也快 100 倍。因此，在快的互联网连接上花费一分钟进行的工作在超高速以太网上将花费小于一秒。

[0313] 顶级玩家分群及锦标赛

[0314] 主机服务 210 极其适合于锦标赛。因为无游戏在本地客户端中执行，所以不存在用户作弊的机会（例如，在现有技术的锦标赛中，他们可能通过修改运行于他们本地 PC 上的游戏副本来给予他们不公平的好处）。而且，由于输出路由 1540 多播 UDP 流的能力，使得主机服务 210 能够同时向观众中的数千人或更多广播较大锦标赛。

[0315] 事实上，当存在如此风行以致数千名用户正接收相同流的特定视频流时（例如，显示较大锦标赛的视图），可以更有效的将视频流发送到内容传送网络 (CDN)（诸如，Akamai（阿卡迈公司）或 Limelight（聚光灯公司））以用于大量分配到许多客户端设备 415。

[0316] 当使用 CDN 来显示顶级玩家分群的游戏取景器页面时，可获得类似水平的效率。

[0317] 对于较大锦标赛，可使用现场直播的名人解说员来在特定比赛期间提供评论。尽管大量用户将是在观看较大锦标赛，且相对小数目将是在锦标赛中玩。可将来自名人解说员的音频路由到对在锦标赛中玩的用户进行主机代管且对锦标赛中的游戏的任何旁观者模式复本进行主机代管的应用程序 / 游戏服务器 1521-1525，且可将音频加录于游戏音频之上。可在游戏上（也可能刚好在旁观者视图上）叠加名人解说员的视频。

[0318] 网页载入的加速

[0319] 全球信息网及其主要传送协议、超文本传输协议 (HTTP) 是被构想并被限定在其中仅商业具有高速互联网连接，且在线的消费者使用拨号数据机或 ISDN 的时代中。此时，用于快速连接的“黄金标准”是 T1 线，其对称地提供 1.5Mbps 数据速率（也即，两个方向中具有相等数据速率）。

[0320] 现今，情形完全不同。大量发达世界中经由 DSL 或电缆调制解调器连接的平均家庭连接速度具有比 T1 线高得多的下行数据速率。事实上，在世界的一些地方中，光纤到路边 (fiber-to-the-curb) 正将高达 50Mbps 至 100Mbps 的数据速率带入家庭。

[0321] 遗憾地，HTTP 没有被架构（也没有被实施）成有效地利用该急剧的速度改善。网站为远程服务器上的档案的集合。非常简单地说，HTTP 请求第一档案，等待下载该档案，且接着请求第二档案，等待下载该档案等。事实上，HTTP 允许一个以上“开放连接”（也即，每次请求一个以上档案），但由于商定的标准（及防止网络服务器被超载的愿望）而使得仅准许非常少的开放连接。此外，由于网页被构造的方式，浏览器常常未意识到可用于立即下载的多个同时页面（也即，仅在剖析一个页面之后才变得显而易见 :需要下载如图像的新档案）。因此，网站上的档案实质上是逐个地载入。此外，由于由 HTTP 使用的请求及响应协议，存在与所载入的每个档案相关的大致（访问美国的典型网络服务器）100 毫秒的延时。

[0322] 在相对低速连接的情况下，这不会引入大量问题，因为用于档案本身的下载时间决定网页的延时。但是，随着连接速度增大（尤其是复杂网页情况下），开始引起问题。

[0323] 在图 24 中所显示的实例中，显示了典型商业网站（该特定网站来自较大的运动鞋商标）。网站上具有 54 个档案。档案包括 HTML、CSS、JPEG、PHP、JavaScript 及 Flash 档案，且包括视频内容。在现场直播网页（也即，用户可单击该网页并开始使用该网页）之前，必须载入总共 1.5M 字节。对于大量档案存在许多原因。首先，该网页为复杂且尖端的网页，且其次，该网页为基于关于存取该页面的用户的信息动态地组合的网页（例如，用户来自哪个国家，何种语言，用户之前是否进行购买等），且视所有这些因素而下载不同档案。但是，其仍为非常典型的商业网页。

[0324] 图 24 显示随着连接速度增大在现场直播网页之前经过的时间量。在 1.5Mbps 连接速度 2401 下，使用具有传统网络浏览器的传统网络服务器，在现场直播网页之前花费 13.5 秒。在 12Mbps 连接速度 2402 下，载入时间减少至 6.5 秒，或约快一倍。但在 96Mbps 连接速度 2403 下，载入时间仅减少至约 5.5 秒。这个原因是因为在这种高下载速度下，下载档案本身的时间最小，但每个档案各自大致 100 毫秒的延时仍保持，从而导致 54 个档案 *100 毫秒 = 5.4 秒的延时。因此，无论到家庭的连接多快，该网站在现场直播之前将总是花费至少 5.4 秒。另一因素是服务器侧排队；每个 HTTP 请求是在队列的后部添加，因此在忙碌的服务器上这将具有显著影响，因为对于要从网络服务器得到的每个小项目，HTTP 请求需要等待其返回。

[0325] 解决这些问题的一个方式是废弃或重新限定 HTTP。或者，可能使网站拥有者较佳地将其档案合并成单一档案（例如，以 Adobe Flash 格式）。但是，作为一个实际问题，该公司以及许多他人在其网站架构中具有大量投资。另外，尽管一些家庭具有 12-100Mbps 连接，但大多数家庭仍具有较缓慢的速度，且 HTTP 在缓慢速度下确实工作良好。

[0326] 一个替代方法是在应用程序 / 游戏服务器 1521-1525 上对网络浏览器进行主机代管，且在 RAID 阵列 1511-1512 上（或潜在地在对网络浏览器进行主机代管的应用程序 / 游戏服务器 1521-1525 上的 RAM 中或本地储存器上）对用于网络服务器的档案进行主机代管。由于经由入埠路由 1502（或至本地储存器）的非常快的互连，并非在使用 HTTP 下每档案具有 100 毫秒的延时，而是在使用 HTTP 下将存在每档案最小延时。接着，并非使家庭中的用户经由 HTTP 存取网页，而是用户可经由客户端 415 存取网页。接着，甚至在 1.5Mbps 连接下（因为此网页不需要大量带宽来用于其视频），网页也将在每个线 2400 小于 1 秒的时间内处于现场直播。实质上，在应用程序 / 游戏服务器 1521-1525 上执行的网络浏览器显示现场直播的页面之前将不存在延时，且在客户端 415 显示来自网络浏览器的视频输出之前将不存在可侦测到的延时。当用户使用鼠标搜寻网页和 / 或在网页上键入字时，将用户的输入信息发送至在应用程序 / 游戏服务器 1521-1525 上执行的网络浏览器，且网络浏览器将相应地作出响应。

[0327] 此方法的一个不利之处是：若压缩器正恒定地传输视频数据，则使用带宽，即使网页变成静态也如此。这可通过配置压缩器以仅在（并且如果）网页改变时才传输数据且接着仅将数据传输到发生改变的页面部分来补救。当存在具有恒定地改变的闪烁标语等的一些网页时，所述网页往往令人讨厌，且除非存在要移动某物（例如，视频剪辑）的原因，否则通常网页为静态的。对于所述网页，可能为以下状况：使用主机服务 210 将传输较少数据（与传统网络服务器相比），因为将仅传输实际显示的图像，没有精简型客户端可执行代码，且没有可能从不被观看的大对象（诸如，滚动翻转图像）。

[0328] 因此，使用主机服务 210 来对旧版网页进行主机代管，可将网页载入时间减少到打开网页是类似改变电视上的频道的程度：有效地即刻地现场直播该网页。

[0329] 便于游戏及应用程序的除错

[0330] 如先前所述，具有实时图形的视频游戏及应用程序为非常复杂的应用程序且通常当其被发行到该领域中时，其含有缺陷。尽管软件开发商将自用户得到关于缺陷的反馈，且其可能具有用于在崩溃之后将机器状态传回的一些方式，但确切地识别是什么引起游戏或实时应用程序崩溃或不适当地执行非常困难。

[0331] 当游戏或应用程序在主机服务 210 中执行时，将游戏或应用程序的视频 / 音频输出恒定地记录在延迟缓冲器 1515 上。另外，看门狗进程执行每个应用程序 / 游戏服务器 1521-1525，该看门狗进程将向主机服务控制系统 401 定期地报告应用程序 / 游戏服务器 1521-1525 正平稳地执行。若看门狗进程未能报告，则服务器控制系统 401 将试图与应用程序 / 游戏服务器 1521-1525 通信，且若成功，则将收集可用的无论什么机器状态。将无论什么可用的信息连同由延迟缓冲器 1515 记录的视频 / 音频一起发送到软件开发商。

[0332] 因此，当游戏或应用程序软件开发商自主机服务 210 得到崩溃的通知时，其得到导致崩溃的原因的帧接帧的纪录。该信息在追踪到缺陷并将缺陷修复中可能是极具价值的。

[0333] 还应该注意，当应用程序 / 游戏服务器 1521-1525 崩溃时，在最近的可重新启动的时刻重新启动服务器，且将消息提供给用户，从而就技术困难道歉。

[0334] 资源共用及成本节省

[0335] 图 4a 及图 4b 中所显示的系统为终端用户与游戏及应用程序开发商两者提供多种益处。举例而言，通常，家庭及办公室客户端系统（例如，PC 或游戏控制台）仅在一周中的小百分比的小时中处于使用中。根据由 Nielsen（尼尔森）娱乐“Active Gamer Benchmark Study（活跃游戏者基准点研究）”(http://www.prnewswire.com/cgi-bin/stories.pl ? ACCT = 104&STORY = /www/story/10-05-2006/0004446115&EDATE =) 的 2006 年 10 月 5 日通信稿，活跃的游戏者一周平均花费 14 个小时来在视频游戏控制台上玩且约一周 17 个小时在手持设备上玩。该报告还陈述：对于所有游戏播放活动（包括控制台、手持设备及 PC 游戏播放），活跃的游戏者平均一周 13 个小时。考虑较高数字的控制台视频游戏播放时间，存在一周 24*7 = 168 个小时，这暗示在活跃游戏者的家中，视频游戏控制台仅在一周的 17/168 = 10％的小时中处于使用中。或者，90％的时间，视频游戏控制台是闲置的。给定视频游戏控制台的高成本，及制造商资助所述设备的事实，这是昂贵资源的非常无效率的使用。商业内的 PC 通常也仅在一周的一部分小时中使用，尤其是高端应用程序（诸如，Autodesk Maya）常常所需的非便携式台式 PC。尽管一些商业在所有小时及假日都操作，且一些 PC（例如，带回家以用于在晚上进行工作的便携式 PC）是在所有小时及假日使用，但大多数商业活动倾向于在给定商业时区中集中在从周一至周五的约 9AM 至 5PM、较少的假日以及休息时间（诸如，午餐），且因为大多数 PC 使用在用户积极地利用 PC 时出现，所以其遵循：台式 PC 的利用倾向于遵循这些操作小时数。若假定一周中的五天的自 9AM 至 5PM 不断地使用 PC，则这将暗示 PC 在一周的 40/168 = 24％的小时中被使用。高性能台式 PC 是用于商业的非常昂贵的投资，且这反映了非常低的利用度。在台式计算机上教学的学校可在一周的甚至更小部分中使用计算机，且尽管其视教学的小时数而改变，但大多数教学在自

周一至周五的日间小时期间出现。因此，一般而言，PC 及视频游戏控制台仅在一周的小部分小时中被利用。

[0336] 值得注意地，因为许多人在非假日的周一至周五的日间小时期间在商业或在学校工作，所以这些人通常在这些小时期间不玩视频游戏，且因此当其确实玩视频游戏时，其通常是在其他小时期间（诸如，晚上、周末及假日）。

[0337] 给定图 4a 中所显示的主机服务的配置，则上述两段中所描述的使用模式导致资源的非常有效的利用。显而易见，存在对于可在给定时间由主机服务 210 来服务的用户的数目的限制，尤其在用户需要用于复杂应用程序（如尖端 3D 视频游戏）的实时响应性的情况下。但是，不同于家庭中的视频游戏控制台或由商业使用的 PC（其通常在大多数时间闲置放置），服务器 402 可由不同用户在不同时间重新利用。举例而言，具有高性能双 CPU 及双 GPU 及大量 RAM 的高性能服务器 402 可由商业及学校在非假日的 9AM 至 5PM 利用，但由玩尖端视频游戏的游戏者在晚上、周末及假日利用。类似地，低性能应用程序可由商业及学校在商业小时期间在具有 Celeron（赛扬）CPU、无 GPU（或非常低端的 GPU）及有限 RAM 的低性能服务器 402 上利用且低性能游戏可在非商业小时期间利用低性能服务器 402。

[0338] 另外，在本文中所描述的主机服务配置的情况下，资源是在数千名（若非数百万名）用户当中有效地共用。一般而言，在线服务仅具有其总用户基础的小百分比在给定时间使用服务。若考虑先前所列出的 Nielsen 视频游戏使用统计数据，则容易了解为什么。若活跃游戏者一周仅 17 个小时玩控制台游戏，且若假定游戏的峰值使用时间是在晚上 (5-12AM，7*5 天＝ 35 小时 / 周）及周末 (8AM-12AM，16*2 ＝ 32 小时 / 周）的典型非工作、非商业小时期间，则对于 17 个小时的游戏播放，一周存在 35+32 ＝ 65 个峰值小时。由于以下许多原因而使得难以估计系统上的确切峰值用户负载：一些用户将在峰值外时间期间玩，可能存在特定日间时间存在用户的丛集 (clustering) 峰值，峰值时间可受所玩游戏的类型（例如，孩子的游戏将可能是在晚上的较早时间玩）等影响。但是，假定当游戏者可能玩游戏时，游戏者玩的平均小时数远小于日间的小时数，则仅主机服务 210 的一部分数目的用户将是在给定时间使用主机服务 210。为了该分析，我们假定峰值负载为 12.5%。因此，仅 12.5% 的计算、压缩及带宽资源是在给定时间使用，从而由于资源的再使用而导致仅 12.5% 的硬件成本来支持给定用户玩性能游戏的给定级别。

[0339] 此外，假定一些游戏及应用程序需要比其他者多的计算能力，则可基于被用户玩的游戏或由用户执行的应用程序来动态地分配资源。因此，选择低性能游戏或应用程序的用户将被分配低性能（较低廉）服务器 402，且选择高性能游戏或应用程序的用户将被分配高性能（较昂贵）服务器 402。实际上，给定游戏或应用程序可能具有游戏或应用程序的较低性能及较高性能区，且可在游戏或应用程序的区之间将用户从一个服务器 402 切换到另一服务器 402，以保持用户在满足游戏或应用程序的需要的最低成本服务器 402 上执行。注意，比单个磁盘快得多的 RAID 阵列 405 将可以被甚至低性能服务器 402 所用，这具有较快磁盘传送速率的益处。因此，跨越所有所玩游戏或所使用的应用程序的每服务器 402 平均成本比玩最高性能游戏或应用程序的大多数昂贵服务器 402 的成本小得多，然而，即使低性能服务器 402 也会从 RAID 阵列 405 得到磁盘性能益处。

[0340] 另外，主机服务 210 中的服务器 402 可能只是不具有磁盘或周边接口（不同于网络接口）的 PC 主机板，且恰好，可向下整合成刚好具有到 SAN 403 的快速网络接口的单个

芯片。而且，RAID 阵列 405 可能将在比存在磁盘的情况多得多的用户当中共用，因此每个活跃的用户的磁盘成本将远小于一个磁盘驱动器。所有该设备将可能驻留于环境上受控制的服务器室环境中的支架中。若服务器 402 出故障，则其可容易地在主机服务 210 处进行修理或替换。相比之下，家庭或办公室中的 PC 或游戏控制台必须坚固，必须能够幸免于合理的磨损及撕裂以防被重击或降落的独立器具需要外壳，具有至少一个磁盘驱动器，必须幸免于不利的环境条件（例如，被勉强塞入具有其他用具的过热 AV 橱柜中），需要服务保证，必须被封装及装运，且由可能收取零售利润的零售商来出售。另外，PC 或游戏控制台必须被配置以满足将在未来某一时刻使用的计算上最密集的预期游戏或应用程序的峰值性能，即使较低性能游戏或应用程序（或游戏或应用程序的区）也可能在大多数时间玩。此外，若 PC 或控制台出故障，则使其得到修理是昂贵且耗时的过程（不利地影响制造商、用户及软件开发商）。

[0341] 因此，假定图 4a 中所显示的系统将相当于本地计算资源的体验的体验提供给用户，以供用户在家庭、办公室或学校中体验给定水平的计算能力，则通过图 4a 中所显示的架构提供所述计算能力要低廉得多。

[0342] 消除对升级的需要

[0343] 另外，用户不必再担忧将 PC 和 / 或控制台升级以玩新游戏或处理较高性能的新应用程序。主机服务 210 上的任何游戏或应用程序（不管所述游戏或应用程序需要何类型的服务器 402）均可为用户所用，且所有游戏及应用程序接近即刻地执行（也即，快速地从 RAID 阵列 405 或服务器 402 上的本地储存器载入）且适当地具有最新更新及缺陷修复（也即，软件开发商将能够选择用于执行给定游戏或应用程序的服务器 402 的理想服务器配置，且接着将服务器 402 配置有最佳驱动器，且接着随着时间的推移，开发商将能够同时将更新、缺陷修复等提供给主机服务 210 中的游戏或应用程序的所有复本）。实际上，在用户开始使用主机服务 210 之后，用户可能发现游戏及应用程序继续提供较佳体验（例如，经由更新和 / 或缺陷修复）且可能是以下状况 ：用户一年后发现新游戏或应用程序可用于利用计算技术（例如，较高性能的 GPU）（其在一年前甚至不存在）的服务 210 上，因此对于用户而言，将不可能购买将在一年后玩游戏或执行应用程序的一年前的技术。因为玩游戏或执行应用程序的计算资源对于用户而言不可见（也即，自用户的观点看，用户仅选择开始接近即刻地执行的游戏或应用程序 - 更像用户改变电视上的信道），所以用户的硬件将在用户甚至未意识到升级的情况下已被“升级”。

[0344] 消除对于备份的需要

[0345] 对于商业、学校及家庭中的用户的另一较大问题是备份。若磁盘出故障，或若存在无意擦除，则储存在本地 PC 或视频游戏控制台中的信息（例如，在控制台的状况下，用户的游戏成果及等级）可能丢失。存在提供用于 PC 的手动或自动备份的许多可用的应用程序，且可将游戏控制台状态上传至在线服务器以供备份，但通常将本地备份复制至必须储存于安全且有组织的某处的另一本地磁盘（或其他非挥发性储存设备），且由于经由典型低成本互联网连接可用的缓慢上行速度而使得对于在线服务的备份常常有限。在图 4a 的主机服务 210 下，储存于 RAID 阵列 405 中的数据可使用为本领域技术人员所熟知的先前技术 RAID 配置技术来配置，以使得当磁盘出故障时，将不丢失数据，且将通知在容纳出故障的磁盘的服务器中心处的技术员，且接着技术员将替换该磁盘，该磁盘接着将被自动地更新以

使得 RAID 阵列再一次容忍故障。另外，因为所有磁盘驱动器彼此接近且其间具有经由 SAN 403 的快速本地网络，所以在服务器中心中将所有磁盘系统配置定期地备份到次级储存器（其可储存于服务器中心处或者经易地重新定位）并不困难。从主机服务 210 的用户的观点看，其数据始终完全安全，且其从不必考虑备份。

[0346] 对演示的存取

[0347] 用户经常希望在购买游戏或应用程序的前试用游戏或应用程序。如先前所述，存在先前技术装置，通过该先前技术装置来演示（“演示”的动词形式意思是试用演示版本，演示版本也被称为“演示”，但作为名词）游戏及应用程序，但其中的每一者遭受限制和 / 或不便利。使用主机服务 210，对于用户而言，容易且便于试用演示。实际上，用户所进行的系经由用户接口（诸如，下文所描述的用户接口）选择演示且试用该演示。演示将几乎即刻地载入适合于该演示的服务器 402 上，且其将完全类似任何其他游戏或应用程序而执行。无论演示需要非常高性能的服务器 402 还是低性能的服务器 402，且无论用户使用的家庭或办公室客户端 415 是何类型，自用户的观点看，演示均将工作。游戏演示或应用程序演示的软件出版商将能够确切地控制准许用户试用何演示及试用多长时间，且当然，演示可包括为用户提供获得对所演示的游戏或应用程序的全版本的存取机会的用户接口要素。

[0348] 因为演示可能是低于成本价或免费提供，所以一些用户可能试图使用重复的演示（尤其是重复地玩可能有趣的游戏演示）。主机服务 210 可使用各种技术来限制用于给定用户的演示使用。最直接的方法是建立用于每个用户的用户 ID 且限制允许给定用户 ID 播放演示的次数。然而，用户可设置多个用户 ID，尤其是其是自由的情况下。用于解决此问题的一个技术是限制允许给定客户端 415 播放演示的次数。若客户端为独立设备，则该设备将具有一序号，且主机服务 210 可限制演示可由具有所述序号的客户端存取的次数。若客户端 415 正以 PC 或其他设备上的软件执行，则可由主机服务 210 来指派序号且将该序号储存于 PC 上并使用该序号来限制演示使用，但假定 PC 可由用户来重新程序化，且序号被擦除或改变，则另一选项是主机服务 210 保持 PC 网络适配器媒体访问控制 (MAC) 地址（和 / 或其他机器特定识别符，诸如硬盘驱动器序号等）的纪录并将演示使用限制于该 MAC 地址。假定可改变网络适配器的 MAC 地址，然而，这并非极简单的方法。另一方法是限制演示可被播放到给定 IP 地址的次数。尽管可由电缆调制解调器及 DSL 提供者来周期性地重新指派 IP 地址，但其在实践中不会非常频繁地发生，且若可确定（例如，通过联系 ISP）IP 是处于用于住宅 DSL 或电缆调制解调器存取的 IP 地址的区块中，则通常可建立用于给定家庭的小数目的演示使用。而且，在家庭中在共用同一 IP 地址的 NAT 路由器之后可能存在多个设备，但通常在住宅背景中，将存在有限数目的所述设备。若 IP 地址是处于服务商业的区块中，则可建立用于商业的较大数目的演示。但是，最后，所有先前所述方法的组合是限制 PC 上的演示的数目的最佳方式。尽管可能不存在使得所确定的且技术上熟练的用户可能在重复播放演示的数目中受到限制的极简单的方式，但建立大量障碍可建立足够阻碍以使得大多数 PC 用户不值得费神去滥用演示系统，且相反，其在其意欲试用新游戏及应用程序时使用演示。

[0349] 对学校、商业及其他机构的益处

[0350] 显著益处尤其出现在利用图 4a 中所显示的系统的商业、学校及其他机构。商业及学校具有与安装、维护及升级 PC 相关联的实质成本，尤其当谈及执行诸如 Maya 的高性能应

用程序的PC时。如先前所陈述，PC通常仅在一周的小时的一部分中被利用，且如在家庭中，具有给定水平的性能能力的PC在办公室或学校环境中的成本远高于在服务器中心环境中的成本。

[0351] 在较大商业或学校（例如，大的大学）的状况下，所述实体的IT部门设置服务器中心且维护经由LAN级连接而远程地存取的计算机可以是实际的。存在用于经由LAN或经由办公室之间的私用高带宽连接而远程存取计算机的许多解决方法。举例而言，通过Microsoft的Windows终端机服务器，或者通过虚拟网络计算应用程序（如来自RealVNC（远程控制）有限公司的VNC）或者通过来自Sun Microsystems（太阳计算机系统公司）的精简型客户端装置，用户可获得对PC或服务器的远程存取，在图形响应时间及用户体验中具有一定范围的质量。另外，所述自行管理的服务器中心通常专用于单个商业或学校，且因此不能够利用在全异应用程序（例如，娱乐及商业应用程序）在一周的不同时间利用同一计算资源时所可能的使用的重叠。因此，许多商业及学校缺乏独立设置具有至每一用户的LAN速度的网络连接的服务器中心的规模、资源或专门技能。实际上，大百分比的学校及商业具有与家庭相同的互联网连接（例如，DSL、电缆调制解调器）。

[0352] 然而，所述组织仍可能具有对于非常高性能的计算的需要（或者定期地或者周期性地）。举例而言，小建筑公司可能仅具有小数目的建筑师，当进行设计工作时，具有相对适度的计算需要，但其可能周期性地需要非常高性能的3D计算（例如，当建立用于客户端的新建筑设计的3D飞越时）。图4a中所显示的系统极其适合于所述组织。所述组织仅需要为提供至家庭的同一种类的网络连接（例如，DSL、电缆调制解调器）且通常非常低廉。其可利用低廉的PC作为客户端415，或者完全没有PC也可以，而利用简单实施控制信号逻辑413及低延时视频解压缩412的低廉的专用设备。该特征对于可能具有PC的偷窃或对PC内的专用组件的损坏的问题的学校特别有吸引力。

[0353] 这种配置解决了用于所述组织的许多问题（且许多这种优点也为进行通用计算的家庭用户共用）。举例而言，操作成本（其最终必须以某种形式传递回至用户以便具有可行的商业）可能低得多，因为(a)计算资源是与在一周中具有不同峰值使用时间的其他应用程序共用，(b)所述组织可仅在需要时获得（且招致成本）对高性能计算资源的存取，(c)所述组织不必提供用于备份或以其他方式维护高性能计算资源的资源。

[0354] 盗版的消除

[0355] 另外，游戏、应用程序、互动式电影等可能不再如现今这样被盗版。因为每一游戏是在主机服务210处被存储及执行，所以用户不具备对于基本程序码的存取，因此不存在盗版。即使用户将要复制原始码，用户也不能够在标准游戏控制台或家庭计算机上执行该码。此打开了标准视频游戏不可用的世界各地（诸如，中国）的市场。已使用的游戏的重新销售也是不可能的，因为没有游戏副本分发给用户。

[0356] 对于游戏开发商而言，如同现今的状况，新一代的游戏控制台或PC被引入市场，存在较少市场不连续性。与全新的一代控制台或PC技术迫使用户及开发商升级且游戏开发商取决于硬件平台的及时传送给用户（例如，在游戏机3的情况下，引入被推迟了超过一年，且开发者必须等待直至其可得到并且大量单元被购买）的当前情形对比，可随着时间随着游戏要求改变而利用更先进的计算技术逐渐地更新主机服务210。

[0357] 流动互动式视频

[0358] 以上描述提供由以通用互联网为基础的低延时流动互动式视频（其隐含地也包括连同视频一起的音频，如本文中所使用）的新颖基本概念致能的多种应用。经由互联网而提供流动视频的先前技术系统仅具有可通过高延时互动实施的所致能的应用。举例而言，用于线性视频的基本回放控制（例如，暂停、回倒、快进）在高延时下适当地工作，且有可能在线性视频馈送当中进行选择。此外，如先前所陈述，一些视频游戏的性质允许其以高延时来播放。但是，用于流动视频的先前技术方法的高延时（或低压缩比率）严重限制流动视频的潜在应用或使其部署变窄到专门化的网络环境，且甚至在所述环境中，先前技术也引入网络上的实质负担。本文中所描述的技术打开了在经由互联网的低延时流动互动式视频下可能的多种应用的大门，尤其是经由消费者级互联网连接而致能的所述应用。

[0359] 实际上，在与图 4c 的客户端 465 一般小的客户端设备下，足以通过有效的任意量的计算能力、任意量的快速储存及强大服务器之间的极快网络连接而提供增强的用户体验，其使新的计算时代成为可能。另外，因为带宽要求并不随着系统的计算能力增长而增长（也即，因为带宽要求仅关于显示分辨率、质量及帧速率），所以一旦宽带互联网连接性是普遍存在的（例如，经由分布广的低延时无线涵盖）、可靠的且具有足以满足所有用户的显示设备 422 的需要的足够高的带宽，则问题将是典型消费者及商业应用所必要的是厚重客户端（诸如，执行 Windows、Linux、OSX 等的 PC 或移动电话）还是甚至精简型客户端（诸如，Adobe Flash 或 Java）。

[0360] 流动互动式视频的出现导致关于计算架构的结构的假定的重新考虑。该一个实例是图 15 中所显示的主机服务 210 服务器中心实施例。用于延迟缓冲器和 / 或分群视频 1550 的视频路径是反馈回路，其中应用程序 / 游戏服务器 1521-1525 的经多播的流动互动式视频输出经由路径 1552 而实时地或者经由路径 1551 在可选择的延迟之后被反馈回到应用程序 / 游戏服务器 1521-1525 中。这使得通过先前技术服务器或本地计算架构将是不可能或不可行的多种实际应用（例如，诸如图 16、图 17 及图 20 中所说明的应用）成为可能。但是，作为更一般的架构特征，反馈回路 1550 所提供的是流动互动式视频水平下的递归，因为可在应用程序需要视频时将视频无限地循环。这使得之前从未可用的多种应用可能性成为可能。

[0361] 另一关键架构特征在于：视频流是单向 UDP 流。这有效地实现流动互动式视频的任意程度的多播（相比之下，诸如 TCP/IP 流的双向流将随着用户的数目增加而在来自来回通信的网络上产生越来越多的通信停滞）。多播是服务器中心内的重要能力，因为其允许系统对互联网用户（且实际上，世界的人口）的增长的需要作出响应以在一对多或甚至多对多基础上通信。再次，本文中所论述的说明流动互动式视频递归与多播两者的使用的实例（诸如，图 16）仅为具有可能性的非常大的冰山的尖端。

[0362] 非转接对等 (NON-TRANSIT PEERING)

[0363] 在一实施例中，主机服务 210 具有至一个或多个互联网服务提供商 (ISP) 的一个或多个对等连接，该互联网服务提供商 (ISP) 也向用户提供互联网服务，这样主机服务 210 可通过保持在 ISP 网络内的非转接路由而与用户通信。例如，如果主机服务 210WAN 接口 441 直接连接到康卡斯特电缆通信有限公司的网络，为用户场所 211 通过康卡斯特电缆调制解调器而被提供宽带服务，主机服务 210 和客户端 415 之间的路由可全部在康卡斯特的网络内建立。其潜在的优点将包括：较低的通信费用（因为可避免两个或多个 ISP 网络之间的

IP 转接费用），潜在的更可靠的连接（可防止 ISP 网络之间存在拥塞或其他转接中断）和较低延时（可防止 ISP 网络之间存在拥塞、低效路由或其他延迟）。

[0364] 该实施例中，在通话初期，当客户端 415 开始联系主机服务 210 时，主机服务 210 接收用户场所 211 的 IP 地址。之后其使用例如来自 ARIN（美国网络地址注册管理组织）的有效 IP 地址表，查看该 IP 地址是否是分配给连接到主机服务 210 的特定 ISP 的 IP 地址，该主机服务 210 能够路由到用户场所 211 而无需通过另外的 ISP 来 IP 转接。例如，如果 IP 地址位于 76.21.0.0 和 76.21.127.255 之间，则 IP 地址被分配给康卡斯特电缆通信有限公司。在该示例中，如果主机服务 210 保持至康卡斯特、AT&T 和 Cox ISP 的连接，则其选择康卡斯特作为最可能向该特定用户提供最佳路由的 ISP。

[0365] 使用反馈的视频压缩

[0366] 在一实施例中，从客户端设备向主机服务提供反馈以表明成功的（或不成功的）图像块和 / 或帧传送。从客户端提供的反馈信息之后被用于调节主机服务处的视频压缩操作。

[0367] 例如，图 25a-b 示出了本发明的一个实施例，其中在客户端设备 205 和主机服务 210 之间建立反馈信道 2501。客户端设备 205 使用该反馈信道 2501 发送成功接收的图像块 / 帧的分组化确认和 / 或不成功接收的图像块 / 帧的指示。

[0368] 在一实施例中，成功接收每个图像块 / 帧之后，客户将确认信息发送到主机服务 210。该实施例中，如果在指定时段后没有接收到确认和 / 或如果接收到客户端设备 205 所接收的比已发送的图像块 / 帧靠后的图像块 / 帧的确认，则主机服务 210 检测分组丢失。可选地，或另外地，客户端设备 205 可检测分组丢失并将分组丢失的指示连同受该分组丢失影响的图像块 / 帧的指示一起发送给主机服务 210。该实施例中，并不需要成功传送的图像块 / 帧的连续确认。

[0369] 不管如何检测分组丢失，在图 25a-b 所例示的实施例中，生成用于图像的 I- 图像块的初始组（图 25a 中未示出）后，编码器随后仅生成图像的 P- 图像块，直到检测到分组丢失。注意在图 25a 中，诸如 2510 的每一个帧被示为 4 个纵向的图像块。该帧可以按不同的配置被铺放，例如 2×2，2×4，4×4 等，或者该帧可以在没有图像块的情况下整个被编码（例如作为 1 个大的图像块）。为了示例本发明的该实施例而提供了帧铺放配置的上述实例。本发明下面的原理并不限于任何特定的帧铺放配置。

[0370] 只传送 P- 图像块降低了因为上面提出的所有原因的信道带宽需求（例如，p- 图像块通常比 I- 图像块小）。当经由反馈信道 2501 检测到分组丢失时，如图 25b 所示，由编码器 2500 生成新的 I- 图像块，以重新初始化客户端设备 205 上解码器 2502 的状态。如图所示，在一实施例中，I- 图像块分布于多个编码帧上，以限制由每个单独的编码帧消耗的带宽。例如图 25 中，其中每个帧包括 4 个图像块，在 4 个连续的编码帧内的不同位置处传送单个 I- 图像块。

[0371] 编码器 2500 可将所描述的和该此实施例相关的技术与此处描述的其他编码技术组合。例如，除了响应于所检测的分组丢失而生成 I- 图像块外，编码器 2500 可在其他情况中生成 I- 图像块，在该情况中，I- 图像块可能有益于正确地再现图像序列（例如，响应于突然的场景转换）。

[0372] 图 26a 示出了本发明的另一个实施例，其依赖于客户端设备 205 和主机服务 210

之间的反馈信道 2601。并非响应于所检测的分组丢失而生成新的 I- 图像块 / 帧，此实施例的编码器 2600 调整 P- 图像块 / 帧的相依性。作为初始问题，需要注意的是：此实例中所提出的具体细节无需符合本发明的基本原理。例如，虽然通过使用 P- 图像块 / 帧描述本实例，但本发明的基本原理并不限于任何特定的编码格式。

[0373] 图 26a 中，编码器 2600 将多个未压缩的图像块 / 帧 2605 编码为多个 P- 图像块 / 帧 2606，并通过通信信道（例如互联网）将该 P- 图像块 / 帧传送给客户端设备 205。客户端设备 205 上的解码器 2602 解码 P- 图像块 / 帧 2606，生成多个解压缩的图像块 / 帧 2607。编码器 2600 过去的状态 2611 存储于主机服务 210 上的存储设备 2610 内，而解码器 2602 过去的状态 2621 存储于客户端设备 205 上的存储设备 2620 内。解码器的“状态”是诸如 MPEG-2 和 MPEG-4 的视频编码系统领域中公知的术语。在一实施例中，存储器内所存储的过去“状态”包含来自在先 P- 图像块 / 帧的组合数据。存储器 2611 和 2621 可分别集成到编码器 2600 和解码器 2602 中，而非与编码器 2600 和解码器 2602 分离，正如图 26a 所示。此外，可以使用各种类型的存储器，包括（以示例而非限制的方式）随机存取存储器。

[0374] 在一实施例中，当没有分组丢失发生时，编码器 2600 将每个 P- 图像块 / 帧编码为依赖于之前的 P- 图像块 / 帧。因此，正如图 26a 使用的符号所指示，P- 图像块 / 帧 4 依赖于 P- 图像块 / 帧 3（使用符号 4_3 标识）；P- 图像块 / 帧 5 依赖于 P- 图像块 / 帧 4（使用符号 5_4 标识）；而 P- 图像块 / 帧 6 依赖于 P- 图像块 / 帧 5（使用符号 6_5 标识）。在该实例中，P- 图像块 / 帧 4_3 在编码器 2600 和解码器 2602 之间的传送期间已被丢失。可以各种方式将该丢失传送到编码器 2600，包括但不限于上述的那些方式。例如，每次解码器 2606 成功地接收和 / 或解码图像块 / 帧时，可将此信息从解码器 2602 传送到编码器 2600。如果编码器 2600 并未在一时段之后接收到特定图像块 / 帧已被接收和 / 或解码的指示，则编码器 2600 将假定所述图像块 / 帧未被成功接收。可选地，或者另外地，当特定图像块 / 帧未被成功接收时，解码器 2602 可通知编码器 2600。

[0375] 在一实施例中，不管如何检测丢失的图像块 / 帧，一旦出现该情况，则编码器 2600 通过使用已知的由解码器 2602 成功接收的上一个图像块 / 帧，编码下一个图像块 / 帧。在图 26a 所示的实例中，图像块 / 帧 5 和 6 不认为是“成功接收”，由于图像块 / 帧 4 的丢失，其不能由解码器 2602 正确地解码（即，图像块 / 帧 5 的解码依赖于图像块 / 帧 4 而图像块 / 帧 6 的解码依赖于图像块 / 帧 5）。因此在图 26a 所示的实例中，编码器 2600 将图像块 / 帧 7 编码为依赖图像块 / 帧 3（上一个成功接收的图像块 / 帧）而非解码器 2602 不能正确解码的图像块 / 帧 6。虽然图 26a 中没有示出，随后将图像块 / 帧 8 编码为依赖于图像块 / 帧 7 而将图像块 / 帧 9 编码为依赖于图像块 / 帧 8，假定未检测到另外的分组丢失。

[0376] 如上所提及，编码器 2600 和解码器 2602 将过去的解码器和编码器状态 2611 和 2621，分别保持于存储器 2610 和 2620 内。因此当编码图像块 / 帧 7 时，编码器 2600 从存储器 2610 中获取与图像块 / 帧 3 相关的先前编码器状态。同样地，和解码器 2602 相关的存储器 2620 至少存储上一个已知的好解码器状态（实例中与图像块 / 帧 3 相关的状态）。因此，解码器 2602 获取与图像块 / 帧 3 相关的过去状态信息以使得图像块 / 帧 7 可被解码。

[0377] 由于上述技术，可使用相对小的带宽来对实时、低延时、交互的视频进行编码和并使其流动，因为不曾需要 I- 图像块 / 帧（除了在流开始处初始化解码器和编码器）。此外，虽然解码器产生的视频图像可能临时包括因丢失的图像块 / 帧 4 和图像块 / 帧 5 以及 6 而

产生的不想要的失真（由于图像块 / 帧 4 的丢失，其不能被正确解码），但该失真仅在非常短的持续时间可见。此外，如果使用图像块（而非全视频帧），此失真将被限定到所再现的视频图像的特定区域。

[0378] 图 26b 示出了根据本发明一个实施例的方法。在 2650 处，基于之前生成的图像块 / 帧而生成图像块 / 帧。在 2651 处，检测到丢失的图像块 / 帧。在一实施例中，基于从编码器传送到解码器的信息来检测丢失的图像块 / 帧，如上所述。在 2652 处，基于已知的在解码器处已被成功接收和 / 或解码的图像块 / 帧，生成下一个图像块 / 帧。在一实施例中，通过从存储器加载与成功接收和 / 或解码的图像块 / 帧有关的状态，该编码器生成下一个图像块 / 帧。同样地，当解码器接收到新的图像块 / 帧时，其通过从存储器加载与成功接收和 / 或解码的图像块 / 帧有关的状态而解码该图像块 / 帧。

[0379] 在一实施例中，基于编码器处成功接收和 / 或解码的上一个图像块 / 帧而生成下一个图像块 / 帧。在另一实施例中，生成的下一个图像块 / 帧是 I 图像块 / 帧。再一个实施例中，选择基于先前成功接收的图像块 / 帧生成下一个图像块 / 帧还是生成为 I 帧是基于：丢失了多少图像块 / 帧和 / 或信道的延时。在丢失了相对小数量（例如 1 或 2）的图像块 / 帧以及来回行程延时相对低（例如 1 或 2 帧时间）的情况下，生成 P 图像块 / 帧可能是最佳的，因为上一次成功接收的图像块 / 帧和新生成的图像块 / 帧之间的差异可能相对小。如果丢失了若干个图像块 / 帧且来回行程延时高，则生成 I 图像块 / 帧可能是最佳的，因为上一次成功接收的图像块 / 帧和新生成的图像块 / 帧之间的差异可能大。在一实施例中，设定图像块 / 帧丢失阈值和 / 或延时阈值以确定是传送 I 图像块 / 帧还是 P 图像块 / 帧。如果丢失的图像块 / 帧的数量低于图像块 / 帧丢失阈值和 / 或如果来回行程延时低于延时阈值，则生成新的 I 图像块 / 帧；否则生成新的 P 图像块 / 帧。

[0380] 在一实施例中，编码器总是尝试生成与上次成功接收的图像块 / 帧有关的 P 图像块 / 帧，而若编码过程中编码器确定 P 图像块 / 帧将可能大于 I 图像块 / 帧时（例如，如果其已压缩 1/8 的图像块 / 帧且压缩的大小大于先前压缩的 I 图像块 / 帧平均大小的 1/8），则编码器将放弃压缩 P 图像块 / 帧且将改为压缩 I 图像块 / 帧。

[0381] 如果丢失的分组很少发生，则上述使用反馈来报告丢弃的 (dropped) 图像块 / 帧的系统会典型地导致至用户的视频流中出现非常微小的中断，因为由丢失分组所中断的图像块 / 帧在客户端设备 205 和主机服务 210 之间大概一个来回行程的时间内被替换，假定编码器 2600 在短时间内压缩该图像块 / 帧。而且因为被压缩的新图像块 / 帧是基于未压缩视频流中稍后的帧，所以视频流并没有落后于未压缩视频流。但是如果含有新图像块 / 帧的包也被丢失，则这将导致至少两个来回行程的延迟以再次请求和发送另一个新图像块 / 帧，在许多实际情况下其将导致视频流的显著中断。因而，从主机服务 210 向客户端设备 205 成功发送在丢失的图像块 / 帧之后发送的新编码的图像块 / 帧是非常重要的。

[0382] 在一实施例中，前向纠错 (FEC) 技术，例如先前在图 11a、11b、11c 和 11d 中所描述和示出的那些技术，被用于减轻丢失新编码的图像块 / 帧的可能性。如果传送图像块 / 帧时 FEC 编码已被使用，则更强的 FEC 码用于新编码的图像块 / 帧。

[0383] 丢失分组的一个潜在原因是信道带宽的突然损耗，例如，如果用户场所 211 处宽带连接的其他某些用户开始使用大量的带宽。如果新生成的图像块 / 帧也因丢弃分组（即使使用了 FEC）而丢失，则在一实施例中：当客户端 415 通知主机服务 210 第二个新编码的

图像块/帧丢失，视频压缩器 404 在其编码随后新编码的图像块/帧时，降低数据速率。不同的实施例使用不同的技术降低数据速率。例如在一实施例中，通过增加压缩比，降低编码图像块/帧的品质，从而达到此数据速率的降低。在另一实施例中，通过降低视频的帧速率（例如从 60fps 到 30fps）并相应地减慢数据传输的速率而降低数据速率。在一实施例中，用于降低数据速率的技术都被使用（例如既降低帧速率又增加压缩比）。如果数据传输的较低速率对减轻丢弃分组是成功的，则依照先前所述的信道数据速率检测和调节方法，主机服务 210 将继续以较低的数据速率编码，之后随着信道所允许而逐渐地向上或向下调节数据速率。与丢弃分组和/或延时有关的反馈数据的连续接收允许主机服务 210 基于当前信道条件动态调节数据速率。

[0384] 在线游戏系统中的状态管理

[0385] 该发明的一个实施例使用技术来有效地存储和运送服务器间活跃游戏的当前状态。虽然此处描述的实施例是关于在线游戏的，但该发明的基本原理可用于其他各种类型的应用程序（例如设计应用程序、文字处理软件、诸如电邮或即使消息接发的通信软件等）。图 27a 示出了用于实施该实施例的示例性系统架构而图 27b 示出了示例性的方法。虽然将同时描述该方法和系统架构，但图 27b 所示的方法并不限于任何特定的系统架构。

[0386] 图 27b 中的 2751 处，用户从客户端设备 205 上启动主机服务 210a 上的新在线游戏。作为响应，在 2752 处，将游戏的“洁净 (clean)”图像 2702a 从存储器（例如硬盘驱动器，无论是直接连接到执行游戏的服务器，还是通过网络连接到服务器）加载至主机服务 210a 上的存储器（例如 RAM）。“洁净”图像包括在任何游戏播放启动之前，游戏的运行时间程序代码和数据（例如，就像当首次执行游戏时那样）。之后在 2753 处，用户玩游戏，使得“洁净”图像变为非洁净图像（例如，由图 27a 中“状态 A”所表示的正执行的游戏）。在 2754 处，游戏由用户或主机服务 210a 中的任一个暂停或终止。在 2755 处，主机服务 210a 上的状态管理逻辑电路 2700a 确定游戏的“洁净”图像和当前游戏状态（“状态 A”）之间的差异。各种已知的技术可用于计算两个二值图像之间的差异，该技术例如包括 UNIX 操作系统上的公知“diff”工具所使用的那些技术。当然，本发明的基本原理并不限于任何用于差异计算的特定技术。

[0387] 不管如何计算该差异，一旦使用它们，则差异数据会被本地存储于存储设备 2705a 内和/或被传送给不同主机服务 210b。如果被传送到不同的主机服务 210b，该差异数据可能被存储到新主机服务 210b 的存储设备（未示出）上。在任一种情况中，差异数据与主机服务上的用户帐号相关，以使得其可在下一次用户登陆主机服务并启动该游戏时被识别。在一实施例中，并非立刻传送，该差异数据不被传送到新的主机服务，直到下一次用户尝试玩该游戏（而不同的主机服务被确定为作为该游戏的主机的最佳选择）。

[0388] 返回到图 27b 所示的方法，在 2757 处，用户从客户端设备重启游戏，所述客户端设备可以是用户最初玩该游戏使用客户端设备 205 或不同的客户端设备（未示出）。作为响应，在 2758 处，主机服务 210b 上的状态管理逻辑 2700b 从存储设备获取游戏的“洁净”图像和差异数据。在 2759 处，状态管理逻辑路 2700b 将洁净图像和差异数据组合以重新构建游戏在原始主机服务 210a 上的状态（“状态 A”）。可以使用各种已知的技术来使用所述差异数据，重新创建二值图像的状态，该技术例如包括 UNIX 操作系统上公知的“修补 (patch)”工具中所使用的那些技术。也可使用在诸如 PC 备份的公知备份程序中使用差异计算技术。

本发明的基本原理不限于任意使用差异数据重建二值图像的特定技术。

[0389] 此外，在 2760 处，平台相关 (platform-dependent) 数据 2710 被合并到最终的游戏图像 2701b 中。平台相关数据 2710 可包括唯一对应于目标服务器平台的任意数据。以示例而非限制的方式，平台相关数据 2710 可包括新平台的媒体访问控制 (MAC) 地址、TCP/IP 地址、时刻、硬件序列号（例如用于硬盘驱动器和 CPU）、网络服务器地址（例如 DHCP/Wins 服务器）以及软件序列号 / 激活码（包括操作系统序列号 / 激活码）。

[0390] 其他与客户 / 用户有关的平台相关数据可包括（但不限于）下列各项：

[0391] 1. 用户的屏幕分辨率。当用户恢复游戏时，用户可使用具有不同分辨率的不同设备。

[0392] 2. 用户的控制器配置。当游戏恢复时，用户可能已从游戏控制器切换到键盘 / 鼠标。

[0393] 3. 用户权限，例如是否折扣率已经期满（例如：如果用户曾在促销期间玩游戏，且现在正在较高消费的普通时段玩），或者用户或设备是否有某些年龄限制（例如，用户的父母可能已经改变用于孩子的设置，这样孩子不允许看成人资料，或者如果正在播放游戏的设备（例如公共图书馆的计算机）在是否显示成人资料方面具有某些限制）。

[0394] 4. 用户的等级。用户可能已被允许在某一联盟中玩多人游戏，但是因为其他某些用户已经超过该用户的等级，所以用户可能已被降级到较小的联盟中。

[0395] 为了解释本发明该实施例而提供了平台相关数据 2710 的前述实例。本发明的基本原理并不限于平台相关数据的任意特定集合。

[0396] 图 28 图形化地示出了第一主机服务处的状态管理逻辑 2700a 如何从正执行的游戏 2701a 中提取差异数据 2800。之后，第二主机服务处的状态管理逻辑 2700b 将洁净图像 2702b 和差异数据 2800 以及平台相关数据 2710 组合，以重新生成正执行的游戏 2701b 的状态。大体如图 28 所示，差异数据的大小显著地小于整个游戏图像 2701a 的大小，因此通过仅仅存储 / 传输差异数据而节省了大量的存储空间和带宽。虽然图 28 中未示出，但是平台相关数据 2700 在其被合并到最终的游戏图像 2701b 时，可改写某些差异数据。

[0397] 虽然上面描述了在线视频游戏的实施，但本发明的基本原理并不限于视频游戏。例如可在任意类型的在线主机应用程序的环境中实现前述状态管理技术。

[0398] 用于保持客户端解码器的技术

[0399] 在本发明一实施例中，每当用户请求连接主机服务 210 时，主机服务 210 便将新的解码器传送到客户端设备 205。因而，在该实施例中，客户端设备所使用的解码器总是最新的 (up-to-date) 且为客户端设备上实施的软件 / 硬件特别量身定做的。

[0400] 如图 29 所示，该实施例中，永久地安装于客户端设备 205 上的应用程序并不包括解码器。相反地，是客户端下载器应用程序 2903 来在每次客户端设备 205 连接主机服务 210 时，管理临时解码器 2900 的下载和安装。该下载器应用程序 2903 可被实施为硬件、软件、固件或其任意组合。响应于新的在线会话的用户请求，下载器应用程序 2903 通过网络（例如互联网）传送与客户端设备 205 相关的信息。该信息可包括识别客户端设备和 / 或客户端设备硬件 / 软件配置（例如，处理器、操作系统等）的识别数据。

[0401] 基于此信息，主机服务 210 上的下载器应用程序 2901 选择合适的临时解码器 2900 以用于客户端设备 205 上。主机服务上的下载器应用程序 2901 之后传送临时解码器 2900，

之后客户端设备上的下载器应用程序2903在客户端设备205上验证和/或安装解码器。编码器2902之后使用此处所述的任意技术来编码音频/视频内容，并将内容2910传送到解码器2900。一旦安装了新的解码器2900，其解码用于当前在线会话的内容（即，使用此处所述的一个或多个音频/视频解压缩技术）。在一实施例中，当会话终止时，解码器2900被从客户端设备205上移除（例如卸载）。

[0402] 在一实施例中，随着临时解码器2900正被下载，下载器应用程序2903通过做出信道评估（例如，信道上可达到的数据速率（例如，通过确定其花多长时间下载数据）、信道上的分组丢失率以及信道的延时）来描绘信道特性。下载器应用程序2903生成描述所述信道评估的信道特征化数据。之后将该信道特征化数据从客户端设备205传送到主机服务下载器2901，其使用信道特征化数据来确定如何最好地利用信道将媒体传送到客户端设备205。

[0403] 在下载临时解码器2900期间，客户端设备205一般会发送消息回主机服务205。这些消息可包括指示无误还是有误地接收到分组的确认信息。此外，该消息提供反馈至下载器2901，该反馈针对关于数据速率（基于接收分组的速率进行计算）、分组错误率（基于所报告的有误接收的分组的百分比）和信道来回行程延时（基于下载器2901接收反馈之前所花的时间量，所述反馈关于已被传输的给定分组）。

[0404] 以示例的方式，如果数据速率被确定为2Mbps，则和若数据速率被确定为5Mbps（例如1280×720,60fps）相比，下载器选择较小的视频窗口分辨率用于编码器2902（例如640×480,60fps）。可依赖于分组丢失率，选择不同的向前纠错(FEC)或分组结构。

[0405] 如果分组丢失非常低，则可无纠错地传送压缩的音频和视频。如果是分组丢失适中，则可用纠错编码技术（例如，像之前图11a、11b、11c和11d中描述和示出的那些）传送压缩的音频和视频。如果分组丢失非常高，则可确定不能传送优等质量的视听流，而客户端设备205可通知用户无法通过通信信道（例如，“链路(link)”）获得主机服务，或者其可能尝试建立至具有较低分组丢失的主机服务的不同路由（如下所述）。

[0406] 如果延时短，则可以以低延时传输压缩的音频和视频并建立会话。如果延时太高（例如高于80ms），则对于需要短延时的游戏来说，客户端设备205可通知用户无法通过链路获得主机服务（该链路可用但对用户输入的响应时间将是迟缓的或“滞后的”），或者用户可以尝试建立至具有较低延时的主机服务的不同路由（如下所述）。

[0407] 客户端设备205可尝试通过网络（例如互联网）经由另一条路由连接至主机服务210，查看是否降低了减损（例如，分组丢失更小、延时更短、或者甚至数据速率更高）。例如，主机服务210可从多个地理位置（例如，主机中心在洛杉矶而一个在丹佛）连接到互联网，由于洛杉矶拥挤所以可能有更高的分组丢失，但是在丹佛没有拥挤。此外，主机服务210可通过多个互联网服务提供商（例如AT&T和康卡斯特）连接到互联网。

[0408] 因为主机设备205和服务提供商之一（例如AT&T）之间的拥挤或其他问题，分组丢失和/或高延时和/或受约束的数据速率可能发生。然而，如果客户端设备205通过另一个服务提供商（例如康卡斯特）连接到主机服务210，其可能可以连接而无拥挤问题和/或较低的分组丢失和/或较短延时和/或更高的数据速率。因此，当下载临时解码器2900时，如果客户端设备205经历了高于指定阈值（例如，指定持续时间上，指定数量的分组丢

弃）的分组丢失、高于指定阈值的延时和 / 或低于指定阈值的数据速率，在一实施例中，其尝试通过可选的路由重新连接到主机服务 210（一般通过连接到不同的 IP 地址或不同的域名），以确定是否可获得更好的连接。

[0409] 可选连接选项耗尽后，如果连接依旧经历不可接受的减损，则其可能是 ：客户端设备 205 到互联网的本地连接遭受减损，或者是 ：离主机服务 210 太远而不能获得适当的延时。在此情况下，客户端设备 205 可能通知用户无法通过链路获得主机服务或者主机服务仅仅是有损可用的、和 / 或只有某些类型的低延时游戏 / 应用程序是可用的。

[0410] 因为主机服务 210 和客户端设备 205 之间的链路特征的评估和潜在的改进是在临时解码器正被下载时进行的，所以其降低了客户端设备 205 分别下载临时解码器 2900 和评估链路特征所需的花费的时间量。但是在另一实施例中，客户端设备 205 将下载临时解码器 2900 与链路特征的评估和潜在改进（例如，通过使用虚构的测试数据而非解码器程序代码）分开执行。为什么这可能是较佳实施有很多原因。例如在某些实施例中，客户端设备 205 部分或全部实施为硬件。因此，对于这些实施例，本身没有必要下载软件解码器。

[0411] 使用基于标准的图像块大小的压缩

[0412] 如上所提及，当使用基于图像块的压缩时，本发明的基本原理并不限于任何特定的图像块大小、形状或方向。例如在诸如 MPEG-2 和 MPEG-4 的基于 DCT 的压缩系统中，图像块可能是宏块 (macroblock)（数据压缩中使用的部件，其一般表示 16×16 像素的块）的大小。此实施例提供了用于与图像块一起工作的非常精细的粒度等级。

[0413] 此外，不管图像块大小如何，可以使用各种类型的铺放模式。例如，图 30 示出了一实施例，其中 R 帧 3001-3004 的每一个中都使用了多个 I- 图像块。使用了旋转模式，其中在每个 R 帧散布 I- 图像块，使得每四个 R 帧生成一个完整的 I- 帧。以这种方式散布 I- 图像块将降低分组丢失的影响（将丢失限制到显示器的小区域内）。

[0414] 图像块的大小还可为基本压缩算法的整体自然结构。例如，如果在一个实施例中使用 H.264 压缩算法，图像块被设定为 H.264 “片 (slice)” 的大小。这允许此处所述的技术易于集成到各种不同标准压缩算法的环境中，该算法例如是 H.264 和 MPEG-4。一旦图像块大小被设定为自然压缩结构，可实施如上所述的那些技术同样的技术。

[0415] 用于流倒回 (REWIND) 和回放操作的技术

[0416] 如之前连同图 15 所描述的，由应用程序 / 游戏服务器 1521-1525 生成的未经压缩的视频 / 音频流 1529 可由共用硬件压缩 1530 以多种分辨率压缩，同时生成多个经压缩的视频 / 音频流 1539。例如，由应用程序 / 游戏服务器 1521 生成的视频 / 音频流可由共用硬件压缩 1530 以 1280×720×60fps 压缩，并作为出埠互联网通信 1599 经由出埠路由 1540 传送给用户。同样的那种视频 / 音频流同时可由共用硬件压缩 1530 按比例缩小到缩略图尺寸（例如 200×113），并经由路径 1552（或通过延迟缓冲器 1515）发送到应用程序 / 游戏服务器 1522，以显示为图 16 中缩略图集合的一个缩略图 1600。当缩略图 1600 经过图 17 中级尺寸 1700 而被放大到图 18 中的尺寸 1800(1280×720×60fps) 时，并非对缩略图流进行解压缩，应用程序 / 游戏服务器 1522 可解压缩发送到应用程序 / 游戏服务器 1521 的用户的 1280×720×60fps 流的复本，并且缩放到较高分辨率的视频，便如同其被从缩略图尺寸而缩放到 1280×720 尺寸那样。该方法具有两次重用 1280×720 压缩流的优点。但其具有若干缺点 ：(a) 发送到用户的经压缩的视频流可能在图像质量上变化，如果用户互联

网连接的数据吞吐量变化导致由应用程序 / 游戏服务器 1522 的“出席观看 (spectating)”的用户观看的图像质量变化，即使用户的互联网连接并未变化，(b) 应用程序 / 游戏服务器 1522 将不得不使用处理资源来解压缩整个 1280×720 图像，而且之后缩放此图像（可能应用重采样滤波器）以在缩放期间显示小得多的尺寸（例如 640×360），(c) 如果由于有限的互联网连接带宽和 / 或丢失的 / 损坏的分组而丢帧，出席观看的用户“倒回”并“暂停”延迟缓冲器 1515 中记录的视频，出席观看的用户将发现延迟缓冲器中的丢帧丢失（如果用户逐帧“步进”，这将会格外明显），以及 (d) 如果出席观看的用户倒回以延迟缓冲器中所记录的视频中寻找特定帧，则应用程序 / 游戏服务器 1522 将不得不在延迟缓冲器中所记录的视频流中寻找先于所寻帧的 I 帧或 I 图像块，之后解压缩所有的 P 帧 / 图像块，直到达到想要的帧。同样的限制将不仅存在于“出席观看”视频 / 音频流直播的用户，而且还存在于观看视频 / 音频流的存档复本（例如“精彩剪辑 (Brag Clip)”）复本的用户（包括生成视频 / 音频流的用户）。

[0417] 本发明的可选实施例通过用不止一种的尺寸和 / 或结构来压缩视频流来解决这些问题。如此处所描述的，一个流（“直播”流）基于网络连接的特征（例如数据带宽、分组可靠性）和用户的本地客户端能力（例如解压缩能力、显示器分辨率）而被最佳地压缩并流向终端用户。其他流以高质量、一个或更多分辨率、以及可经受视频回放检验的结构来压缩（此处被称为“HQ”流），这种 HQ 流被路由和存储到服务器中心 210 中。例如在一实施例中，所述 HQ 压缩流存储于 RAID 磁盘阵列 1515 并用于提供诸如暂停、倒回的功能以及其他回放功能（例如“精彩剪辑”，其可发布给其他用户进行观看）。

[0418] 如图 31a 所示，本发明的一个实施例包括解码器 3100，其至少能够以至少两种格式压缩视频流 :一种格式 3110 为周期性包括 I- 图像块或 I- 帧，一种格式 3111 不包括 I- 图像块和 I- 帧，除非由于流的中断或因为 I- 图像块或 I- 帧被确定为可能小于 I- 图像块或 I- 帧（如上所述）而有必要包括 I- 图像块和 I- 帧。例如，当播放视频游戏时传送到用户的“直播“流 3111 可仅仅使用 P- 帧（除非 I- 图像块或 I- 帧是必要的或者更小，如上所述）压缩。此外，此发明的编码器 3100 同时以第二种格式压缩直播视频流 3111，在一实施例中，所述第二种格式周期性地包括 I- 图像块或 I- 帧（或者相似类型的图像格式）。

[0419] 虽然上述实施例使用 I- 图像块、I- 帧、P- 图像块和 P- 帧，但本发明的基本原理并不限于任何特定的压缩算法。例如，可使用任意类型的图像格式取代 P- 图像块或 P- 帧，在该图像格式中，帧依赖于先前或随后的帧。同样地，可使用任意类型的图像格式代替上述的 I- 图像块或 I 帧，该图像格式并不依赖先前或随后帧。

[0420] 如上所提及的，HQ 流 3110 包括周期性的 I- 帧（例如在一实施例中，大约每 12 个帧）。这是很重要的，因为如果用户总是想将所存储视频流快速倒回到特定点，则需要 I- 图像块或 I- 帧。在只有 P- 帧（例如，序列的第一帧不是 I- 帧）的压缩流的情况下，解码器必须回到序列的第一帧（其可能有数小时长），之后将 P 帧解压缩，直至到达用户想倒回的点。通过 HQ 流 3110 内每 12 帧存储的 I- 帧，用户可决定倒回到特定点，且 HQ 流的最接近的前一 I- 帧仅比想要的帧靠前不超过 12 帧。即使解码器最大解码率是实时的（例如 :对于 60 帧 / 秒的流来说是一秒的 1/60），则离 I- 帧为 12（帧）/60（帧 / 秒）= 1/5 秒远。而在许多情况下，解码器能够比实时操作得更快，所以例如以 2× 实时的解码率，解码器能够在 6 帧的时间中解码 12 帧，用于“倒回“的延迟仅为一秒延迟的 1/10。不用说，如果最接近的

前一 I- 帧先于倒回点大量帧，即使快速解码器（例如 10× 实时）亦将具有不可接受的延迟（例如，它将花 1 小时 /10 = 6 分钟来做一次倒回）。在另一实施例中，使用周期性 I- 图像块，在此情况下，当用户寻求倒回时，解码器将会寻找先于倒回点的最接近的前一 I- 图像块，之后着手从那点开始该图像块的解码，直到解码所有图像块到该倒回点。虽然周期性 I- 图像块或 I- 帧与完全没有 I- 帧相比会导致更低效的压缩，但是主机服务 210 一般具有足够多的本地可用带宽和存储能力来管理 HQ 流。

[0421] 在另一实施例中，编码器 3100 编码具有周期性 I- 图像块或 I- 帧的 HQ 流，该 I- 图像块或 I- 帧跟随有 P- 图像块或 P- 帧，如前所述，且之前还存在 B- 图像块或 B- 帧。如之前所述，B- 帧是先前于 I- 帧的帧，且基于在时间上往回 (backward) 工作时与 I- 帧的帧差异。B- 图像块是图像块副本，先前于 I- 图像块且基于时间上往回 (backward) 工作时与 I- 图像块的帧差异。此该实施例中，如果想要的倒回点是 B- 帧（或包含 B- 图像块），则解码器会寻找最接近的下一 I- 帧或 I- 图像块，并在时间上往回解码，直到想要的倒回点被解码，之后视频回放从该点向前进行，解码器将逐帧向前解码 B- 帧、I- 帧和 P- 帧（或它们的图像块副本）。除了 I 和 P 类型之外，使用 B- 帧或 B 图像块的优点在于 :在给定压缩比下，通常能够达到更高的质量。

[0422] 在再一个实施例中，编码器 3100 将 HQ 流编码为全部的 I- 帧。此方法的优点在于 :每个倒回点都是 I- 帧，从而为了达到该倒回点，不需要解码其他帧。缺点在于 :和 I、P 或 I、P、B 流编码相比，经压缩的数据速率将会非常高。

[0423] 其他视频流回放操作（例如 :快速或慢速倒回，快速或慢速前向等），一般由周期 I- 帧或 I- 图像块（单独或组合使用 P 和 / 或 B 副本）更加实际地完成，因为在每种情况下，与在时间上逐帧前向相比，流均以不同的帧顺序被回放，从而解码器需要寻找和解码序列中特定的（通常是任意的）帧。例如，在非常快速的快进（例如 100× 速度）情况下，所播放的每下一帧均在先前帧 100 帧之后。即使利用具有以 10× 实时运行且在 1 个帧时间内解码 10 帧的解码器，它将仍然是 10×，太慢而不能获得 100× 的快进。然而，利用如前所述的周期 I- 帧或 I- 图像块，解码器能寻找最接近下一次播放所需的帧的适用 I- 帧和 I- 图像块，且仅解码至目标帧的点的中间帧或图像块。

[0424] 在另一实施例中，I 帧以一致的周期数（例如总是每 8 个帧）被编码入 HQ 流中，用户可用于快进和倒回的倍速就是 I- 帧周期数的精确倍数，所述快进和倒回比 I- 帧速率更快。例如，如果 I- 帧周期数是 8 帧，则使得用户可用的快进或倒回速度可能是 1×，2×，3×，4×，8×，16×，64× 和 128× 以及 256×。对于比 I- 帧周期数更快的速度，解码器将首先以该速度向前跳转到一定数量的帧之前的最接近的 I- 帧（例如，如果当前所显示的帧比 I- 帧领先 3 帧，则以 128×，解码器将跳到靠前 128+3 帧的帧），之后对于每个后续的帧来说，解码器将随着所选速度（例如以 128× 的所选速度，解码器将跳转 128 帧）跳转精确数量的帧，每次它将精确地落在 I- 帧上。因此假定比 I- 帧周期数快的所有速度是 I- 帧周期数的精确倍数，解码器将永不需要解码任何先前或后继帧来寻找想要的帧，且仅需要针对每一显示的帧，解码一个 I- 帧。对于比 I- 帧周期数更慢的速度（例如 :1×，2×，3×，4×），或者对于作为 I- 帧周期数的非倍数的更快速度，对于所显示的每个帧，解码器寻求一帧，该帧使得显示想要的帧所需要的额外新解码的帧的数量最少，该帧可为未解码的 I- 帧或且仍为解码形式的已解码的帧（在 RAM 或其他快速存储器中），之后，如果必要，可解码中

间帧，直到想要的帧被解码和显示。例如以 4× 的速度快进，在具有 8×I- 帧周期数的 I、P 编码序列中，如果当前帧是靠后 I- 帧 1 帧的 P- 帧，而将被显示的想要的帧为 4 帧之后帧，其为之前的 I- 帧之后的第 5 个 P- 帧。如果当前显示的帧（其刚被解码）被用作起始点，该解码器将需要再解码 4 个 P- 帧来显示想要的帧。如果使用之前的 I- 帧，为了显示想要的帧，解码器将需要解码 6 帧（I- 帧和随后的 5 个 P- 帧）。（明显地，在这种情况下，使用当前显示的帧来最小化需额外解码的帧是有利的）。之后，下一个将被解码的靠前 4 帧，为 I- 帧之后的第一个 P- 帧。这种情况下，如果当前解码的帧被用作起始点，解码器还需解码 4 个帧（2 个 P- 帧、一个 I- 帧和一个 P- 帧）。但是，如果使用下一个 I- 帧来替代，解码器将仅需要解码 I- 帧和后续的 P- 帧。（明显地，在这种情况下，使用下一个 I- 帧作为起始点来最小化需额外解码的帧是有利的。）因此，在此实例中，解码器可在使用当前解码的帧作为起始点与使用随后的 I- 帧作为起始点之间交替。作为一主要原则，不管 HQ 视频流回放模式（快进、倒回或步进）和速度如何，对于该回放模式和速度下所显示的每个后续帧，解码器将从一使得显示想要的帧所需要的新解码的帧的数量最少的帧开始，即 I- 帧或之前解码的帧。

[0425] 如图 31b 所示，主机服务 210 的一个实施例包括用于管理重放 HQ 流 3110 的用户请求的流重放逻辑 3112。流重放逻辑 3112 接收客户请求（该请求含有视频回放命令（例如：暂停、倒回、从一指定点回放等））、翻译该命令并从指定点解码 HQ 流 3110（开始于 I- 帧或之前解码的帧，视情况而定，之后向前或向后进行到指定点）。在一实施例中，向编码器 3100 提供经解码的 HQ 流（可能是同一个编码器 3100，如果其一次能够编码不止一个流，或者是单独的编码器 3100），以便其可被重新压缩（使用此处所述的技术）并被传送给客户端设备 205。客户端设备上的解码器 3102 之后如上所述地解码并再现该流。

[0426] 在一实施例中，流重放逻辑 3112 并不解码 HQ 流并在之后使编码器 3100 重新编码该流。相反地，其简单地从指定点将 HQ 流 3110 直接流向客户端设备 205。客户端设备 205 上的解码器 3102 之后解码 HQ 流。因为此处所述的回放功能一般不具有像播放实时视频游戏那样的低延时需求（例如：如果玩家只是简单地查看先前的游戏播放，而不是主动玩该游戏），所增加的普通高质量 HQ 流一般固有的延时可导致可接受的终端用户体验（例如，具有更长延时但是高质量的视频）。

[0427] 以示例而非限制的方式，如果用户正在玩视频游戏，则编码器 3100 提供具有本质上所有 P- 帧的直播流，所述 P- 帧针对用户的连接和本地客户端而被优化（例如，大约 1.4Mbps，640×360 的分辨率）。同时，编码器 3100 还在主机服务 310 内将视频流压缩为 HQ 流 3100，并将该 HQ 流存储在本地数码视频解码器 RAID 阵列上，例如以 1280×720、10Mbps，每 12 帧具有 I- 帧。如果用户击“暂停”按钮，则游戏将暂停在客户最后解码的帧上，且屏幕将冻结。之后如果用户击“倒回”按钮，流重放逻辑 3112 将从 DVRRAID 中读取开始于最近的 I- 帧或可用的已被解码的帧的 HQ 流 3110，如上所述。若有需要的话，流重放逻辑 3112 将解压缩中间的 P 或 B 帧，若有需要的话并对帧进行重新排序，以便以想要的倒回速度倒退 (backward) 回放序列，之后调整（使用本领域公知的现有技术的图像缩放技术）想要解码的、打算从以 1280×720 进行显示变为以 640×360 进行显示的流，之后直播流编码器 3100 将以 640×360 的分辨率重新压缩重排序的流并将其传送给用户。如果用户再次暂停，并单步浏览视频以近距离地观看序列，DVR RAID 上的 HQ 流 3110 可具有可用于单步执行的每个

帧（即使原始直播流可能由于此处所述众多原因中的任何原因而丢帧）。此外，HQ 流中每一点上的视频回放的质量将非常高，而在直播流中可能存在某些点因带宽受损而导致经压缩的图像的质量的临时降低。当受损图像质量持续一简短时段或在运动图像中时，其是可被用户接受的，如果用户在特定的帧处停止（或者缓慢地单步执行）并仔细近距离地研究帧，受损质量可能不被接受。通过指定 HQ 流内的点（例如，前 2 分钟），还向用户提供快进或跳特定点的能力。在直播视频流仅是 P- 帧或者很少（或者不可预测的）有 I- 帧的情况下，所有这些操作都是无法以其完全通用性及高质量执行的。

[0428] 在一实施例中，向用户提供视频窗口（未示出，诸如 Apple QuickTime 或 Adobe Flash 视频窗口），该视频窗口具有“刷子 (scrubber)”（即，左 - 右滑动块控制），其早在 HQ 流存储视频之前，就允许用户向前或向后扫描浏览视频流。虽然对用户来说便好像是他或她正“刷”过直播流，事实上他或她正“刷”过所存储的 HQ 流 3110，该 HQ 流 3110 之后被调整大小并被重新压缩为直播流。此外，如前所述，如果同一时刻的其他任何人或不同时刻的用户观看 HQ 流，该 HQ 流可以以比当同时编码 HQ 流时的直播流的分辨率更高的（或更低）的分辨率被观看，而质量将和观看者的直播流的质量一样高，潜在地可达到 HQ 流的质量。

[0429] 因此，通过同时编码直播流（如此处所述，以针对其低延时、带宽和分组容错要求的合适方式）和具有其高质量、流回放操作需求的 HQ 流，从而向用户提供这两种场景的想要的配置。而事实上，对用户来说高效透明的是 :存在两种被不同编码的不同流。从用户的角度看，该体验高度响应的，具有很低的延时，尽管运行于高度可变的且相对低带宽的互联网连接上，但数码视频记录 (DVR) 功能是非常高质量的，具有灵活的操作和灵活的速度。

[0430]

[0431] 因为上述技术，用户在在线游戏游玩或其他在线交互期间受益于直播和 HQ 视频流，不受直播流或 HQ 流中任一个的限制。

[0432] 图 31c 示出了执行上述操作的系统架构的一个实施例。如图所示，在此实施例中，编码器 3100 分别编码一系列的“直播”流 3121L、3122L 和 3125L 以及对应系列的“HQ”流 3121H1-H3、3122H1-H3 和 3125H1-H3。每个 HQ 流 H1 被全分辨率编码，而每个编码器 H2 和 H3 在编码之前将视频流缩放到的更小尺寸。例如，如果视频流是 1280×720 的分辨率，H1 将以 1280×720 的分辨率编码，而 H2 可缩放到 640×320 且以该分辨率编码，而 H3 可缩放到 320×180 并以该分辨率编码。可使用任意数量的同时的 Hn 缩放器 / 编码器，提供各种分辨率的多个同时 HQ 编码。

[0433] 每个直播流响应于经由入埠互联网连接 3101 接收的信道反馈信号 3161、3162 和 3165 而工作，如上所述（例如参见图 25-26 中反馈信号 2501 和 2601 的论述）。直播流经由出埠路由逻辑 3140 在互联网（或其他的网络）上向外传送。直播压缩器 3121L-3125L 包括基于信道反馈（包括缩放、丢帧等）而调节 (adapting) 压缩视频流的逻辑。

[0434] 入埠路由逻辑 3141 和 1502 经由信号路径 3151 将 HQ 流路由到内部延迟缓冲器（例如 RAID 阵列 3115）或其他数据存储设备，和 / 或经由信号路径 3152 将 HQ 流反馈给用于额外处理的应用程序 / 游戏服务器和编码器 3100。如上所述，一旦请求，HQ 流 3121Hn-3125Hn 随后被流入到终端用户（例如参见图 31b 和关联内容）。

[0435] 在一实施例中，编码器 3100 被实施为图 15 中所示的共用硬件压缩逻辑 1530。在另一实施例中，某些或所有编码器和缩放器是单独的子系统。本发明的基本原理不限于任

何特定的缩放或压缩资源的共用或硬件 / 软件配置。

[0436] 图 31c 配置的优点在于 :需要小于全尺寸视频窗口的视频窗口的应用程序 / 游戏服务器 3121-3125 将无需处理和解压缩全尺寸的窗口。此外,需要介于中间的窗口尺寸的应用程序 / 游戏服务器 3121-3125 可接收接近于所要窗口尺寸的经压缩的流,之后按比例增大或缩小至想要的窗口尺寸。此外,如果多个应用程序 / 游戏服务器 3121-3125 向另一个应用程序 / 游戏服务器 3121-3125 请求同样尺寸的视频流,入埠路由 3141 可执行 IP 多播技术,例如本领域公知的那些技术,并将所请求的流一次性广播到多个应用程序 / 游戏服务器 3121-3125 上,无需将单独的流广播到做出请求的每个应用程序 / 游戏服务器。如果接收广播的应用程序 / 游戏服务器改变视频窗口的尺寸,其可切换至不同视频尺寸的广播上。因此任意大数量的用户能够同时观看应用程序 / 游戏服务器视频流,每一用户可灵活缩放其视频窗口,并总是获得缩放到接近想要窗口尺寸的视频流的好处。

[0437] 图 31c 所示方法的一个缺点在于 :在主机服务 210 的许多实际实施中,从未有出现过所有的压缩 HQ 流被一起观看的情况,更不用说所有压缩 HQ 流的所有尺寸。当编码器 3100 被实现为共用资源 (例如 :缩放器 / 压缩器,可实现为软件或硬件) 时,减轻了此浪费。但是,由于涉及带宽,将大量的未压缩流连接到公共的共用资源时可能会存在实际的问题。例如,每个 1080p60 流几乎是 3Gbps,其甚至远远超过千兆位以太网。下面的可选实施例解决了该问题。

[0438] 图 31d 示出了主机服务 210 的可选实施例,其中每个应用程序 / 游戏服务器 3121-3125 具有两个分配给它的压缩器 :(1) 直播流压缩器 3121L-3125L,其基于信道反馈 3161-3165 调节经压缩的视频流,以及 (2) 输出全分辨率的 HQ 流的 HQ 流压缩器,如上所述。值得注意的是,直播压缩器是动态和自适应的,与客户端 205 使用双向通信,而 HQ 流是非自适应性的和单向的。流之间的其他差异为 :直播流质量可以动态地变化,依赖于信道条件和视频资料的特性。一些帧可能是低质量的,可能存在丢帧。此外,直播流可能几乎全部是 P- 帧或 P- 图像块,少有 I- 帧和 I- 图像块出现。HQ 流和直播流相比一般具有更高的数据速率,其将提供一致的高质量,不会丢弃任何帧。HQ 流可能全是 I- 帧,或者可具有频繁的和 / 或规律的 I- 帧或 I- 图像块。HQ 流还可包括 B- 帧或 B- 图像块。

[0439] 在一实施例中,共用视频缩放和重新压缩 3142(下面详述) 只选择某些 HQ 视频流 3121H1-3125H1,在发送到如前所述的用于路由的入埠路由 3141 之前,其将以一个或多个不同的分辨率而被缩放和重新压缩。其他 HQ 视频流或者以全尺寸通过之前所述的用于路由的入埠路由 3141,或者根本不通过。在一实施例中,基于是否有请求特定分辨率 (或者接近于缩放的分辨率或全分辨率) 的特定 HQ 流的应用程序 / 游戏服务器 3121-3125 来做出该决定 :哪些 HQ 流被缩放和重新压缩,和 / 或哪些 HQ 流完全通过。通过该方法,只有被缩放和重新压缩的 HQ 流 (或者潜在地完全通过) 是实际需要的 HQ 流。在主机服务 210 的众多应用中,这导致缩放和压缩资源的显著减少。此外,假定每个 HQ 流至少由压缩器 3121H1-3125H1 以全分辨率压缩,和若其是接受的未经压缩的视频相比,被路由所需的带宽以及共用视频缩放和重新压缩 3142 中的带宽会显著地降低。例如,3Gbps 未经压缩的 1080p60 流将被压缩到 10Mbps,且仍然保留非常高的质量。因此,利用千兆位以太网连接,虽然不能传送 (carry) 一个未压缩的 3Gbps 视频流,但可传送许多的 10Mbps 视频流,在质量上并不会具有明显的减少。

[0440] 图 31f 示出了共用视频缩放和重新压缩 3142 以及大量 HQ 视频压缩器 HQ 3121H1-3131H1 的细节。内部路由 3192 根据来自应用程序 / 游戏服务器 3121-3125 的将特定视频流缩放到特定尺寸的请求，一般从 HQ 视频压缩器 HQ 3121H1-3131H1 中选择一个子集的经压缩 HQ 流。如果所请求的流要被缩放，则通过解压缩器 3161-3164 路由该所选流子集中的流，或者如果所请求的流是全分辨率，则在未经缩放的视频路径 3196 上路由该所选流子集中的流。要被缩放的流由解压缩器 3161-3164 解压缩为未经压缩的视频，之后由缩放器 3171-3174 将每一个缩放到所请求的尺寸，之后由压缩器 3181-3184 将每一个压缩。注意 :如果以不止一种分辨率请求特定的 HQ 流，则内部路由 3192 将该流多播（使用本领域从业者公知的 IP 多播技术）到一个或多个解压缩器 3161-3164 上和（如果所请求的尺寸之一是全分辨率）出埠路由 3193 上。所有所请求的流，无论其被缩放（从压缩器 3181-3184）与否（从内部路由 3192），之后被发送到出埠路由 3193。路由 3193 之后将每个所请求的流发送到请求它的应用程序 / 游戏服务器 3121-3125 上。在一实施例中，如果不止一个应用程序 / 游戏服务器请求同样分辨率的同一流，则出埠路由 3193 将该流多播到做出该请求的所有应用程序 / 游戏服务器 3121-3125 上。

[0441] 在共用视频缩放和重新压缩 3142 的当前优选实施例中，所述路由通过使用千兆位以太网开关来实施，所述解压缩、缩放以及压缩通过执行每一功能的离散的专门的半导体器件来实施。同样的功能可以以更高集成度实施在硬件中或由非常快的处理器实施。

[0442] 图 31e 显示了主机服务 210 的另一个实施例，其中之前所述的延迟缓冲器 3115 的功能在共用视频延迟缓冲、缩放和解压缩子系统 3143 中实现。图 31g 示出了子系统 3143 的细节。子系统 3143 的操作类似于图 31f 中所示的子系统 3142 的操作，除了 3191 首先根据来自应用程序 / 游戏服务器 3121-3125 的请求，选择哪些 HQ 视频流将被路由，之后请求延迟的 HQ 流通过延迟缓冲器 3194（在本实施例被实施为 RAID 阵列（但其可被实施为具有足够带宽和能力的任何存储媒体））被路由，未被请求延迟的流通过未经延迟视频路径 3195 被路由。之后，延迟缓冲器 3194 和未经延迟视频 3195 的输出基于所请求的流将被缩放或不被缩放而由内部路由 3192 路由。经缩放的流通过解压缩器 3161-3164、缩放器 3171-3174 和压缩器 3181-3184 而被路由至出埠路由 3193。未经缩放的视频 3196 也被发送至出埠路由 3193，之后，以之前图 31f 的子系统 3142 中所述的同样方式，出埠路由 3193 以单播或多播模式将视频发送到应用程序 / 游戏服务器。

[0443] 视频延迟缓冲、缩放和解压缩子系统 3143 的另一个实施例示于图 31h 中。在该实施例中，针对每个 HQ 流提供单独的延迟缓冲器 HQ3121D-HQ 3131D。假定 RAM 和闪存 ROM 的成本快速下降，其可用于延迟单独的经压缩的视频流，与具有共用延迟缓冲器 3194 相比，这将最终使得成本更低和 / 或更灵活。或者，在再一实施例中，单个延迟缓冲器 3197（以虚线示出）可以以高性能集合资源（例如，非常快的 RAM、闪存或磁盘）单独为所有 HQ 流提供延迟。在任何一种情况中，每个延迟缓冲器 HQ3121D-3131D 能够对来自 HQ 视频源的进行可变地延迟，或者无延迟地通过该流。在另一实施例中，每个延迟缓冲器能够提供具有不同延迟量的多个流。所有的延迟或未经延迟均由应用程序 / 游戏服务器 3121-3125 来请求。在所有这些情况中，延迟和未经延迟的视频流 3198 被发送到内部路由 3192，并通过之前所述的与图 31g 有关的子系统 3143 的其他部分继续下去。

[0444] 与各种图 31n 相关的前述实施例中，注意 :直播流使用双向连接，且是为特定用户

量身定做的，具有最小的延时。HQ 流使用单向连接，且为单播和多播。注意：虽然多播功能在这些图中被示为单个单元，例如可被实施为千兆位以太网开关，但在大规模系统中，多播功能可能通过由多个开关构成的树结构来实施。实际上，对于来自高等级视频游戏玩家的视频流，即为这一情况，该玩家的 HQ 流被上百万用户同时观看。在这种情况下，在广播被多播的 HQ 流的连续阶段，有可能有大量的单独开关。

[0445] 出于诊断的目的，且为了向用户提供反馈（例如，让用户知道他的游戏玩法表演是多么的流行），在一实施例中，主机服务 210 将明了每个应用程序 / 游戏服务器 3121-3125 的视频流同时有多少观众。这可由用于特定视频流的应用程序 / 游戏服务器通过保持计算有效请求的数量来实现。因此，同时具有 100000 名观众的玩家将知道他或她的游戏玩法是非常流行的，它将为游戏玩家造成激励，以做出更好的成绩并吸引观众。当（例如视频游戏锦标赛比赛的）视频流具有非常高度收视率时，可能需要讲解员来在电视游戏比赛期间进行讲解，这样观看该多播的某些用户或所有用户能够听到他们的解说词。

[0446] 应用程序 / 游戏服务器上运行的应用程序和游戏将被提供应用程序编程接口 (API)，其中应用程序和 / 或游戏可提交对具有特定特征（例如，分辨率和延迟量）的特定视频流的请求。此外，服从于应用程序 / 游戏服务器上运行的操作环境或者图 4a 的主机服务控制系统 401 的这些 API 可以因为各种原因拒绝这些请求。例如，所请求的视频流可能具有某些许可权限限制（例如，使得它仅可由单个观众观看，而不会被广播给其他人），可能有订阅限制（例如，观众可能必须付费才有权观看该流），可能有年龄限制（例如，观众可能必须有 18 岁才能观看该流），可能有保密限制（例如，使用该应用程序或玩该游戏的人可能仅限定所选数量或类别的观众可以观看（例如他或她的“朋友”），或者可能根本不允许观看），可能有需要资料被延迟的限制（例如，如果用户正在玩隐蔽类游戏，其中他或她的位置可被暴露）。有任意数量的可能限制流的观看的其他限制。在任意的这些情况中，应用程序 / 游戏服务器的请求将被拒绝，出具该拒绝的原因，在一实施例中，出具可选方案（例如，申明须交什么费用来进行订阅），通过该可选方案，所述请求可被接受。

[0447] 之前任意实施例中存储于延迟缓冲器中的 HQ 视频流可被导出到主机服务 210 外的其他目的中。例如，特定的感兴趣的视频流可由应用程序 / 游戏服务器请求（一般由用户请求）导出到 YouTube 上。这种情况下，视频流和合适的描述信息（例如，玩家的姓名、游戏、时间、得分等）一起以与 YouTube 格式一致的格式通过互联网而被传送。这可通过通过在单独流中将解说音频多播到请求这种解说的所有游戏 / 应用程序服务器 3121-3125 上来实现。通过使用本领域从业者公知的音频混合技术，游戏 / 应用程序服务器将解说的音频与发送到用户场所 211 的音频流相合并。很可能存在多个讲解员（例如，具有不同观点，或者用不同语言），而用户能够在其中间选择。

[0448] 以相似的方式，分离的音频流可被混合，或者起作为主机服务 210 中特定视频流（或单独的流）的音频轨的替换，混合或替换实时的或来自延迟缓冲器的视频流中的音频。这种音频可以是解说词或叙述，或者其能够为视频流中角色提供声音。这将使引擎电影 (Machinima)（来自视频游戏视频流的用户生成动画）很容易被用户创建。

[0449] 全篇文档描述的视频流被示为：从应用程序 / 游戏服务器的视频输出中被捕捉的视频流，之后该视频流被流出和 / 或延迟并以各种方式而被再使用或发布。同样的延迟缓冲器可用于保留来自非应用程序 / 游戏服务器源的视频资料并提供用于回放和发布的相

同灵活度，具有适当的限制。这种源包括来自电视台的直播馈送（或者无线电通信，或者非无线电通信，例如CNN，或者付费的，例如HBO，或者免费的）。这种资源还包括预录制的电影或电视节目、家庭电影、广告以及直播视频电视会议馈送。直播馈送可像游戏/应用服务器的直播输出那样被处理。预录制的资料可像延迟缓冲器的输出那样被处理。

[0450] 在一个实施例中，本文中所说明的各种功能模组及相关联的步骤可由含有用于执行所述步骤的固线式逻辑的特定硬件组件（诸如，特殊用途集成电路（“ASIC”））或由被编程的计算机组件与定制硬件组件的任何组合来执行。

[0451] 在一个实施例中，可将所述模组实施于诸如Texas仪器的TMS320x架构（例如，TMS320C6000、TMS320C5000，... 等）的可编程数字信号处理器（“DSP”）上。可使用各种不同的DSP，同时仍遵守所述基本原理。

[0452] 实施例可包括如上文所阐述的各种步骤。所述步骤可体现于引起通用或专用处理器执行特定步骤的机器可执行指令中。已将与这些基本原理无关的各种元件（诸如，计算机存储器、硬碟机、输入设备）从图中省去以避免混淆相关方面。

[0453] 所公开的标的物的要素也可作为用于储存机器可执行指令的机器介质来提供。机器可读介质可包括（但不限于）闪存、光碟、CD-ROM、DVD ROM、RAM、EPROM、EEPROM、磁卡或光卡、传播媒体或适合于储存电子指令的其他类型的机器可读介质。举例而言，本发明可作为计算机程序来下载，该计算机程序可经由通信链路（例如，数据机或网络连接）借助于体现在载波或其他传播媒体中的数据信号而从远程计算机（例如，服务器）传送到请求计算机（例如，客户端）。

[0454] 还应理解，所公开的标的物的要素也可作为计算机程序产品来提供，该计算机程序产品可包括在上面储存有指令的机器可读介质，所述指令可用于程序化计算机（例如，处理器或其他电子设备）以执行一序列操作。或者，所述操作可通过硬件与软件的组合来执行。机器可读介质可包括（但不限于）软盘、光盘、CD-ROM，及磁光盘、ROM、RAM、EPROM、EEPROM、磁卡或光卡、传播媒体或适合于储存电子指令的其他类型的媒体/机器可读介质。举例而言，所公开的标的物的要素可作为计算机程序产品来下载，其中程序可经由通信链路（例如，数据机或网络连接）借助于体现于载波或其他传播媒体中的数据信号而自远程计算机或电子设备传送至请求过程。

[0455] 另外，尽管已结合特定实施例描述所公开的标的物，但众多修改及变更将适当地处于本公开的范畴内。因此，说明书及图式应视为说明性的而非限制性的意义。

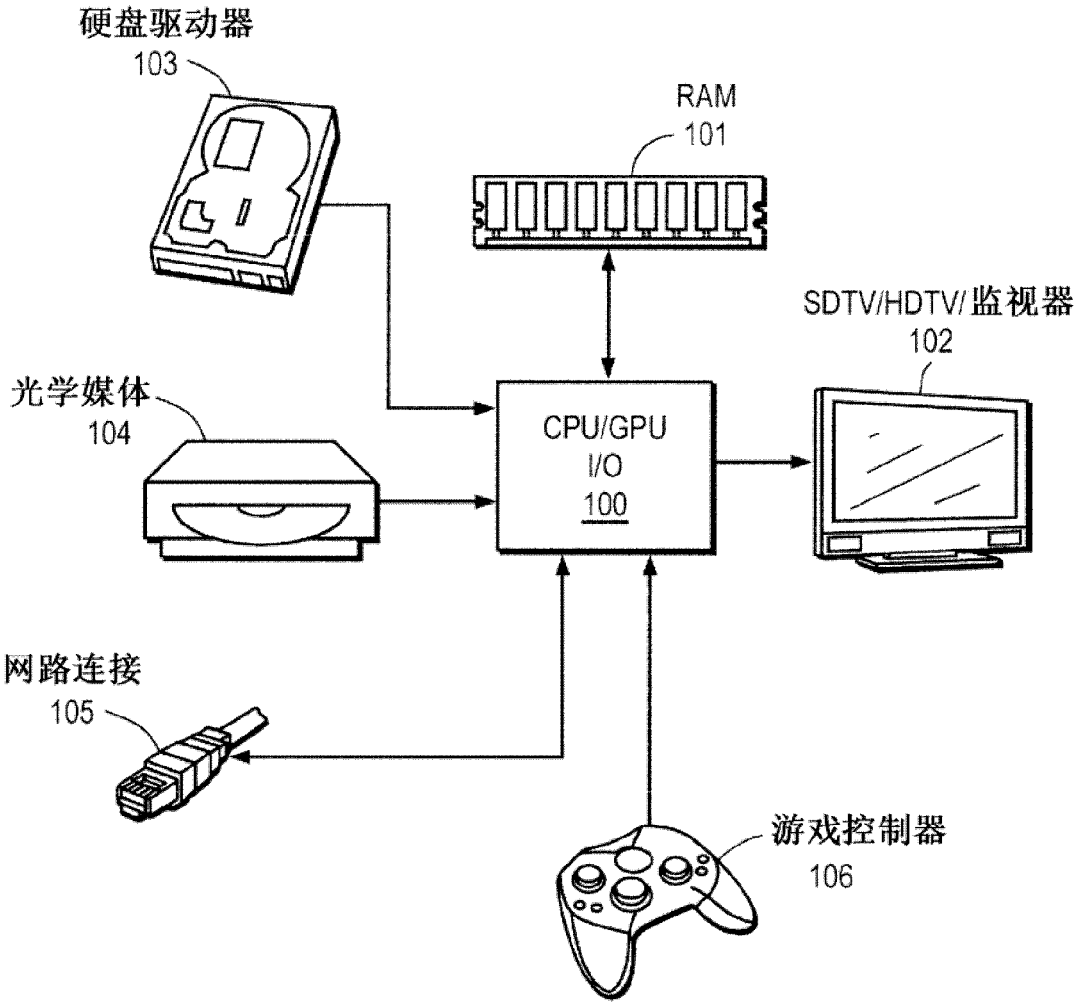

硬盘驱动器
103
RAM
101
SDTV/HDTV/监视器
102
光学媒体
104
CPU/GPU
I/O
100
网路连接
105
游戏控制器
106

图 1（现有技术）

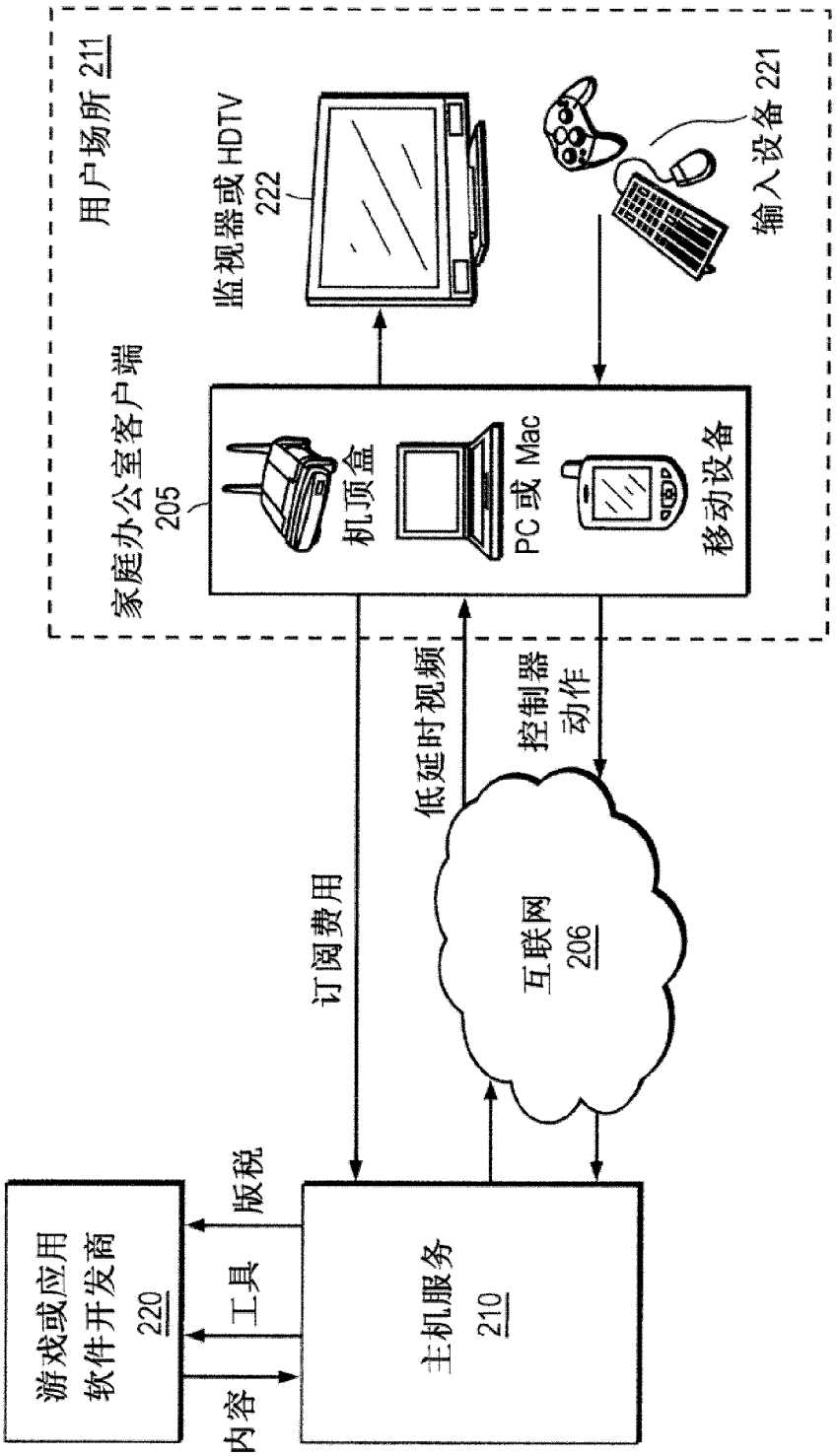

用户场所 211
家庭办公室客户端
205
监视器或 HDTV
222
输入设备 221
机顶盒
PC 或 Mac
移动设备
低延时视频
控制器
动作
订阅费用
互联网
206
游戏或应用
软件开发商
220
版税
工具
内容
主机服务
210

图 2a

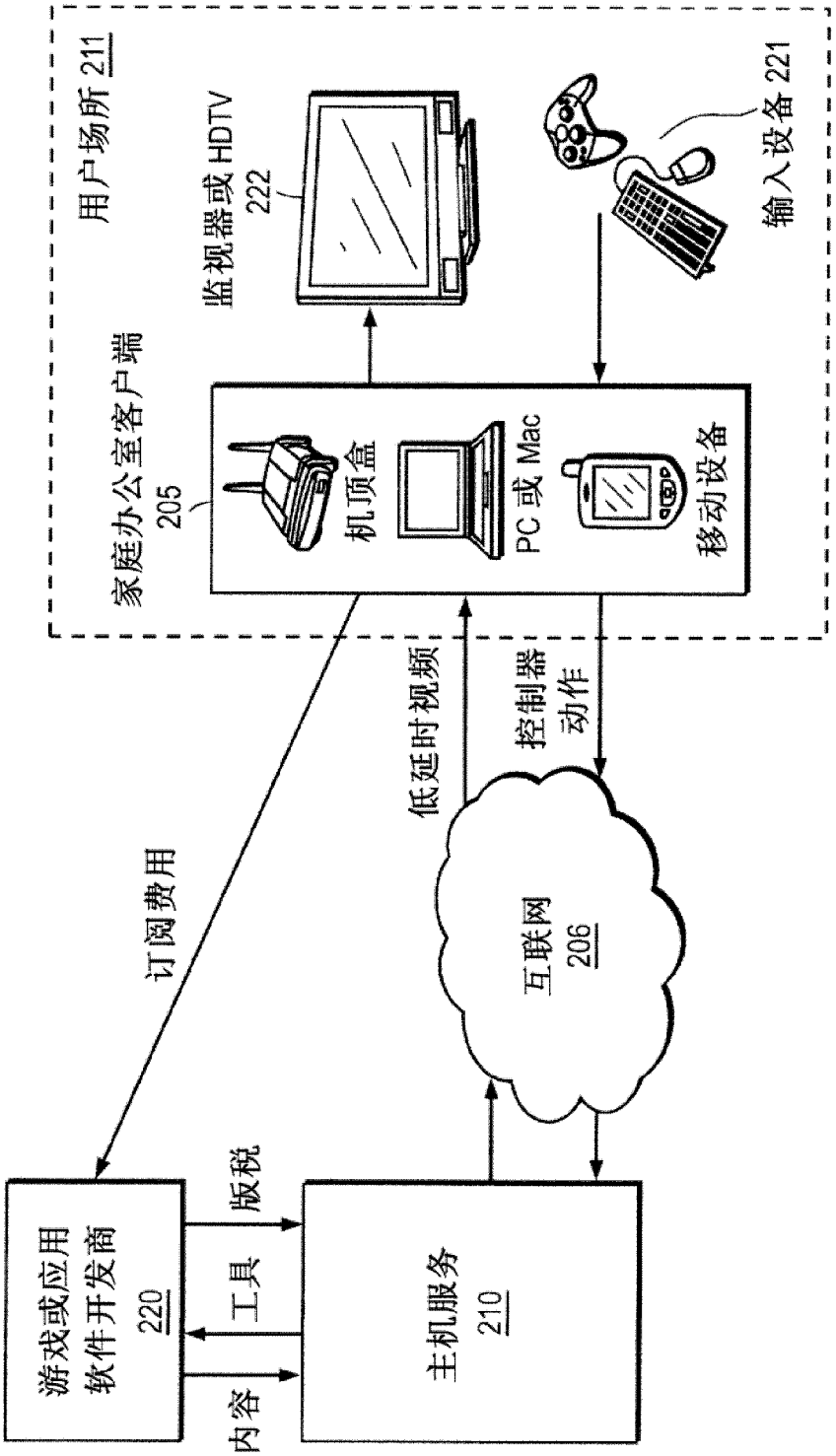

用户场所 211
家庭办公室客户端
205
监视器或 HDTV
222
输入设备 221
机顶盒
PC 或 Mac
移动设备
低延时视频
控制器
动作
订阅费用
互联网
206
游戏或应用
软件开发商
220
版税
工具
内容
主机服务
210

图 2b

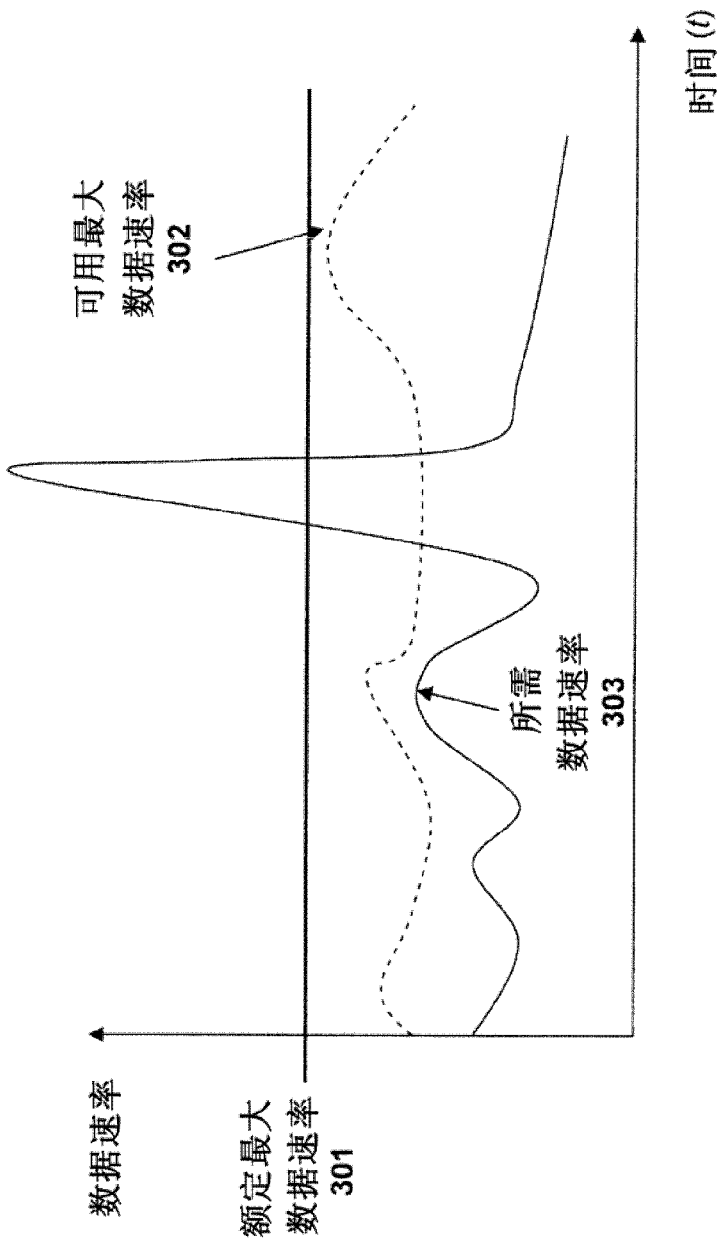


图 3

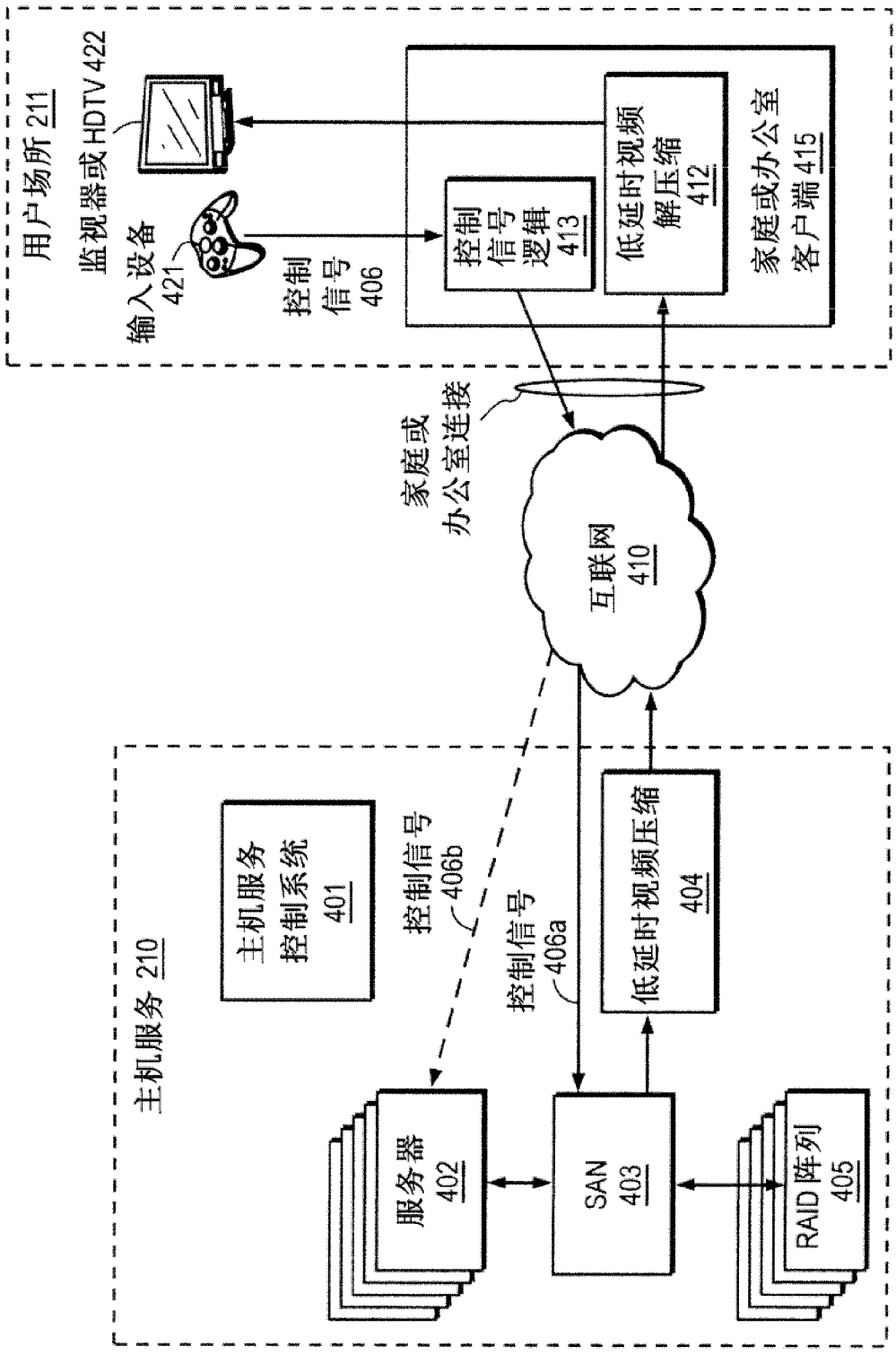

用户场所 211
监视器或 HDTV 422
输入设备 421
控制信号 406
控制信号逻辑 413
低延时视频解压缩 412
家庭或办公室客户端 415
家庭或办公室连接
互联网 410
主机服务 210
主机服务控制系统 401
控制信号 406b
控制信号 406a
低延时视频压缩 404
服务器 402
SAN 403
RAID 阵列 405

图 4a

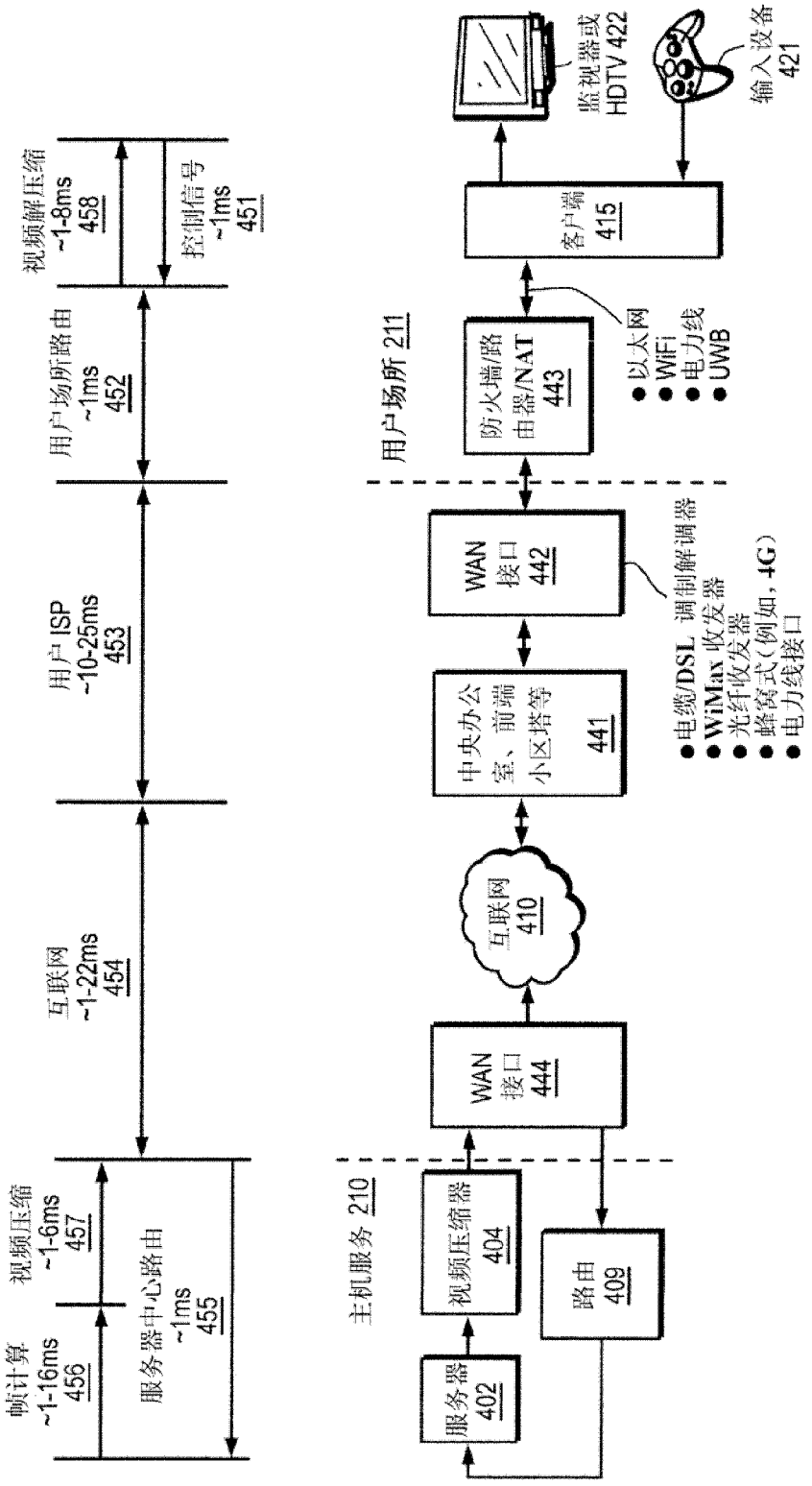

帧计算
~1-16ms
456
视频压缩
~1-6ms
457
服务器中心路由
~1ms
455
互联网
~1-22ms
454
用户 ISP
~10-25ms
453
用户场所路由
~1ms
452
视频解压缩
~1-8ms
458
控制信号
~1ms
451
主机服务 210
服务器
402
视频压缩器
404
路由
409
WAN
接口
444
互联网
410
中央办公
室、前端
小区塔等
441
WAN
接口
442
用户场所 211
防火墙/路
由器/NAT
443
客户端
415
监视器或
HDTV 422
输入设备
421
以太网
WiFi
电力线
UWB
电缆/DSL 调制解调器
WiMax 收发器
光纤收发器
蜂窝式(例如，4G)
电力线接口

图 4b

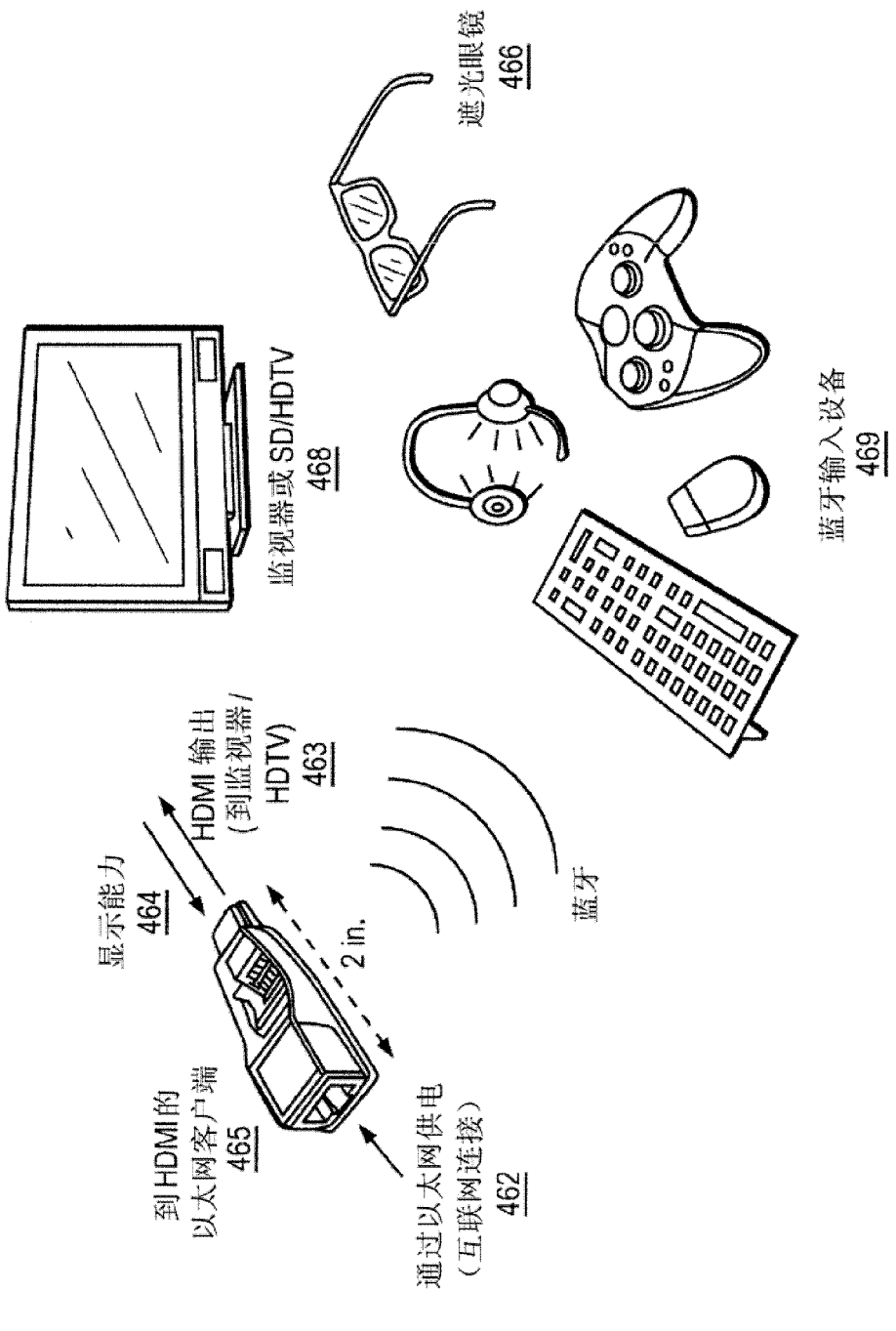

遮光眼镜
466
监视器或 SD/HDTV
468
蓝牙输入设备
469
显示能力
464
HDMI 输出
(到监视器/
HDTV)
463
2 in.
蓝牙
到 HDMI 的
以太网客户端
465
通过以太网供电
(互联网连接)
462

图 4c

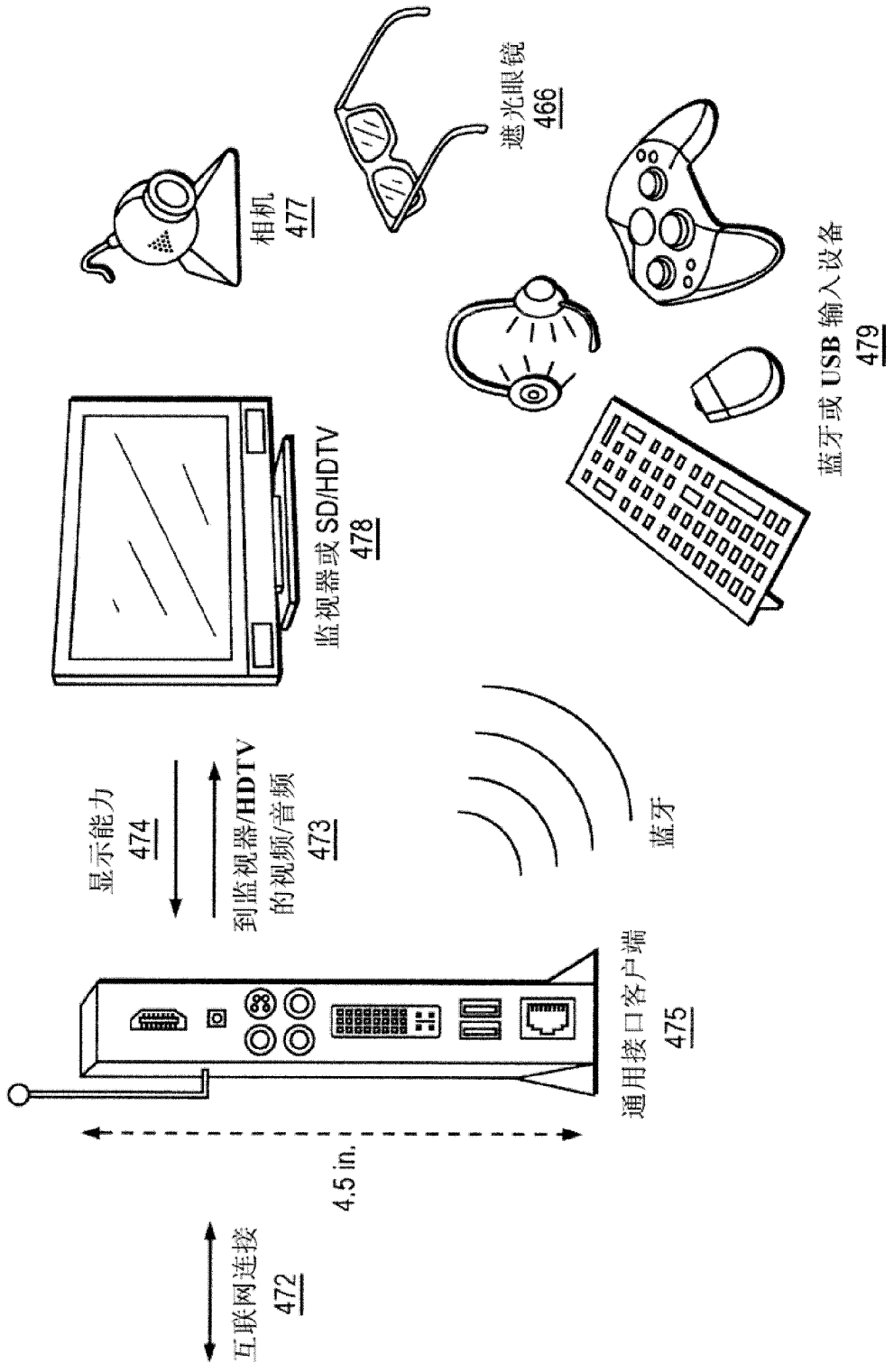

互联网连接
472
4.5 in.
通用接口客户端
475
蓝牙
显示能力
474
到监视器/HDTV
的视频/音频
473
监视器或 SD/HDTV
478
相机
477
遮光眼镜
466
蓝牙或 USB 输入设备
479

图 4d

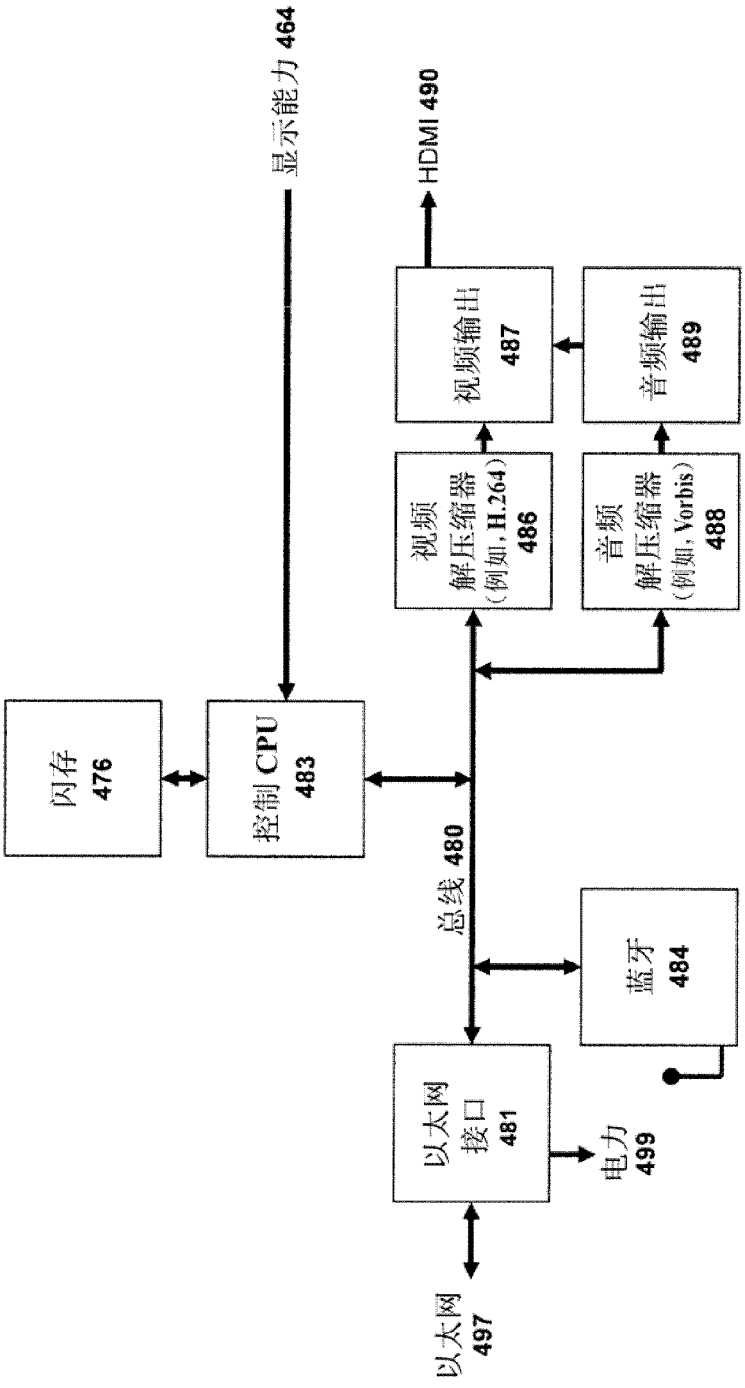

显示能力 464
HDMI 490
视频输出 487
音频输出 489
视频解压缩器（例如，H.264）486
音频解压缩器（例如，Vorbis）488
闪存 476
控制 CPU 483
总线 480
蓝牙 484
以太网接口 481
电力 499
以太网 497

图 4e

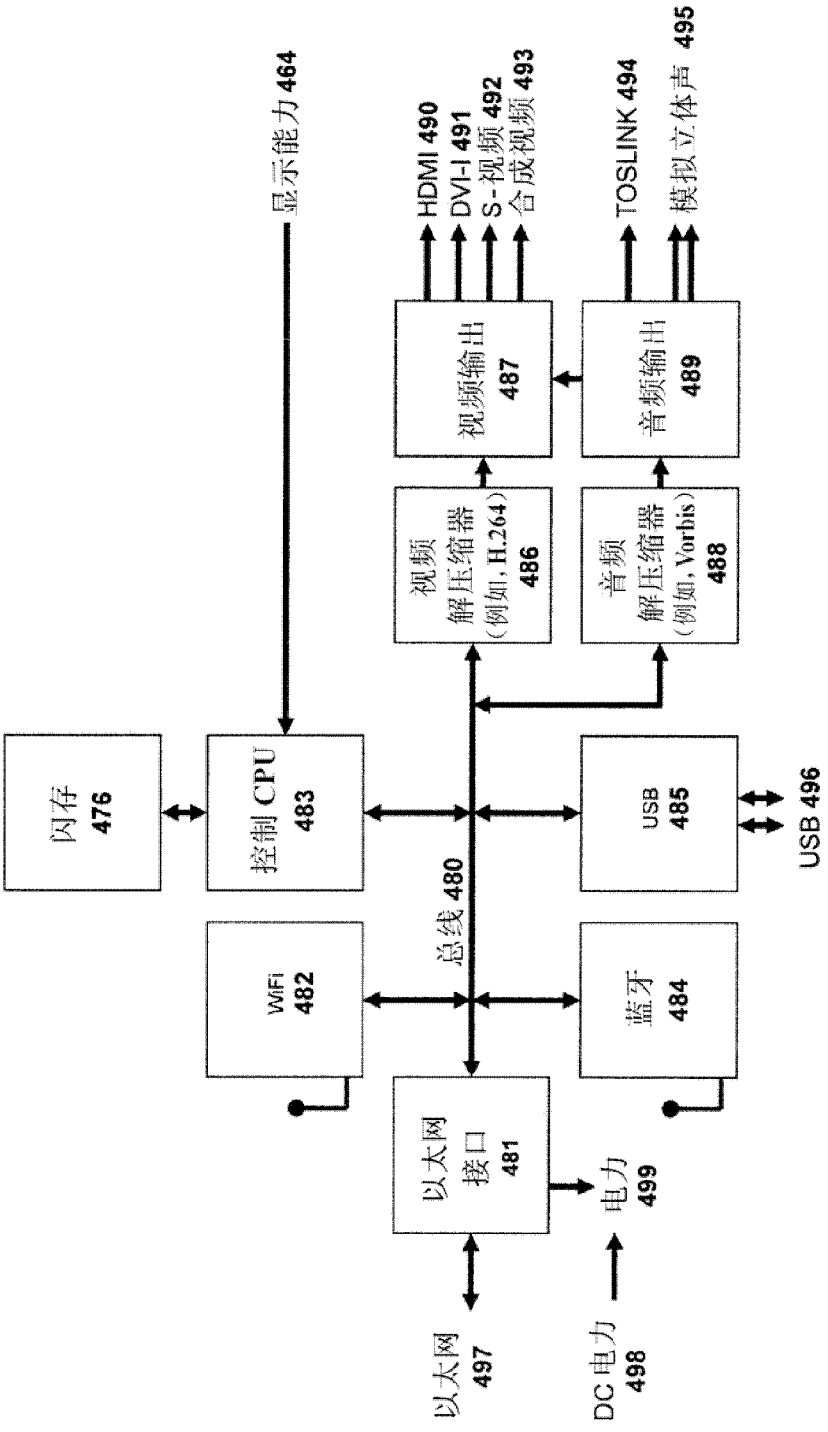

显示能力 464
HDMI 490
DVI-I 491
S-视频 492
合成视频 493
TOSLINK 494
模拟立体声 495
视频输出 487
音频输出 489
视频解压缩器（例如，H.264）486
音频解压缩器（例如，Vorbis）488
闪存 476
控制 CPU 483
USB 485
USB 496
总线 480
WiFi 482
蓝牙 484
以太网接口 481
电力 499
以太网 497
DC 电力 498

图 4f

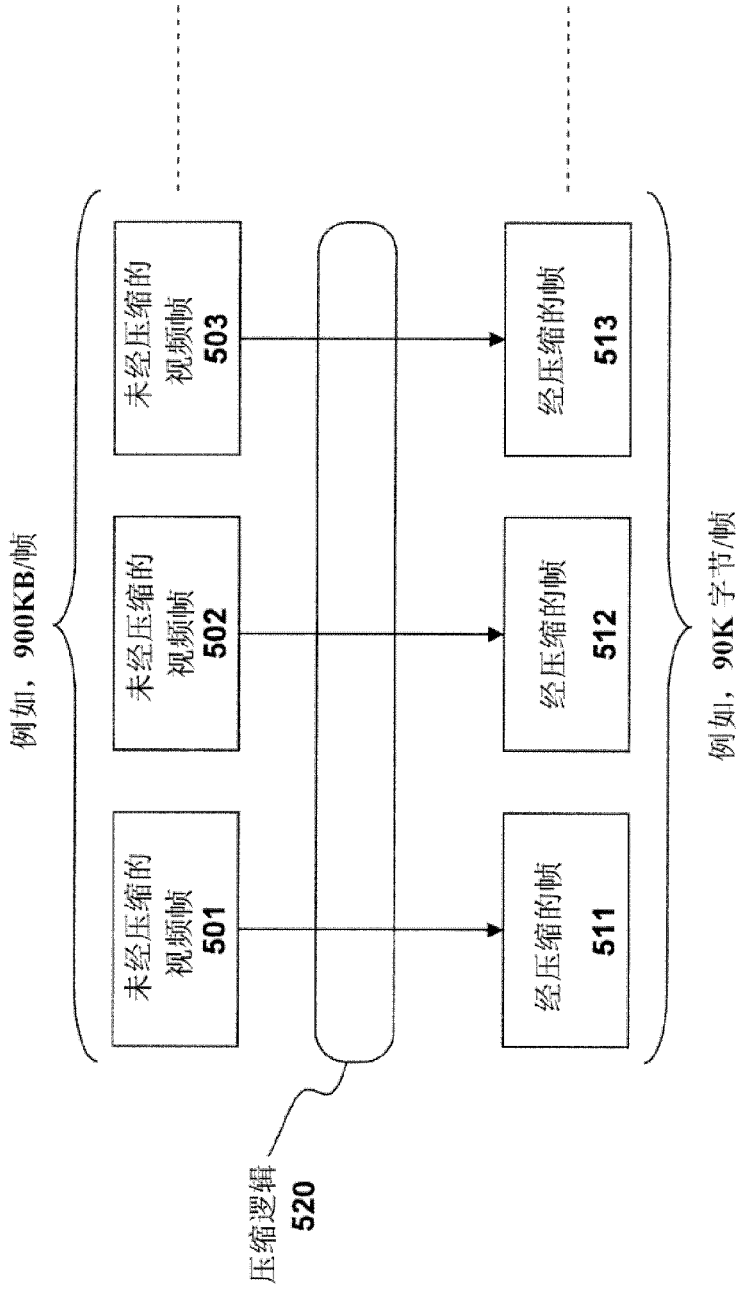

例如，900KB/帧
未经压缩的视频帧 503
未经压缩的视频帧 502
未经压缩的视频帧 501
经压缩的帧 513
经压缩的帧 512
经压缩的帧 511
例如，90K 字节/帧
压缩逻辑 520

图 5（现有技术）

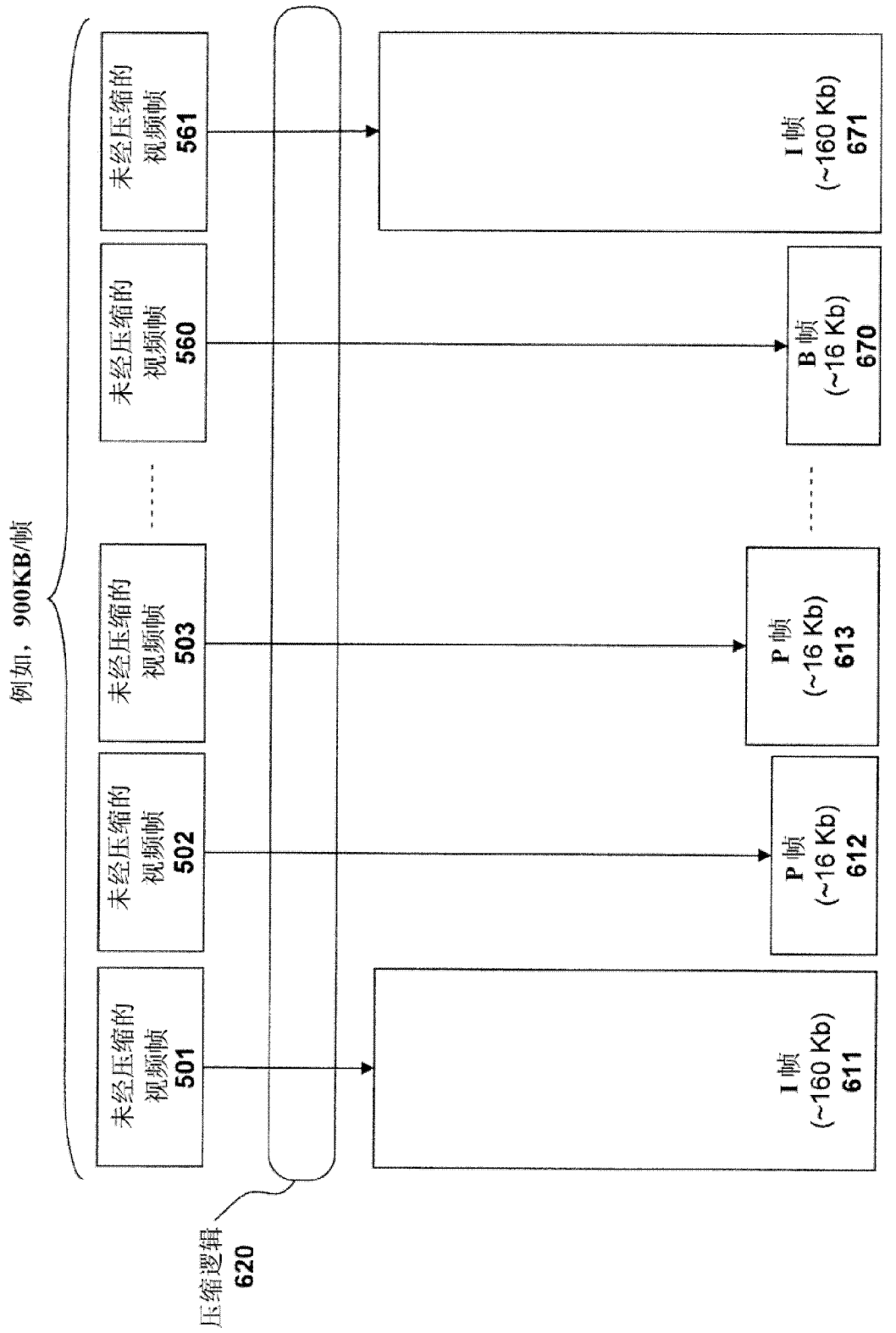

例如，900KB/帧
未经压缩的视频帧 501
未经压缩的视频帧 502
未经压缩的视频帧 503
未经压缩的视频帧 560
未经压缩的视频帧 561
压缩逻辑 620
I 帧 (~160 Kb) 611
P 帧 (~16 Kb) 612
P 帧 (~16 Kb) 613
B 帧 (~16 Kb) 670
I 帧 (~160 Kb) 671

图 6a（现有技术）

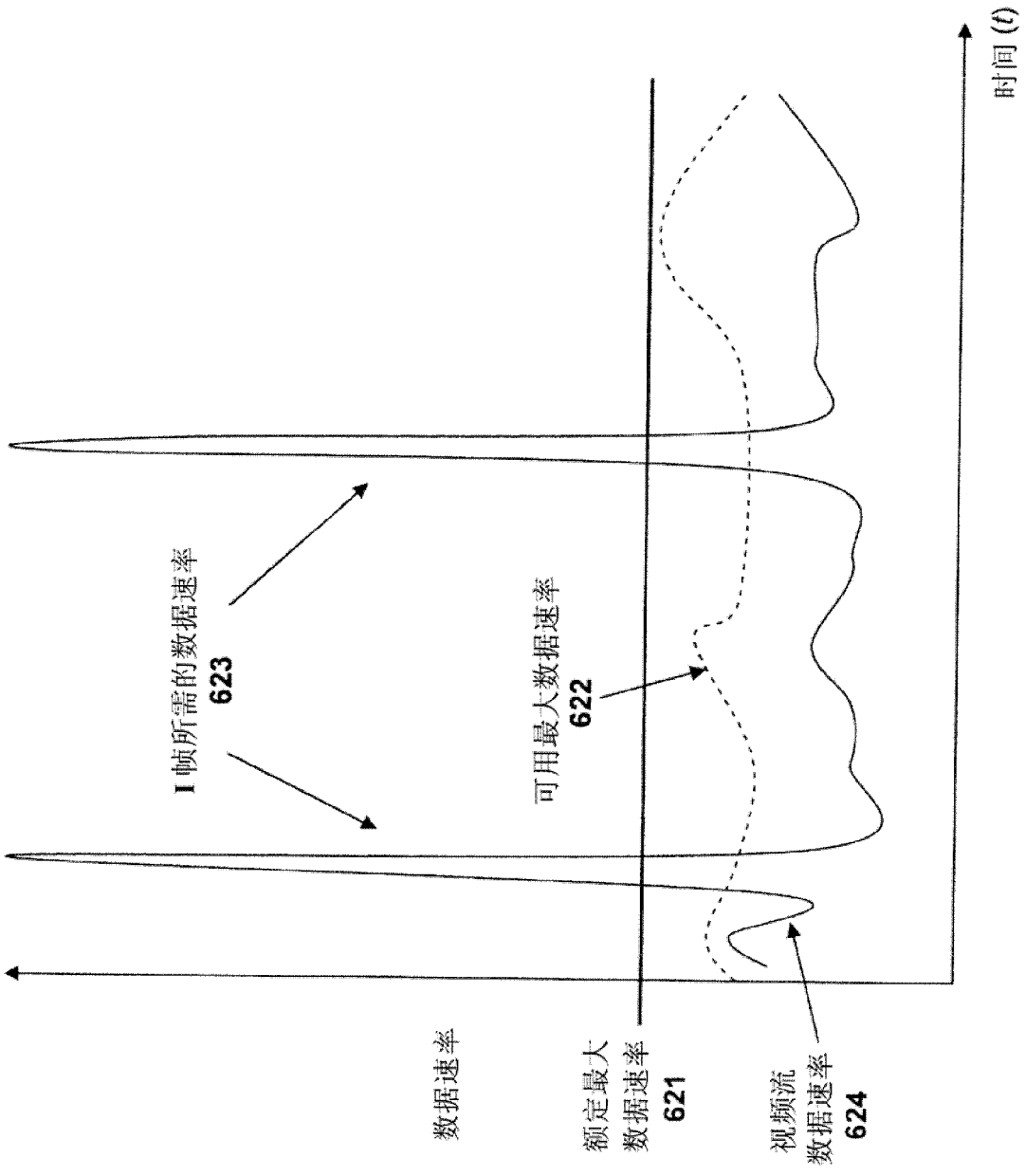

时间 (t)
数据速率
I 帧所需的数据速率
623
可用最大数据速率
622
额定最大
数据速率
621
视频流
数据速率
624

图 6b

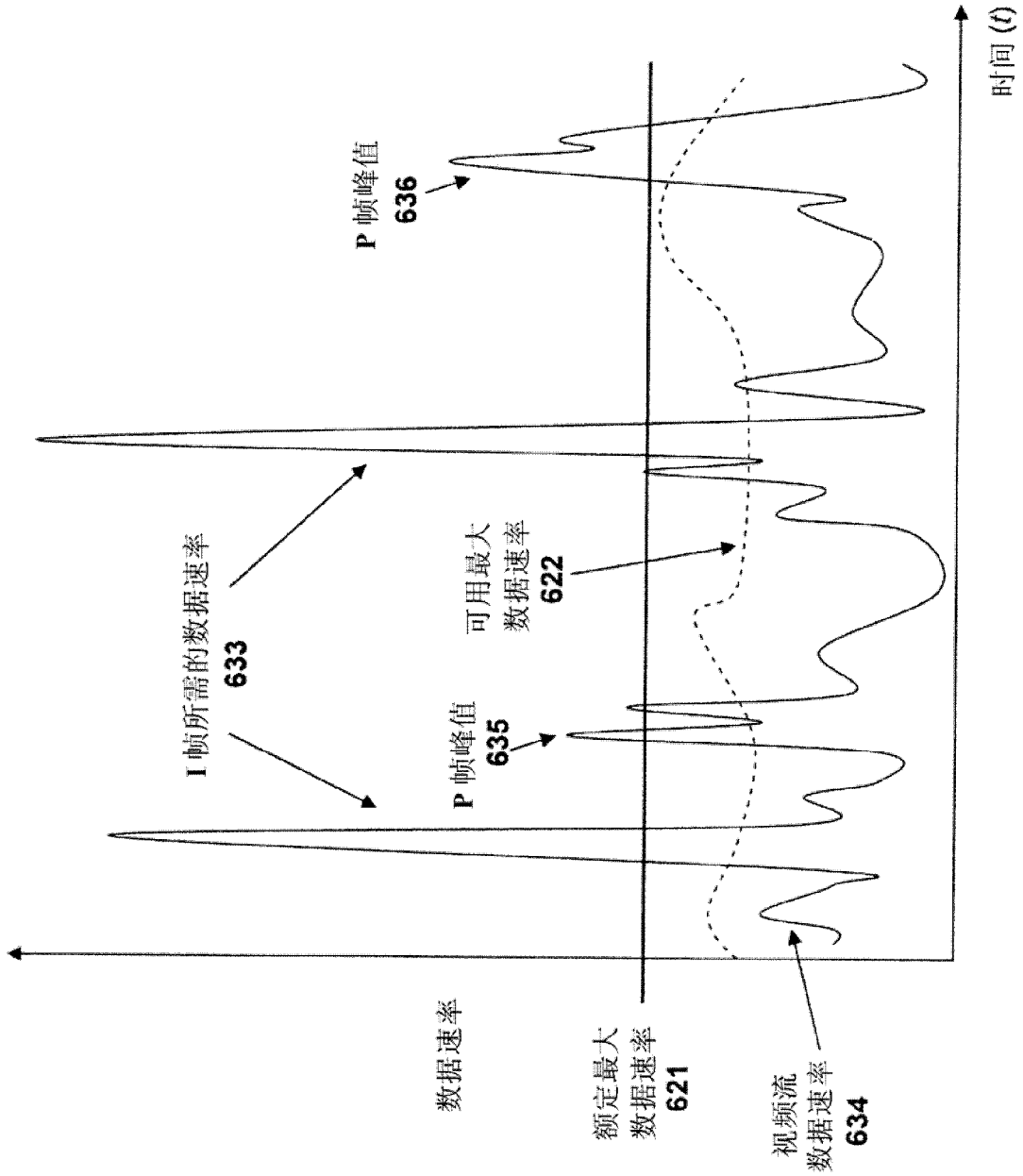

时间 (t)
P 帧峰值
636
I 帧所需的数据速率
633
可用最大
数据速率
622
P 帧峰值
635
数据速率
额定最大
数据速率
621
视频流
数据速率
634

图 6c

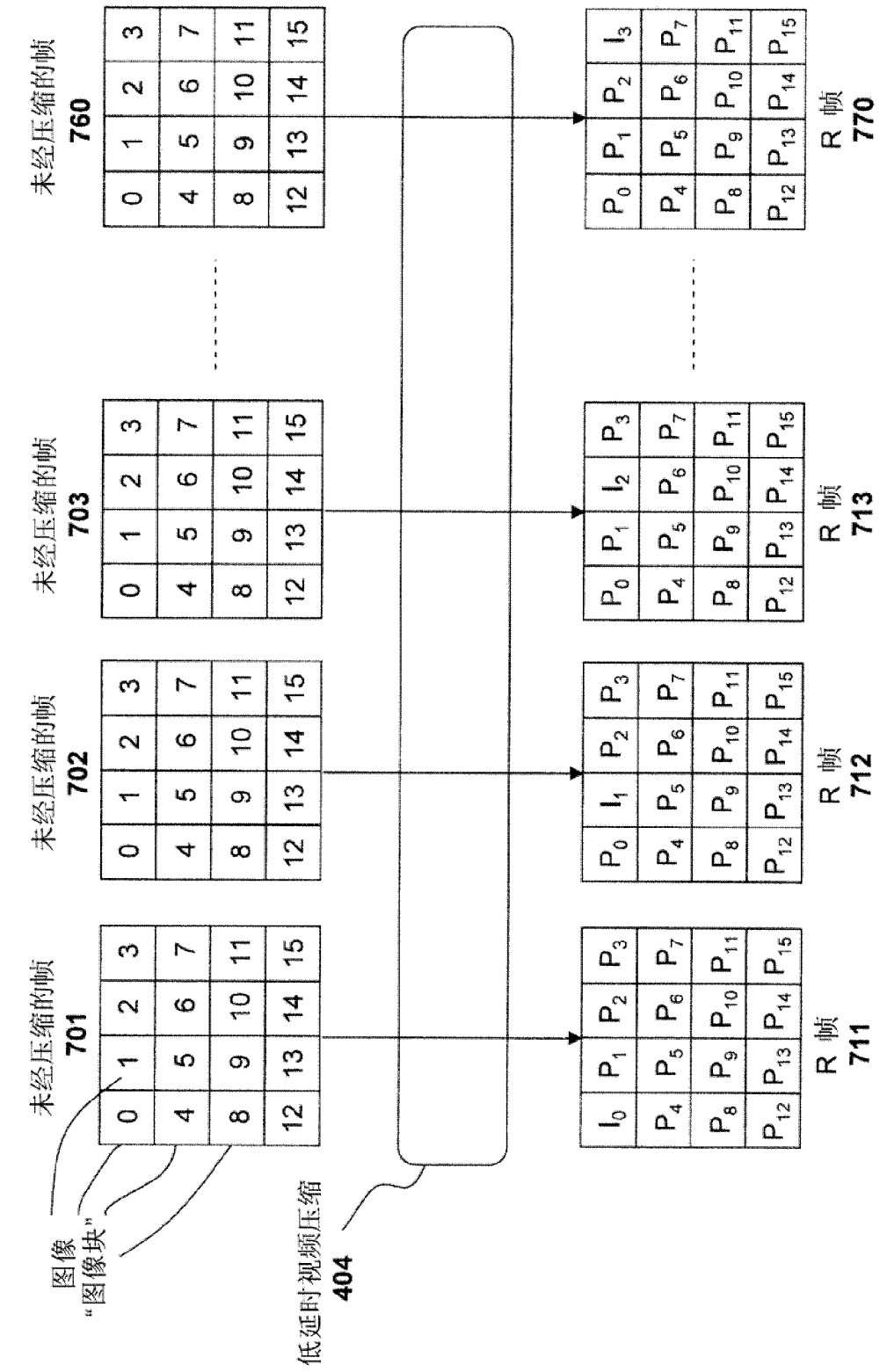

未经压缩的帧
701
未经压缩的帧
702
未经压缩的帧
703
未经压缩的帧
760
图像
“图像块”
低延时视频压缩
404
R 帧
711
R 帧
712
R 帧
713
R 帧
770

图 7a

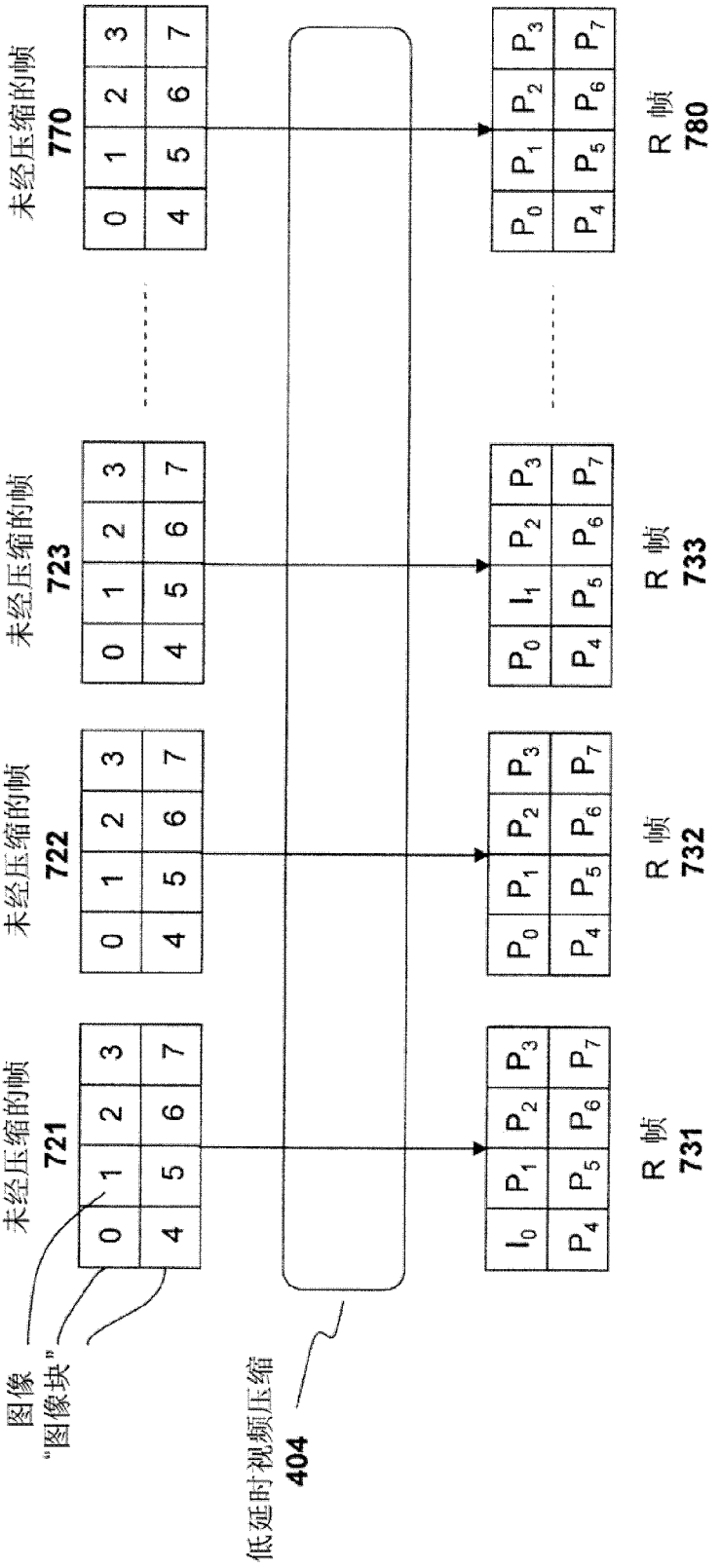

图像
"图像块"
未经压缩的帧
721
未经压缩的帧
722
未经压缩的帧
723
未经压缩的帧
770
低延时视频压缩
404
R 帧
731
R 帧
732
R 帧
733
R 帧
780

图 7b

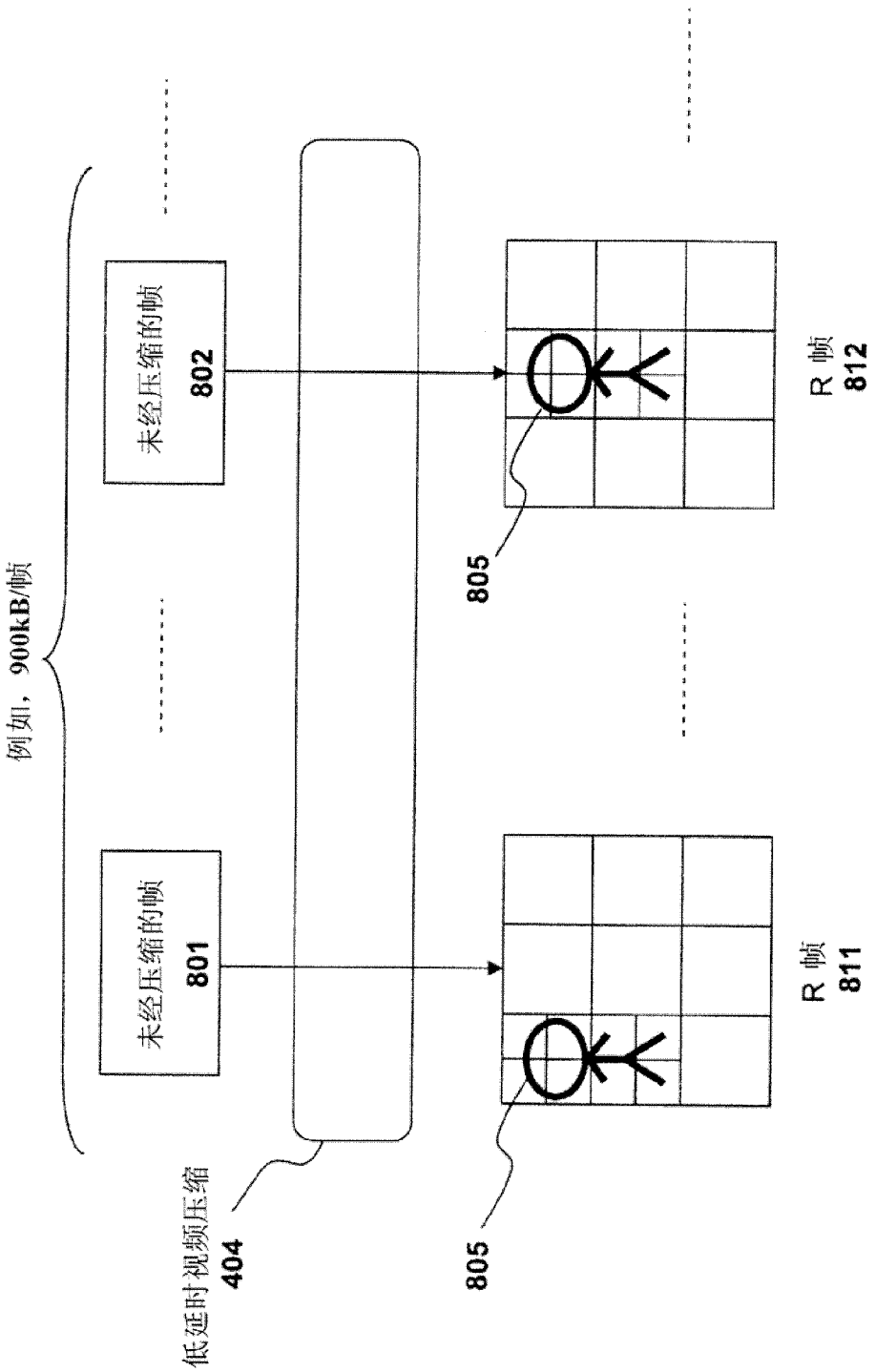

例如，900kB/帧
未经压缩的帧
801
未经压缩的帧
802
低延时视频压缩
404
805
805
R 帧
811
R 帧
812

图 8

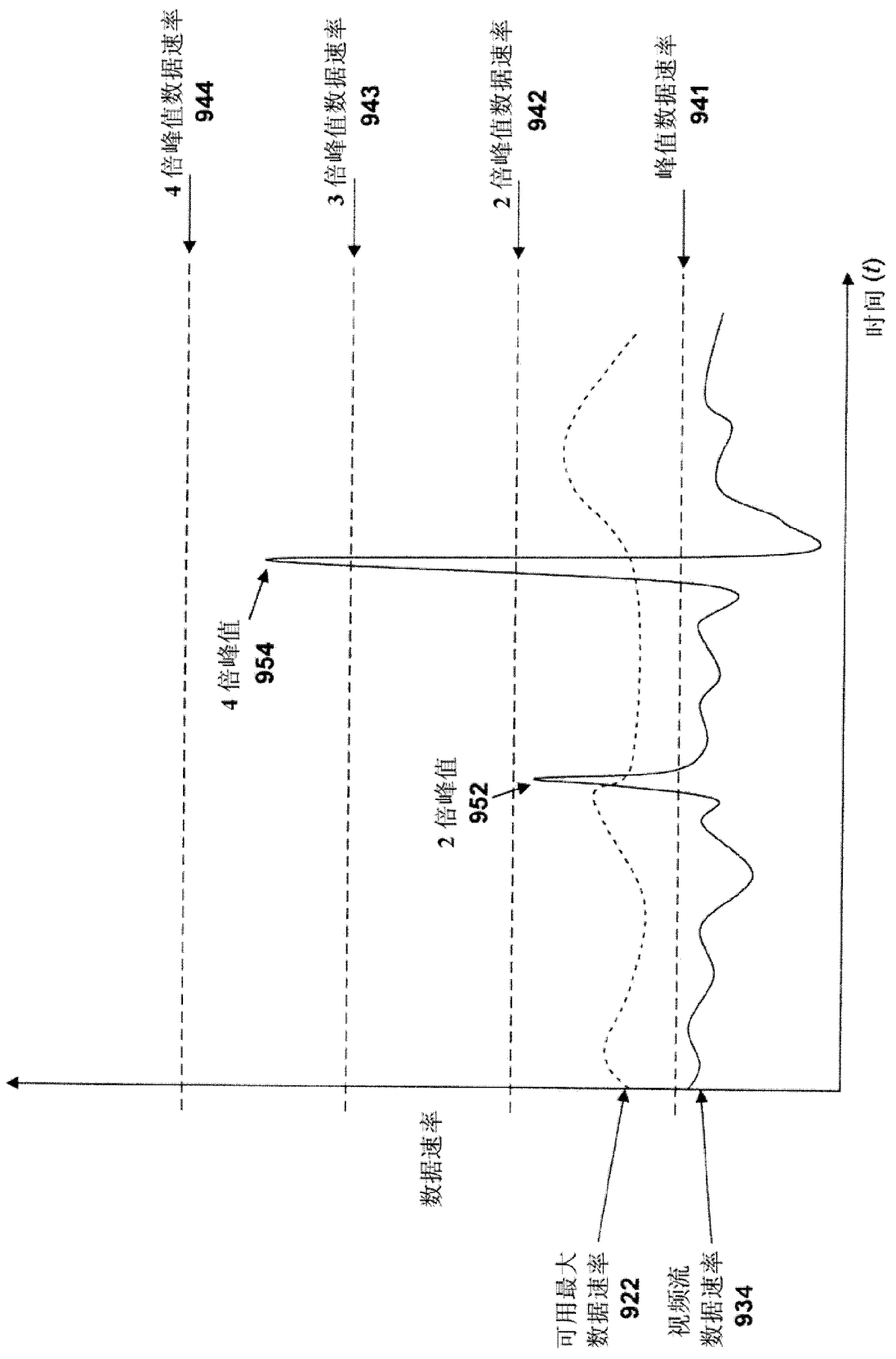

4 倍峰值数据速率
944
3 倍峰值数据速率
943
2 倍峰值数据速率
942
峰值数据速率
941
4 倍峰值
954
2 倍峰值
952
数据速率
时间 (t)
可用最大
数据速率
922
视频流
数据速率
934

图 9a

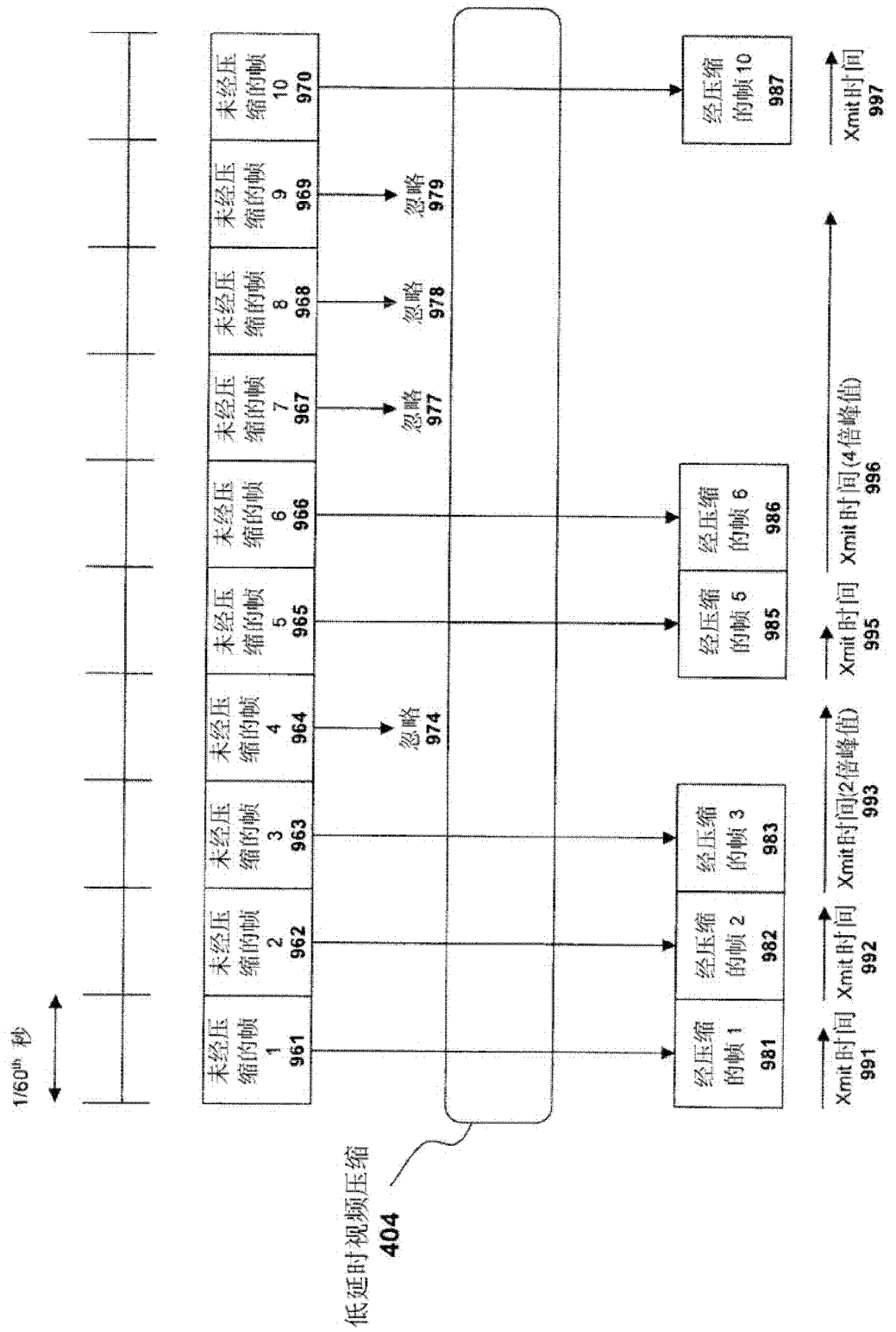

1/60th 秒
未经压缩的帧 1 961
未经压缩的帧 2 962
未经压缩的帧 3 963
未经压缩的帧 4 964
未经压缩的帧 5 965
未经压缩的帧 6 966
未经压缩的帧 7 967
未经压缩的帧 8 968
未经压缩的帧 9 969
未经压缩的帧 10 970
忽略 974
忽略 977
忽略 978
忽略 979
低延时视频压缩 404
经压缩的帧 1 981
经压缩的帧 2 982
经压缩的帧 3 983
经压缩的帧 5 985
经压缩的帧 6 986
经压缩的帧 10 987
Xmit 时间 991
Xmit 时间 992
Xmit 时间(2倍峰值) 993
Xmit 时间 995
Xmit 时间(4倍峰值) 996
Xmit 时间 997

图 9b

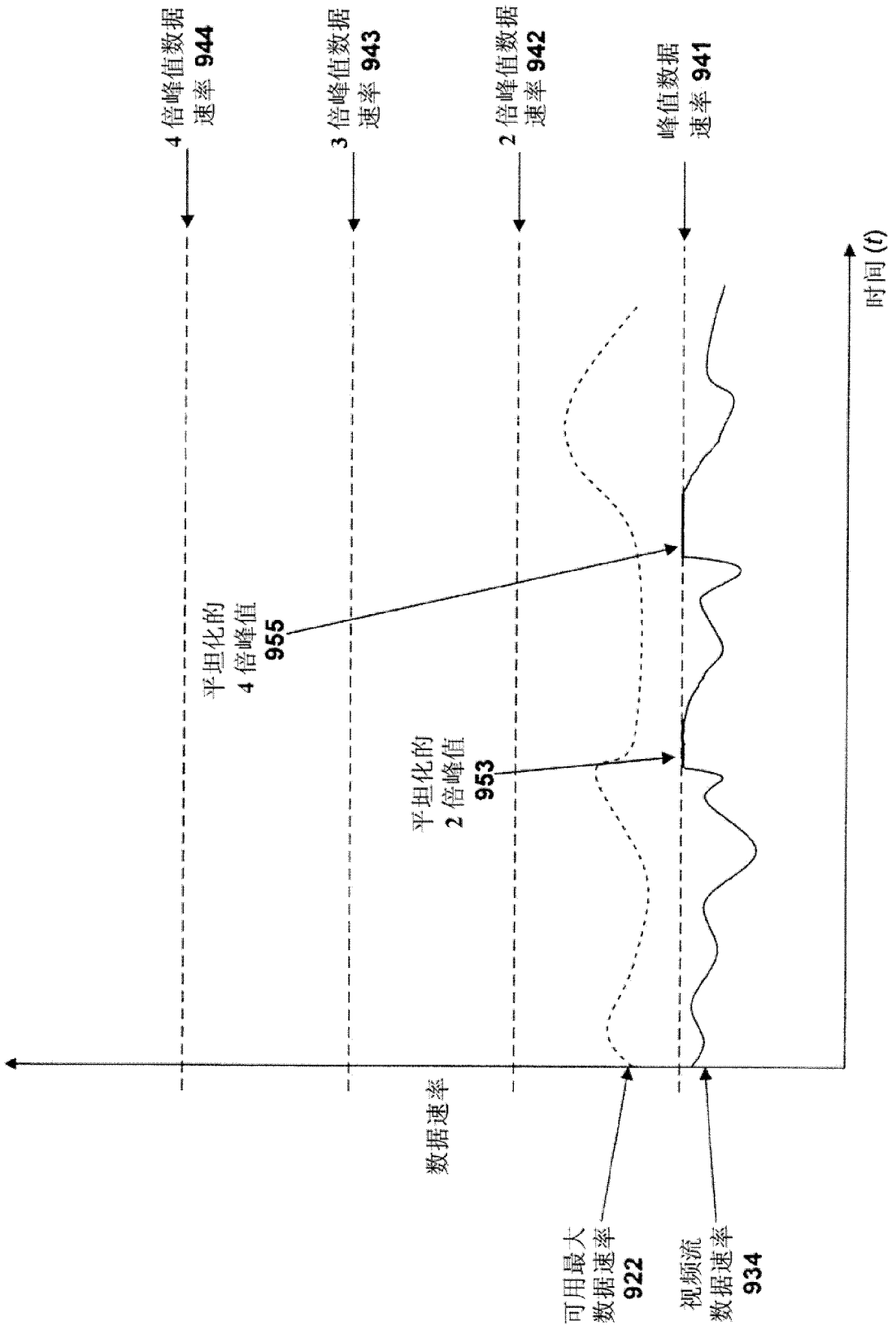

4 倍峰值数据速率 944
3 倍峰值数据速率 943
2 倍峰值数据速率 942
峰值数据速率 941
平坦化的 4 倍峰值 955
平坦化的 2 倍峰值 953
数据速率
时间 (t)
可用最大数据速率 922
视频流数据速率 934

图 9c

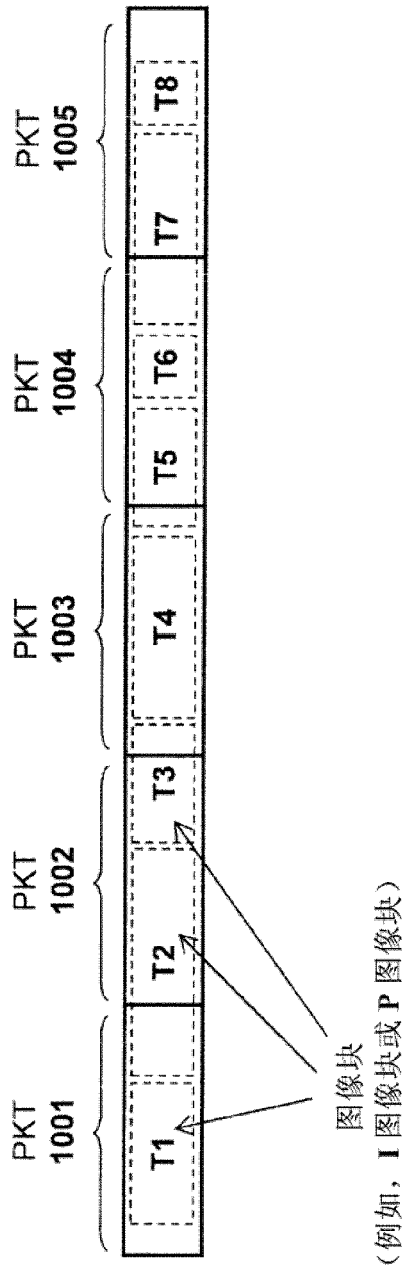

PKT 1001
PKT 1002
PKT 1003
PKT 1004
PKT 1005
T1
T2
T3
T4
T5
T6
T7
T8
图像块
（例如，I 图像块或 P 图像块）

图 10a

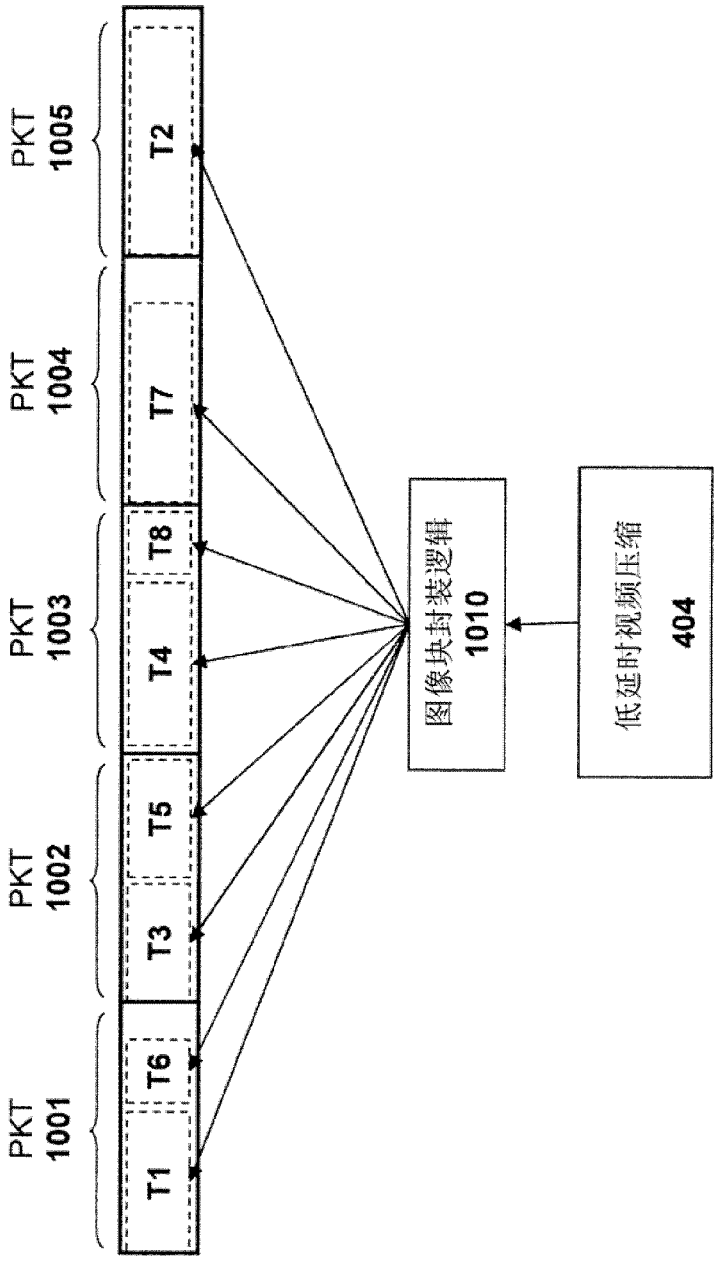

PKT 1001
PKT 1002
PKT 1003
PKT 1004
PKT 1005
T1
T6
T3
T5
T4
T8
T7
T2
图像块封装逻辑 1010
低延时视频压缩 404

图 10b

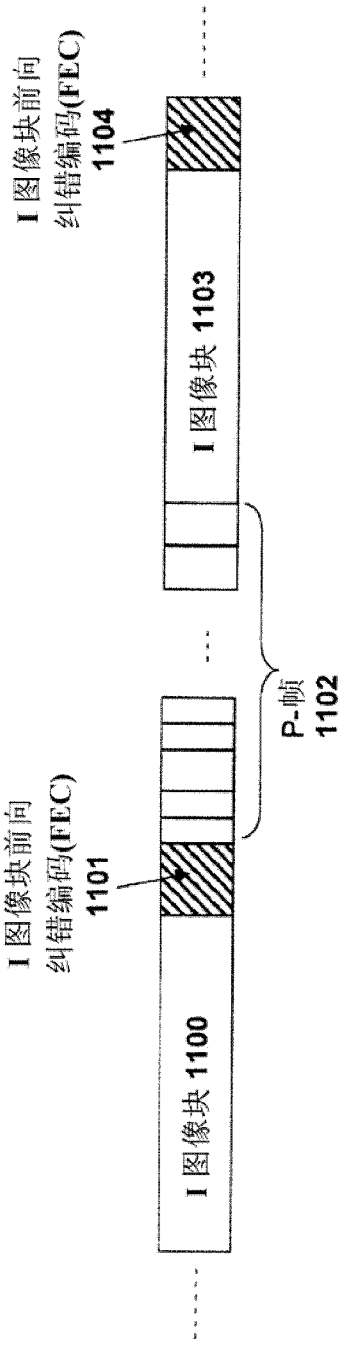


图 11a

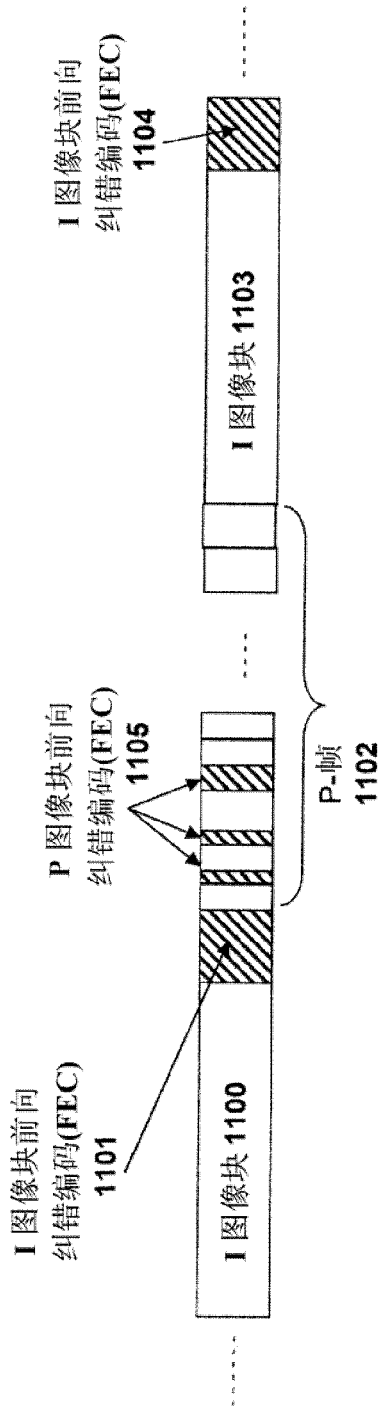


图 11b

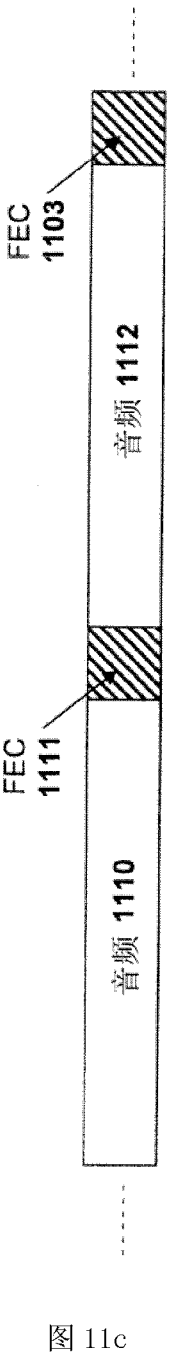


图 11c

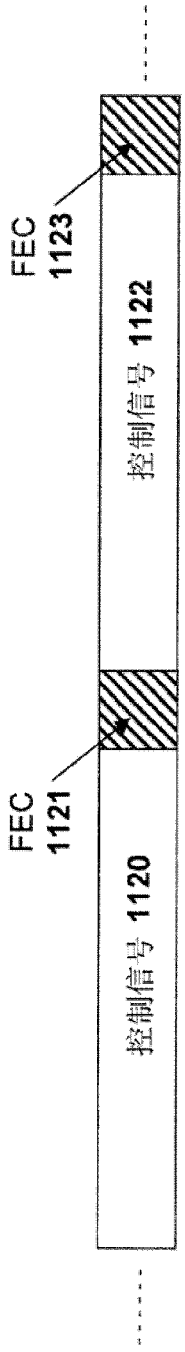


图 11d

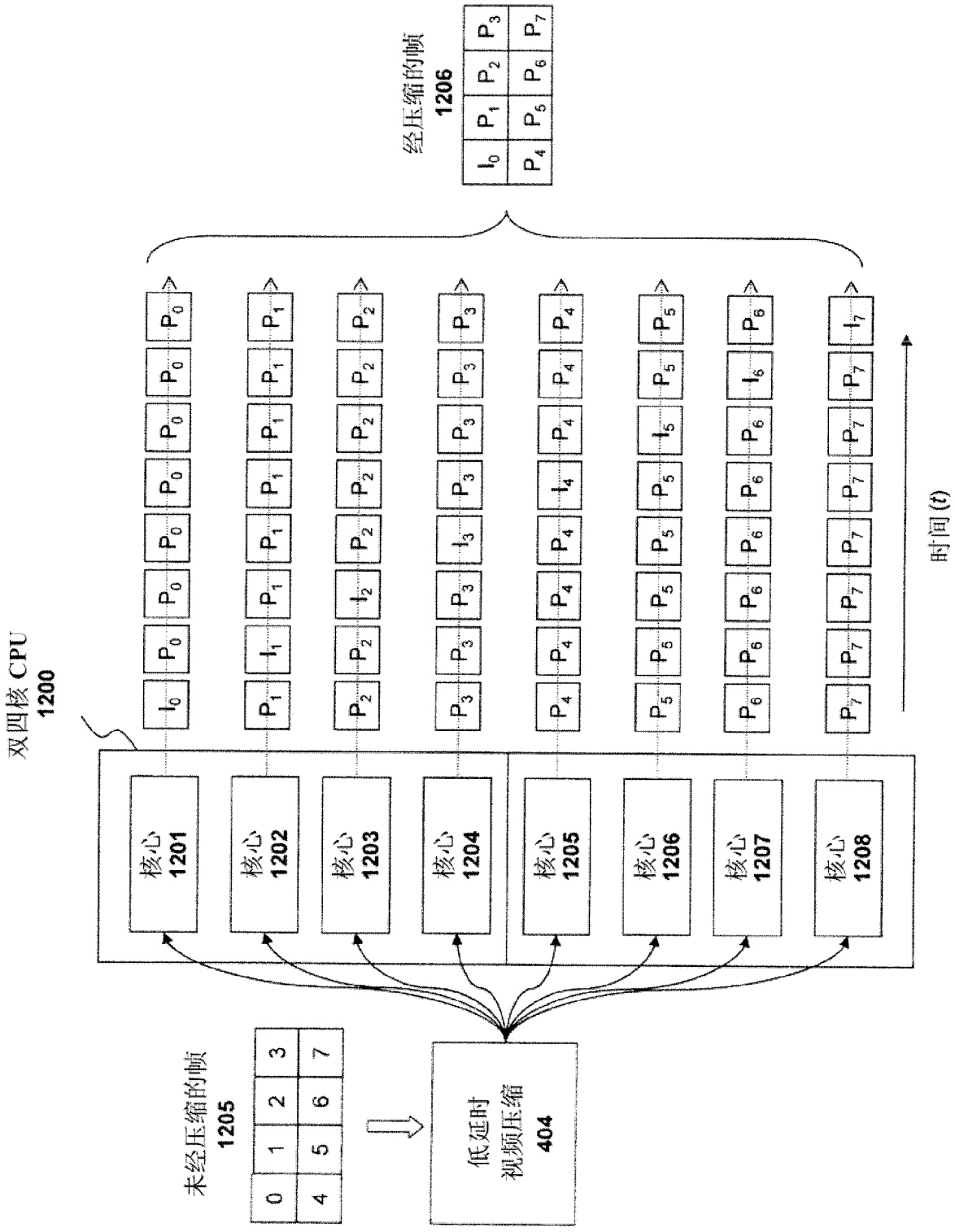

双四核 CPU
1200
核心
1201
核心
1202
核心
1203
核心
1204
核心
1205
核心
1206
核心
1207
核心
1208
经压缩的帧
1206
未经压缩的帧
1205
低延时
视频压缩
404
时间 (t)

图 12

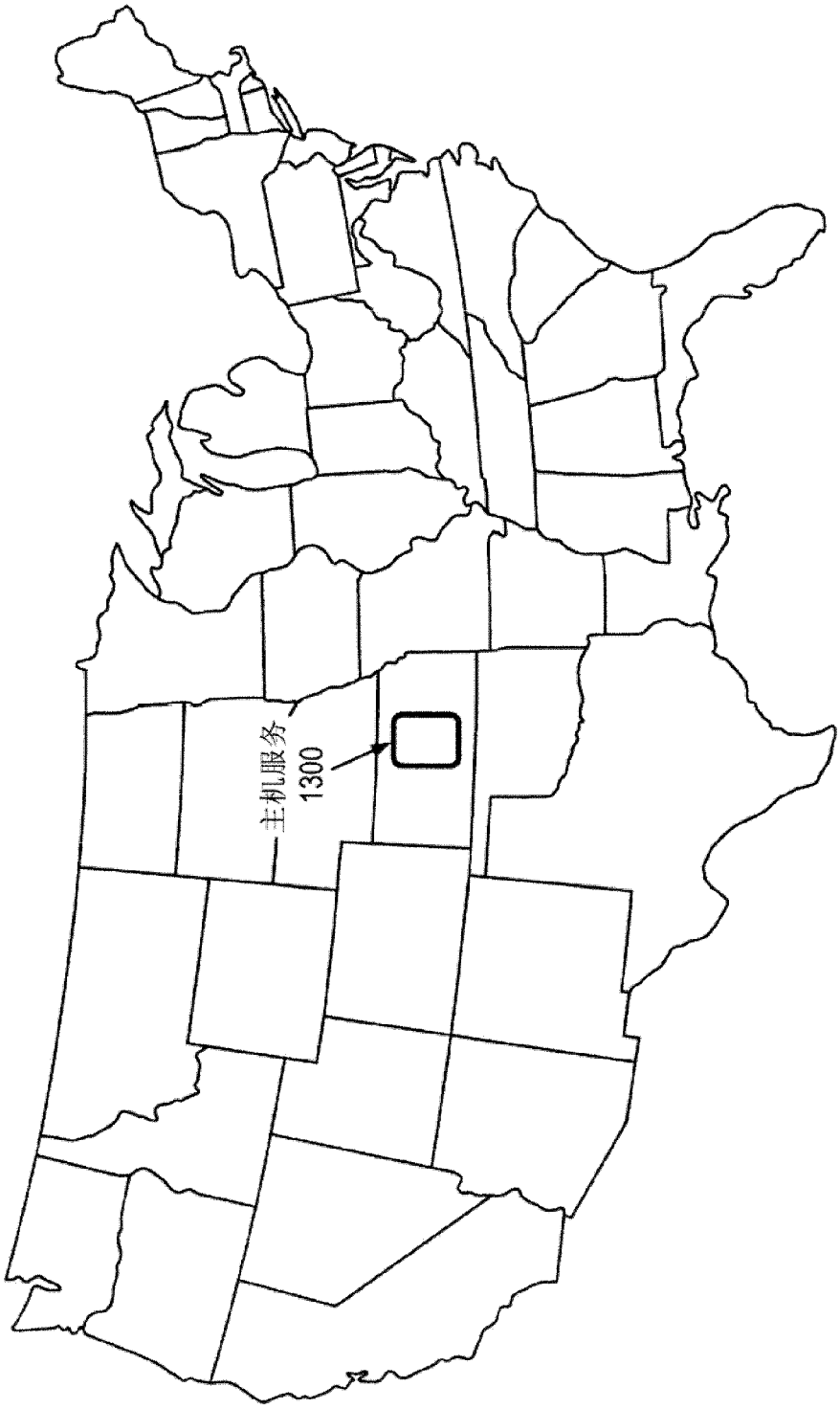


图 13a

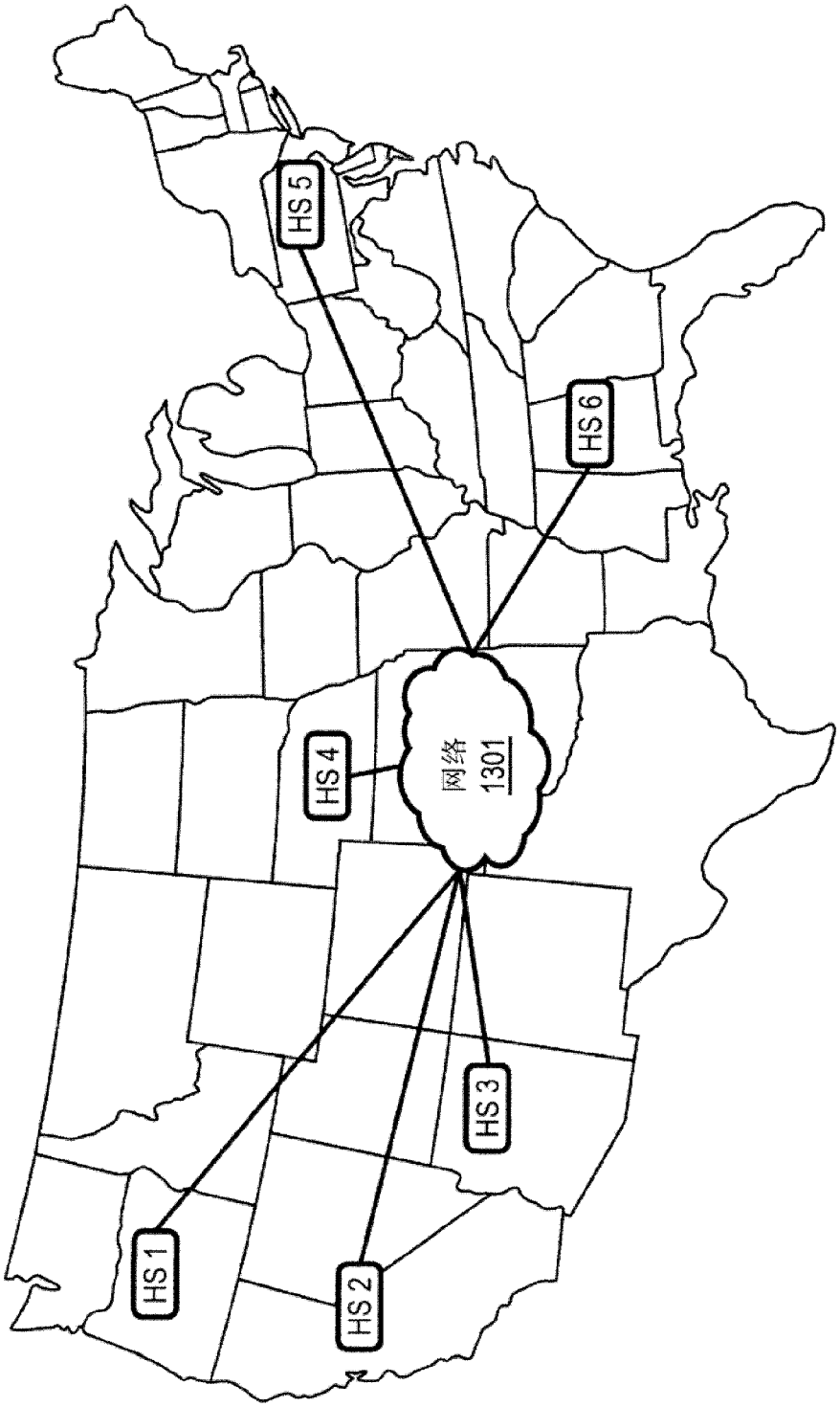


图 13b

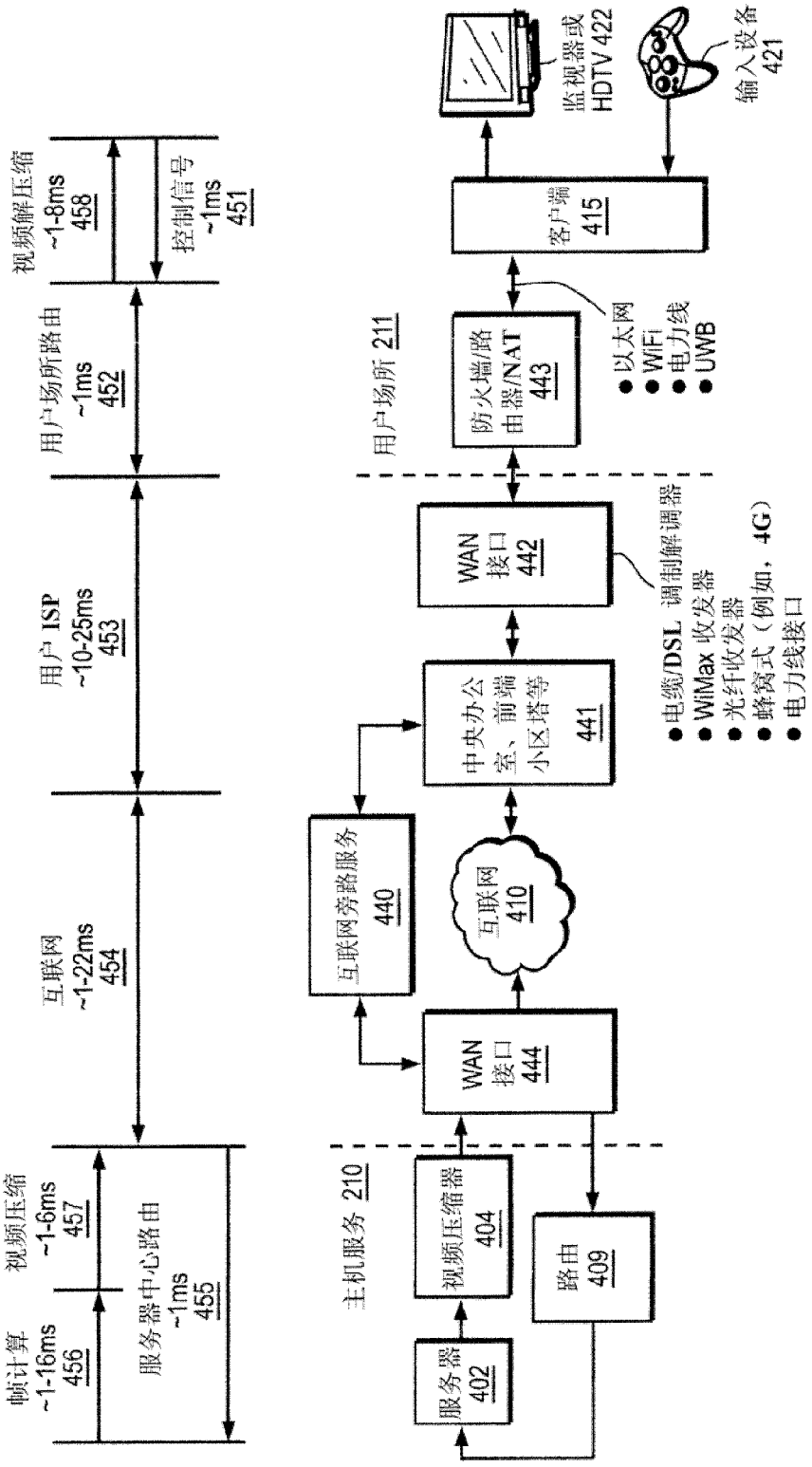

帧计算
~1-16ms
456
视频压缩
~1-6ms
457
服务器中心路由
~1ms
455
互联网
~1-22ms
454
用户 ISP
~10-25ms
453
用户场所路由
~1ms
452
视频解压缩
~1-8ms
458
控制信号
~1ms
451
主机服务 210
服务器
402
视频压缩器
404
路由
409
WAN
接口
444
互联网旁路服务
440
互联网
410
中央办公
室、前端
小区塔等
441
WAN
接口
442
用户场所 211
防火墙/路
由器/NAT
443
客户端
415
监视器或
HDTV 422
输入设备
421
以太网
WiFi
电力线
UWB
电缆/DSL 调制解调器
WiMax 收发器
光纤收发器
蜂窝式（例如，4G）
电力线接口

图 14

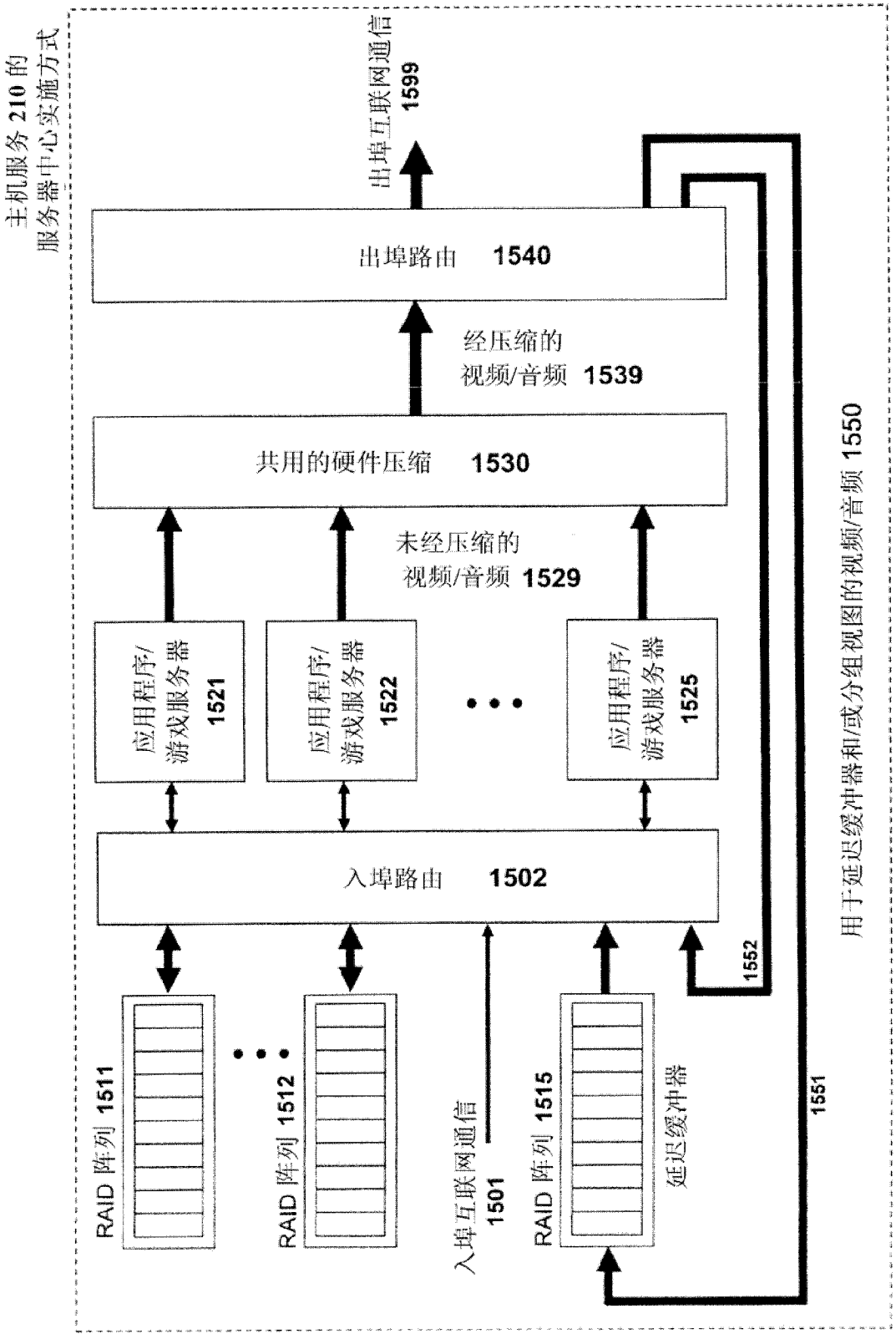

主机服务 210 的
服务器中心实施方式
出埠互联网通信
1599
出埠路由 1540
经压缩的
视频/音频 1539
共用的硬件压缩 1530
未经压缩的
视频/音频 1529
应用程序/
游戏服务器
1521
应用程序/
游戏服务器
1522
应用程序/
游戏服务器
1525
入埠路由 1502
RAID 阵列 1511
RAID 阵列 1512
入埠互联网通信
1501
RAID 阵列 1515
延迟缓冲器
1552
1551
用于延迟缓冲器和/或分组视图的视频/音频 1550

图 15

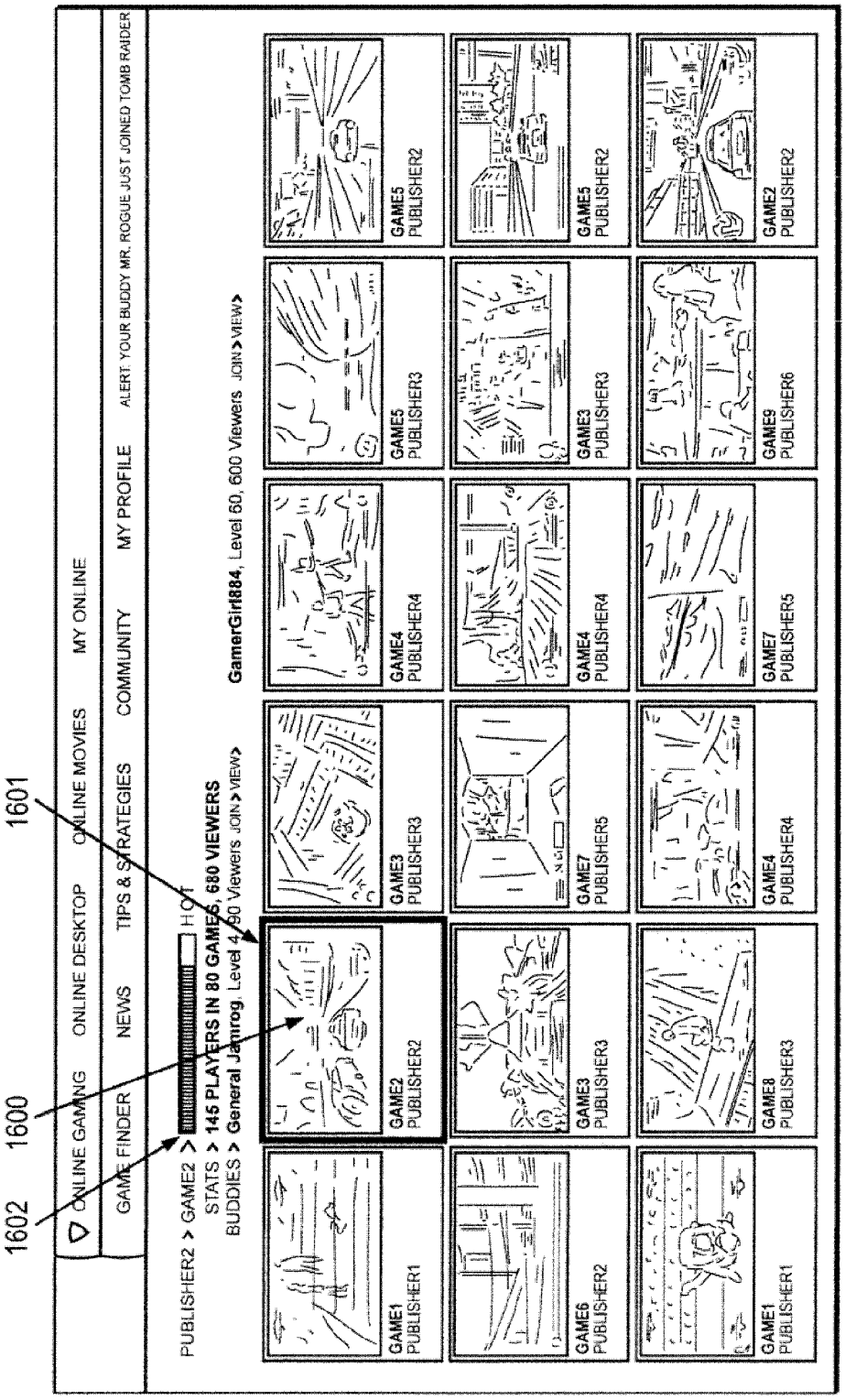

1602
1600
1601
ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
ALERT: YOUR BUDDY MR. ROGUE JUST JOINED TOMB RAIDER
PUBLISHER2 > GAME2 >
HOT
STATS > 145 PLAYERS IN 80 GAMES, 680 VIEWERS
BUDDIES > General Jamrog, Level 4, 80 Viewers JOIN > VIEW>
GamerGirl884, Level 60, 600 Viewers JOIN > VIEW>
GAME1 PUBLISHER1
GAME2 PUBLISHER2
GAME3 PUBLISHER3
GAME4 PUBLISHER4
GAME5 PUBLISHER2
GAME6 PUBLISHER2
GAME3 PUBLISHER3
GAME7 PUBLISHER5
GAME4 PUBLISHER4
GAME5 PUBLISHER3
GAME1 PUBLISHER1
GAME8 PUBLISHER3
GAME4 PUBLISHER4
GAME7 PUBLISHER5
GAME9 PUBLISHER6
GAME3 PUBLISHER3
GAME5 PUBLISHER2
GAME2 PUBLISHER2

图 16

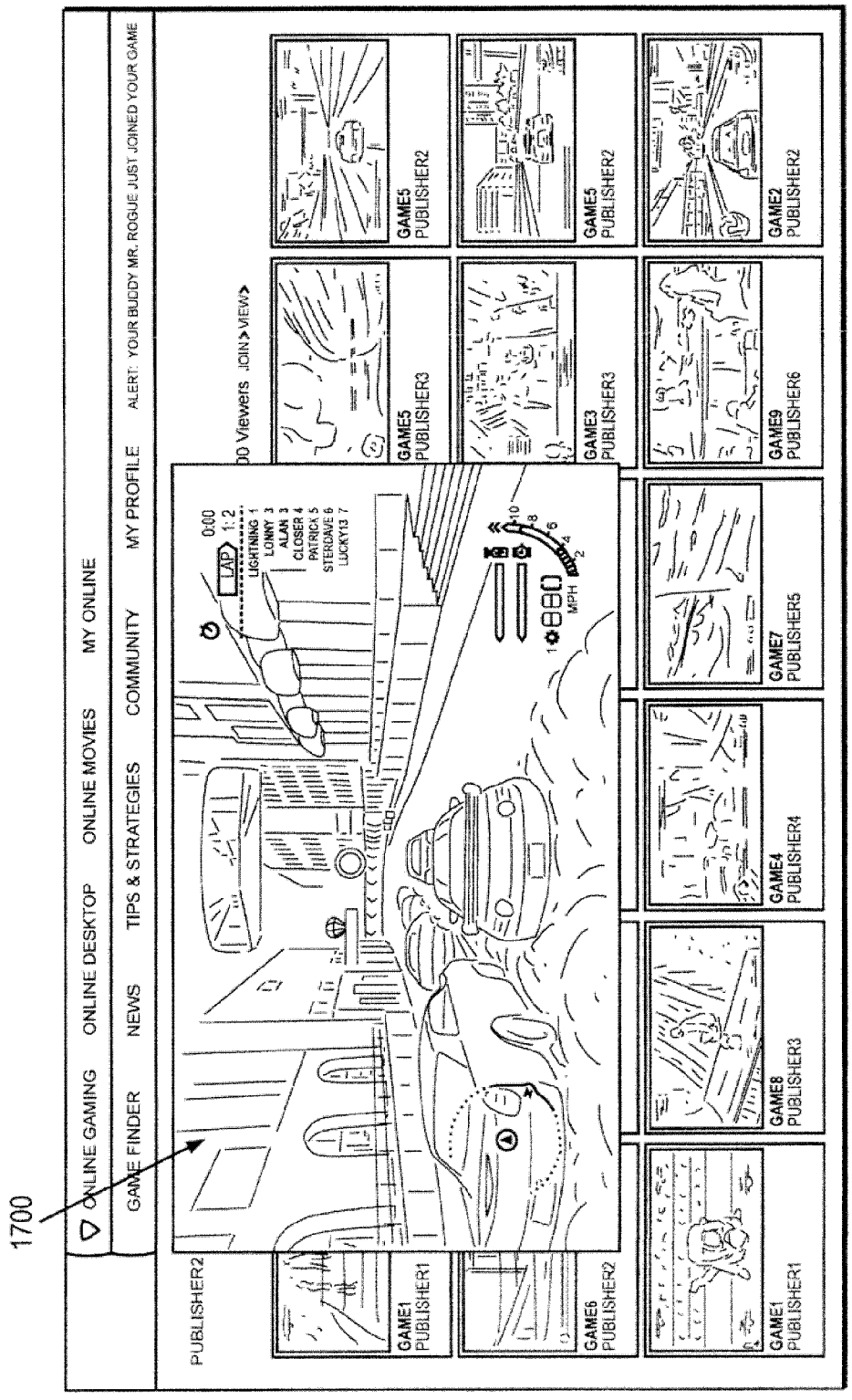

1700
ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
ALERT: YOUR BUDDY MR. ROGUE JUST JOINED YOUR GAME
PUBLISHER2
00 Viewers JOIN>VIEW>
LAP
LIGHTNING 1
LONNY 3
ALAN 3
CLOSER 4
PATRICK 5
STERDAVE 6
LUCKY13 7
MPH
GAME1 PUBLISHER1
GAME6 PUBLISHER2
GAME8 PUBLISHER3
GAME4 PUBLISHER4
GAME7 PUBLISHER5
GAME9 PUBLISHER6
GAME5 PUBLISHER3
GAME3 PUBLISHER3
GAME5 PUBLISHER2
GAME2 PUBLISHER2

图 17

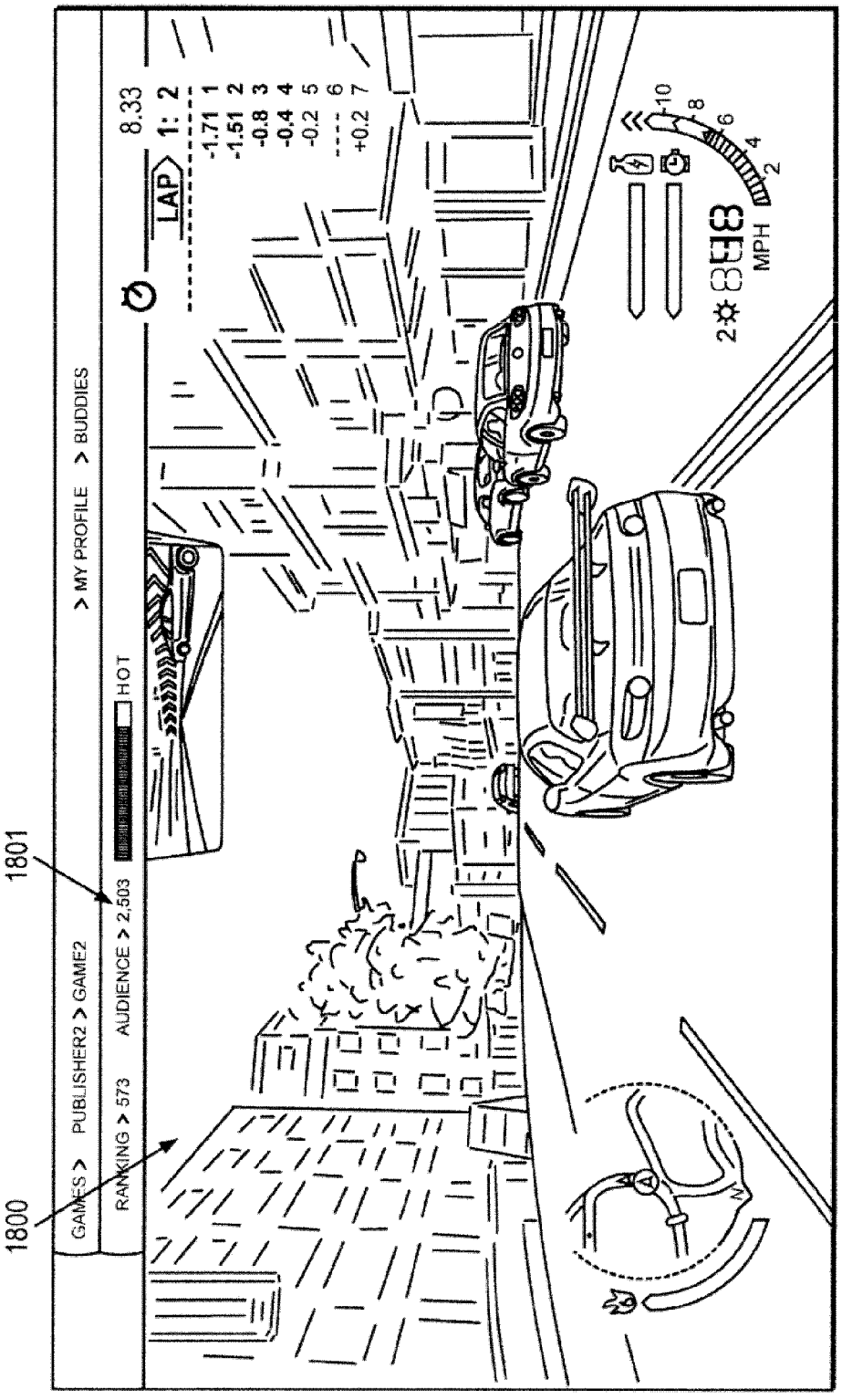

1801
1800
GAMES > PUBLISHER2 > GAME2
RANKING > 573
AUDIENCE > 2,503
HOT
> MY PROFILE > BUDDIES
LAP 1: 2
8.33
-1.71 1
-1.51 2
-0.8 3
-0.4 4
-0.2 5
---- 6
+0.2 7
MPH

图 18

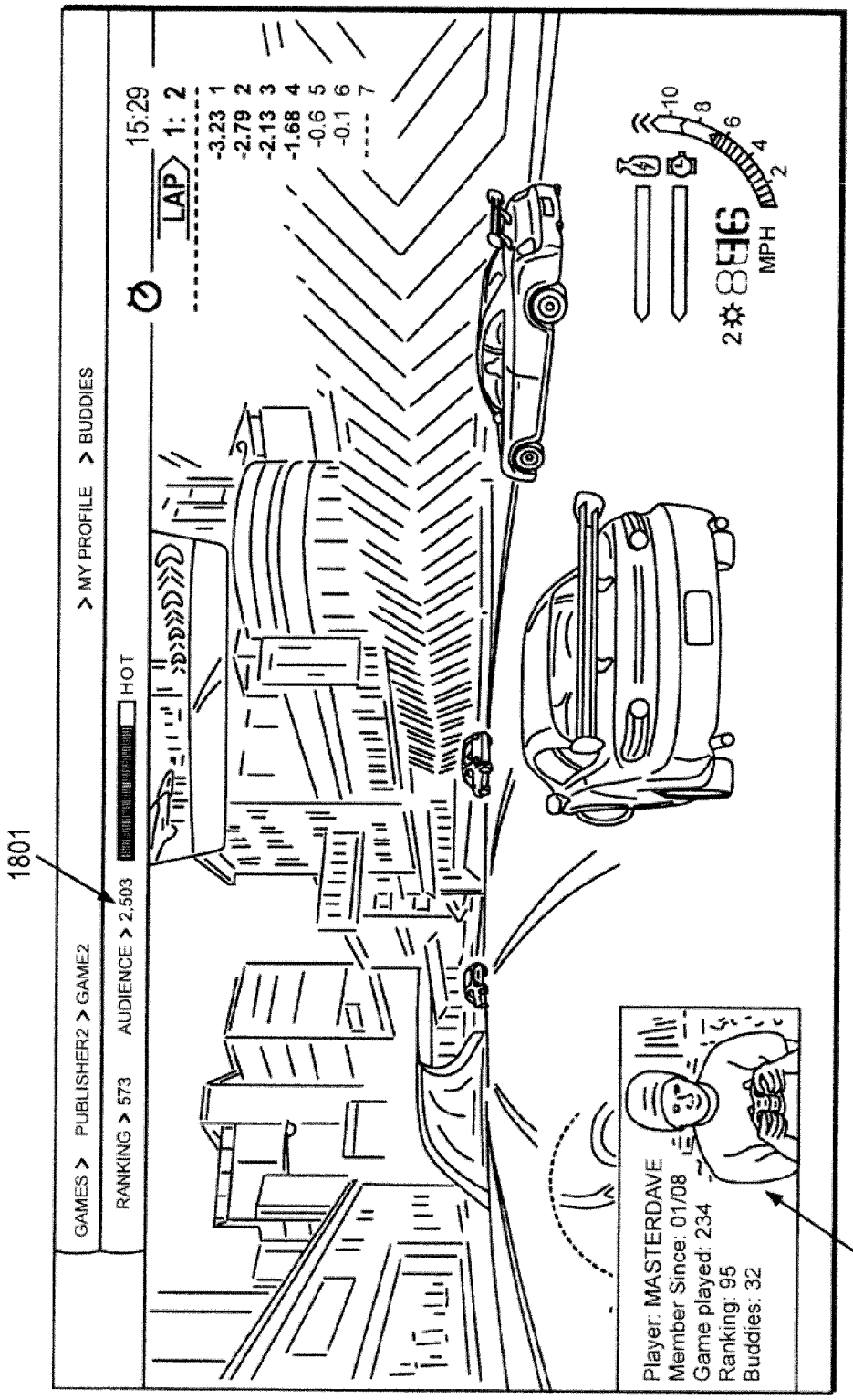

GAMES > PUBLISHER2 > GAME2
> MY PROFILE > BUDDIES
RANKING > 573 AUDIENCE > 2,503 HOT
LAP 1: 2
15:29
-3.23 1
-2.79 2
-2.13 3
-1.68 4
-0.6 5
-0.1 6
7
MPH
Player: MASTERDAVE
Member Since: 01/08
Game played: 234
Ranking: 95
Buddies: 32
1801
1900

图 19

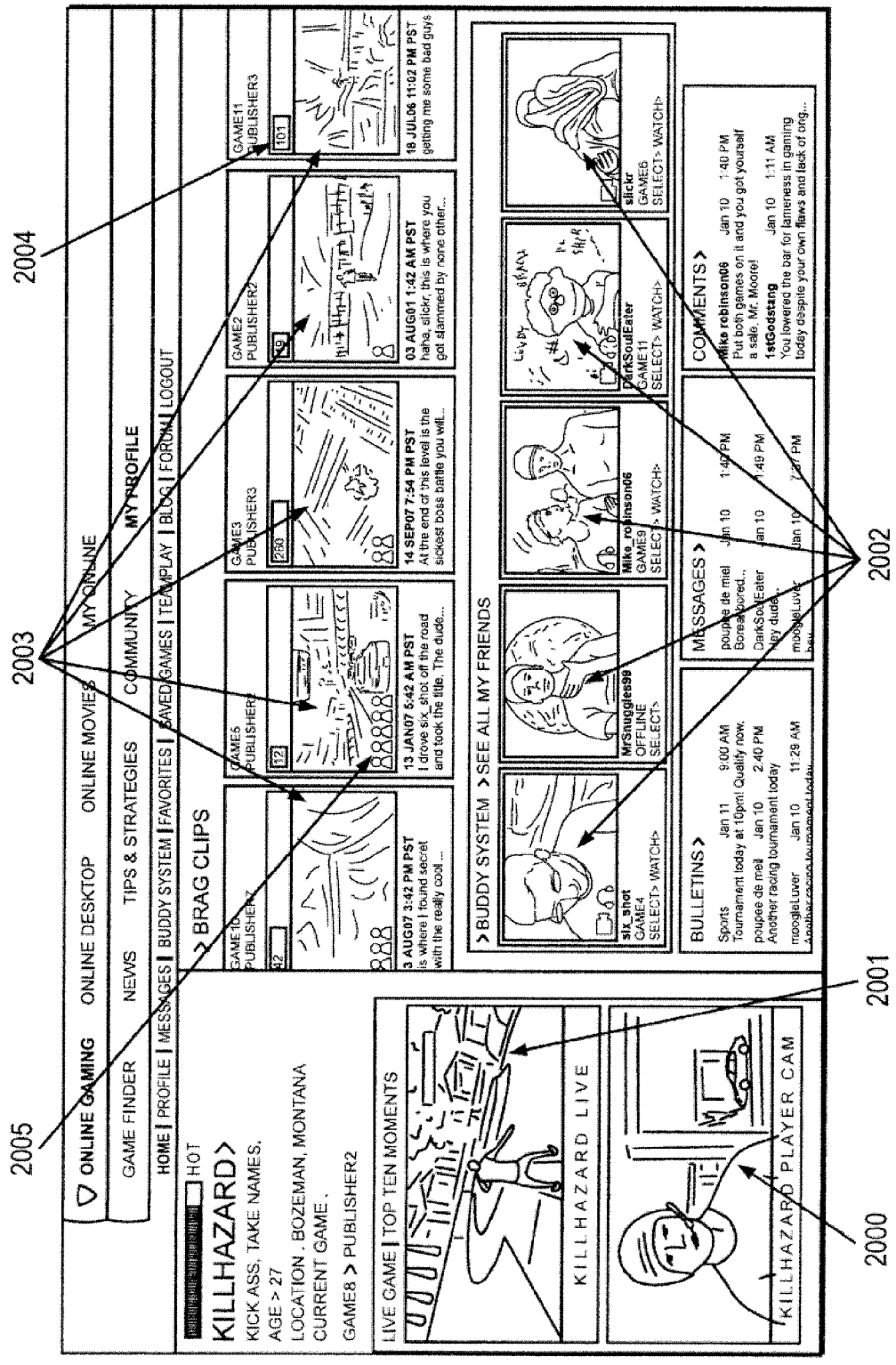

ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
HOME | PROFILE | MESSAGES | BUDDY SYSTEM | FAVORITES | SAVED GAMES | TEAMPLAY | BLOG | FORUM | LOGOUT
HOT
KILLHAZARD>
KICK ASS, TAKE NAMES.
AGE > 27
LOCATION . BOZEMAN, MONTANA
CURRENT GAME .
GAME8 > PUBLISHER2
> BRAG CLIPS
> BUDDY SYSTEM > SEE ALL MY FRIENDS
LIVE GAME | TOP TEN MOMENTS
KILLHAZARD LIVE
KILLHAZARD PLAYER CAM
BULLETINS >
MESSAGES >
COMMENTS >
2000
2001
2002
2003
2004
2005

图 20

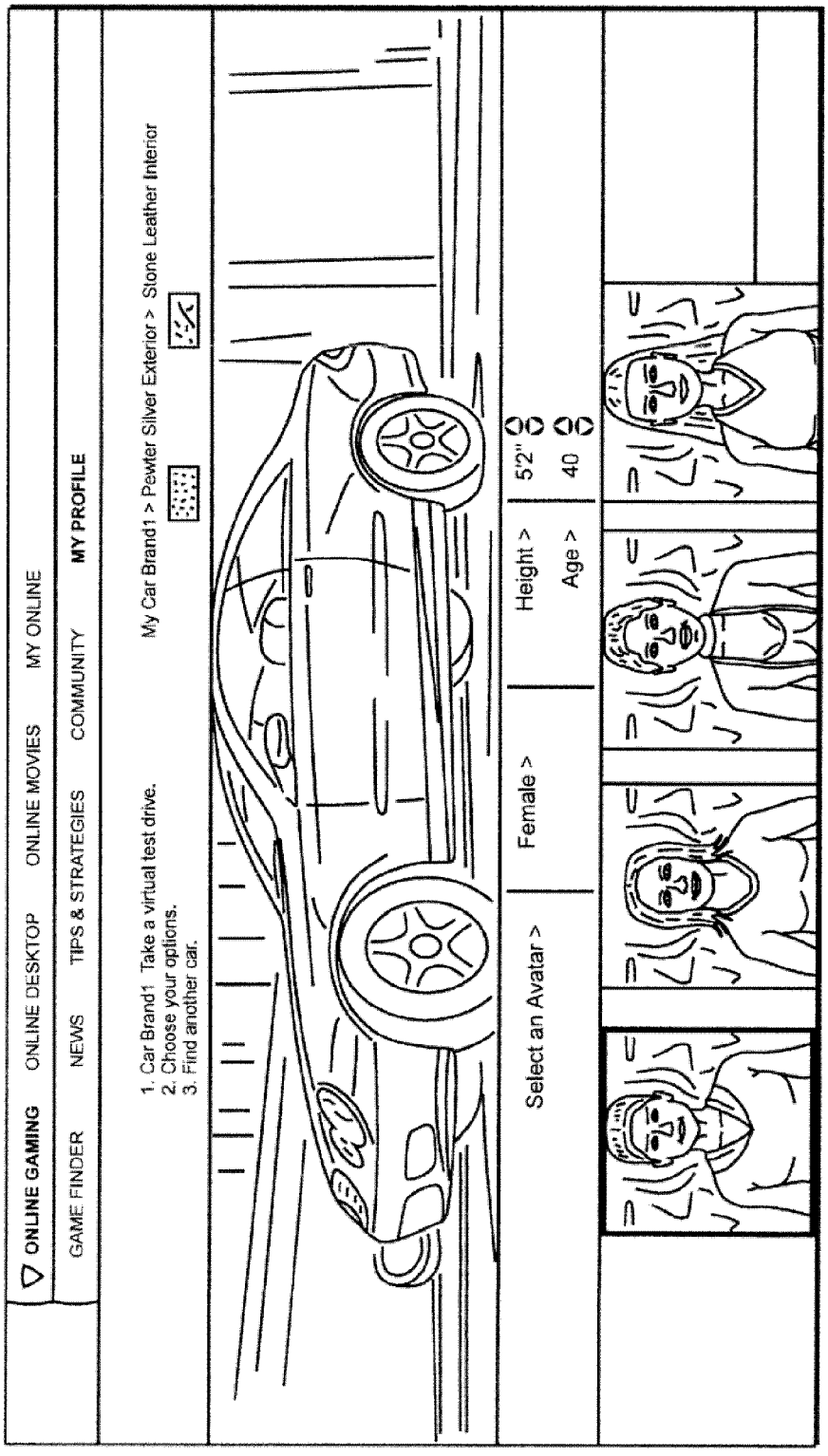

ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GAME FINDER
NEWS
TIPS & STRATEGIES
COMMUNITY
MY PROFILE
1. Car Brand1 Take a virtual test drive.
2. Choose your options.
3. Find another car.
My Car Brand1 > Pewter Silver Exterior > Stone Leather Interior
Select an Avatar >
Female >
Height >
5'2"
Age >
40

图 21

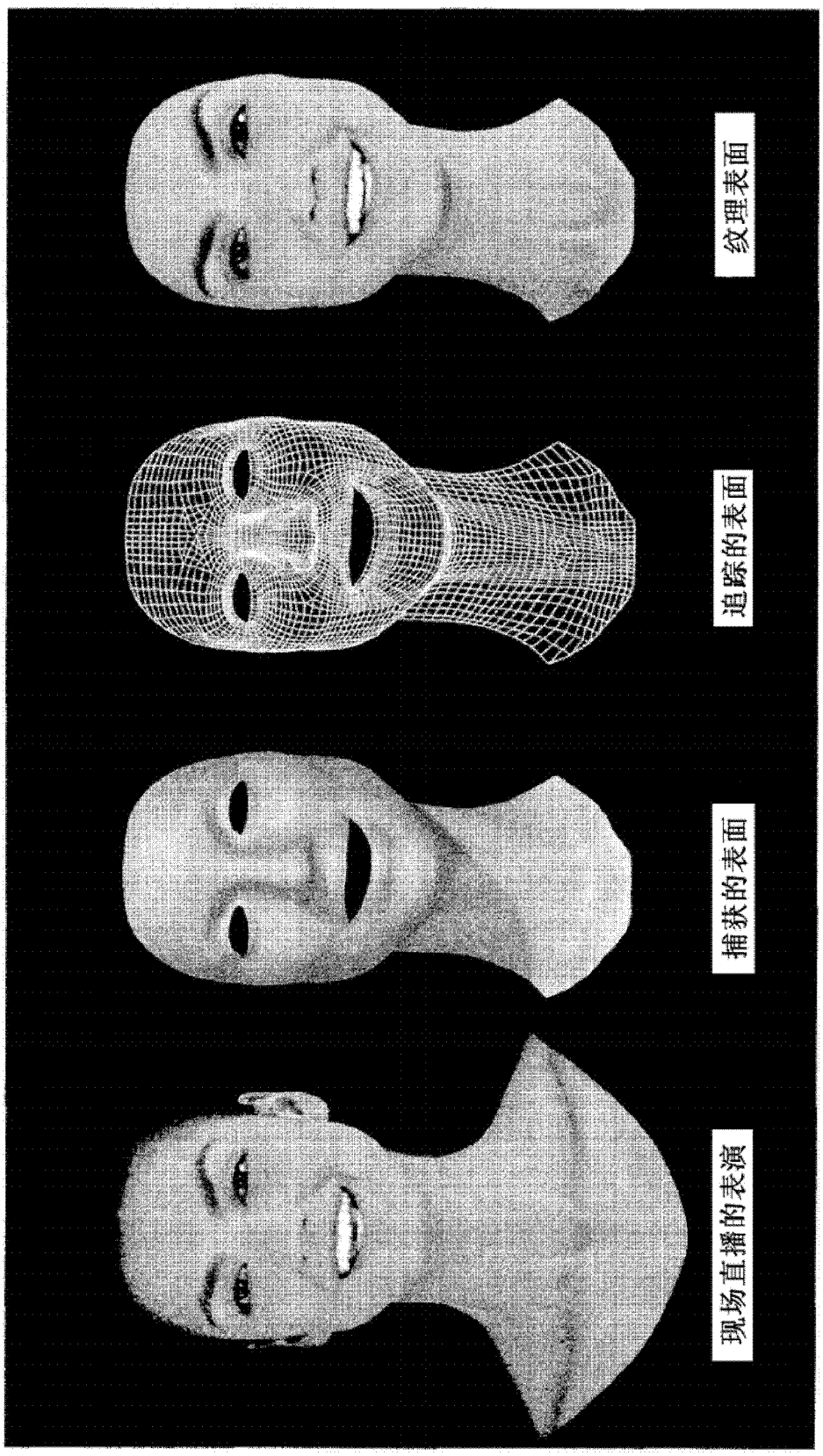


图 22

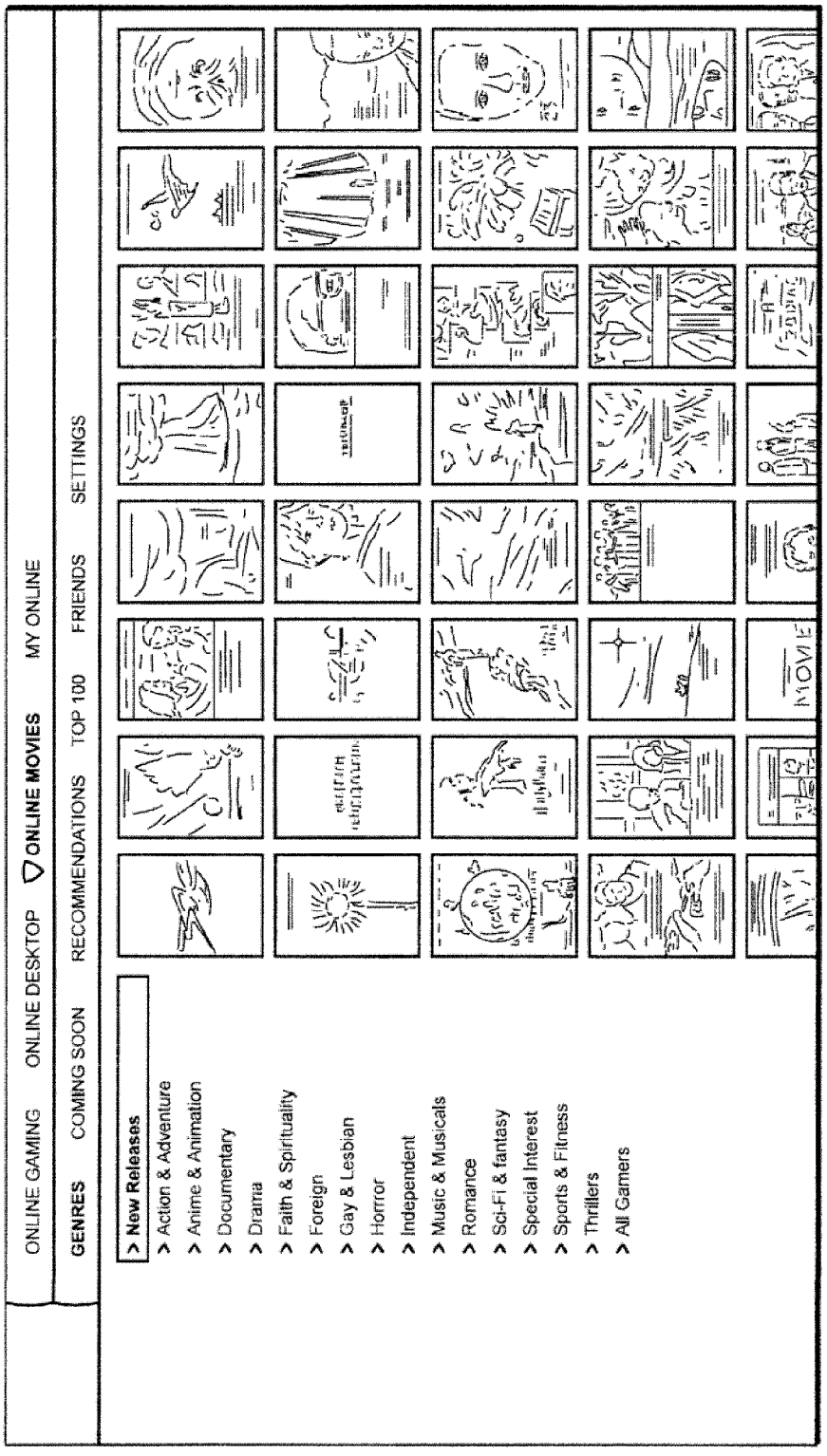

ONLINE GAMING
ONLINE DESKTOP
ONLINE MOVIES
MY ONLINE
GENRES
COMING SOON
RECOMMENDATIONS
TOP 100
FRIENDS
SETTINGS
> New Releases
> Action & Adventure
> Anime & Animation
> Documentary
> Drama
> Faith & Spirituality
> Foreign
> Gay & Lesbian
> Horror
> Independent
> Music & Musicals
> Romance
> Sci-Fi & fantasy
> Special Interest
> Sports & Fitness
> Thrillers
> All Gamers

图 23

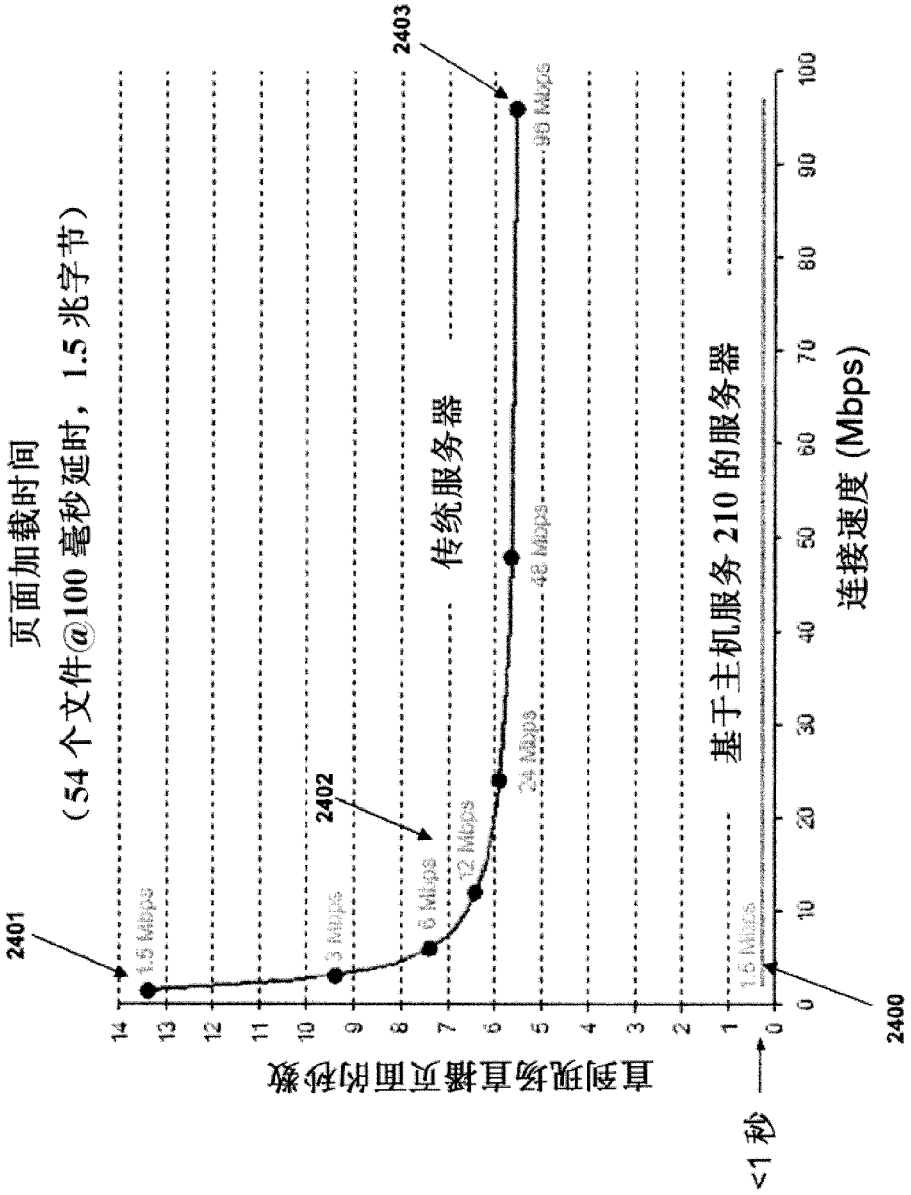

页面加载时间
（54 个文件@100 毫秒延时，1.5 兆字节）
直到现场直播页面的秒数
连接速度 (Mbps)
<1 秒
2400
2401
2402
2403
传统服务器
基于主机服务 210 的服务器
1.5 Mbps
3 Mbps
6 Mbps
12 Mbps
24 Mbps
48 Mbps
96 Mbps

图 24

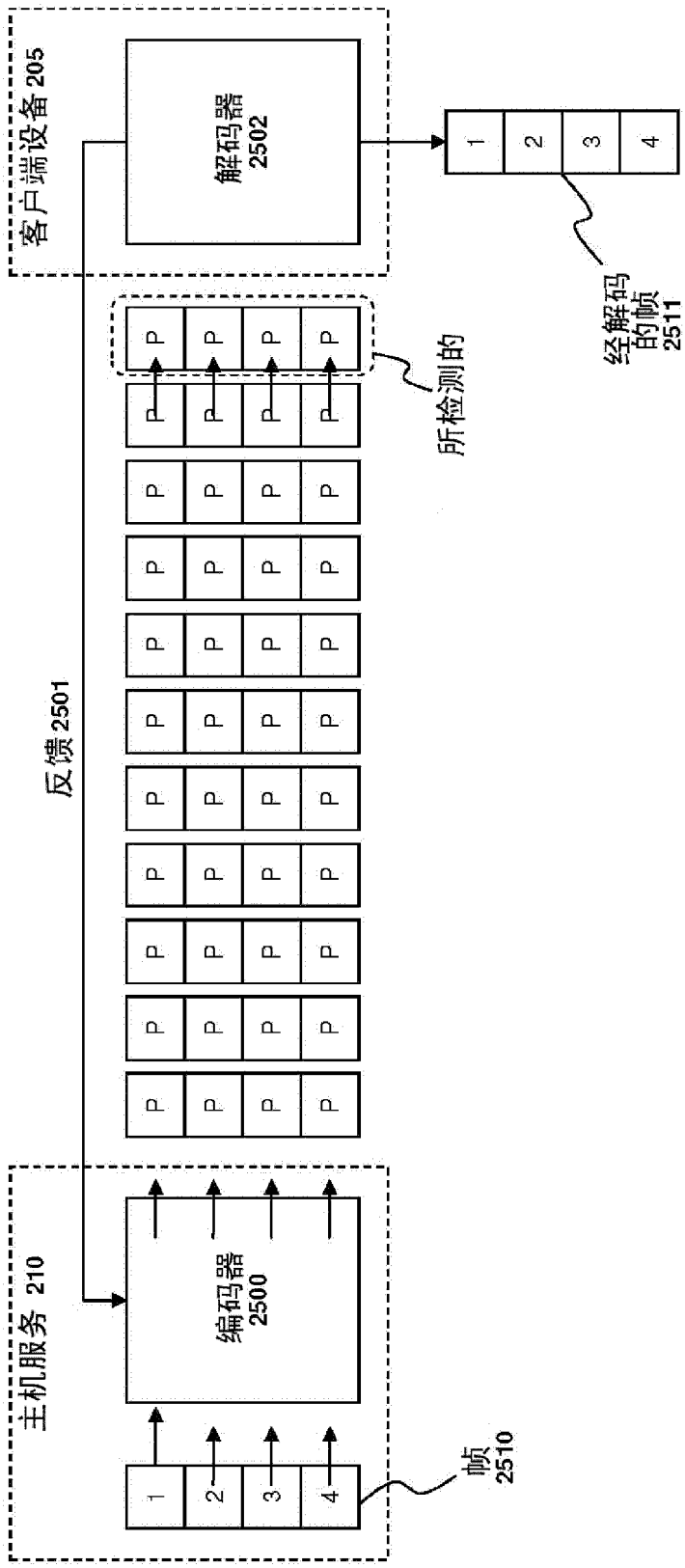

客户端设备 205
解码器 2502
1
2
3
4
经解码的帧 2511
所检测的
P
反馈 2501
主机服务 210
编码器 2500
帧 2510

图 25a

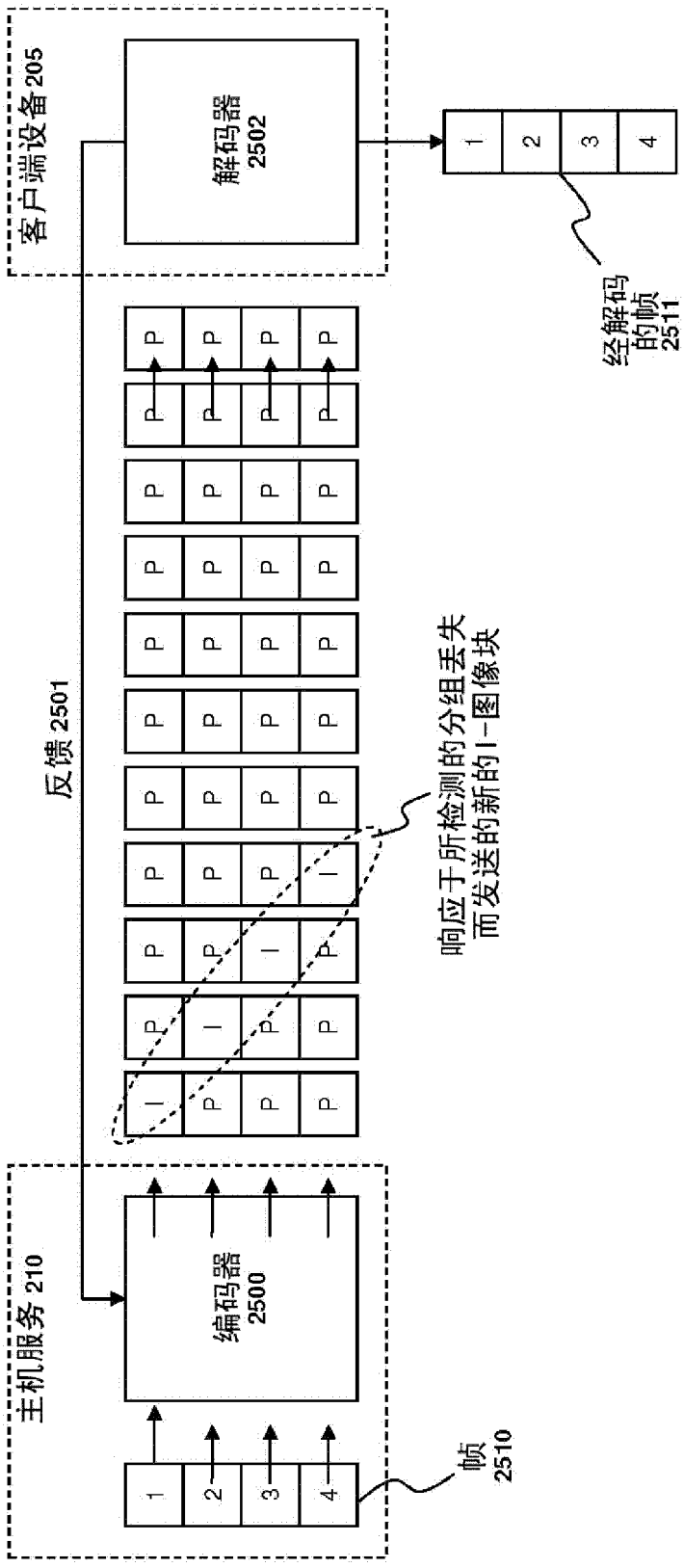

客户端设备 205
解码器 2502
1
2
3
4
经解码的帧 2511
反馈 2501
响应于所检测的分组丢失而发送的新的I-图像块
主机服务 210
编码器 2500
帧 2510

图 25b

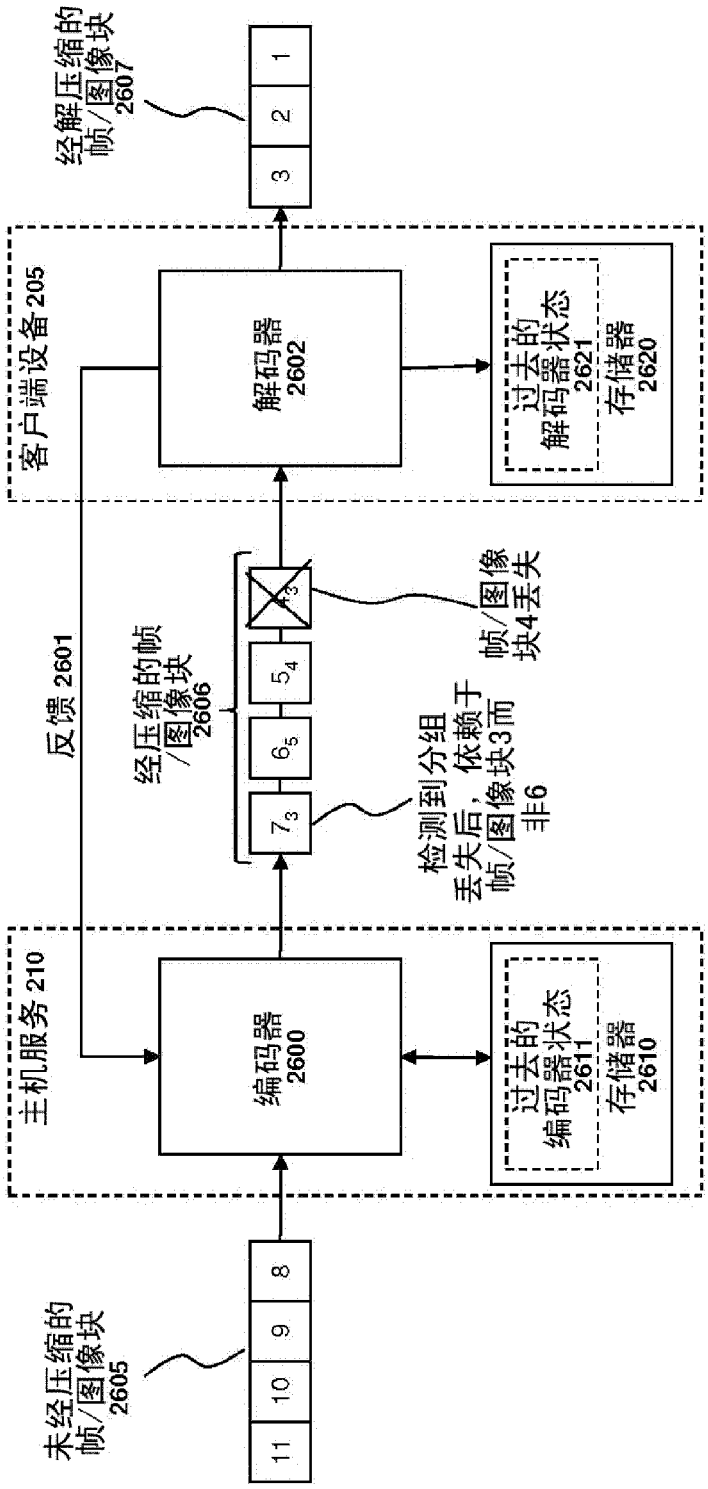

经解压缩的帧/图像块2607
1
2
3
客户端设备205
解码器2602
过去的解码器状态2621
存储器2620
帧/图像块4丢失
反馈2601
经压缩的帧/图像块2606
检测到分组丢失后，依赖于帧/图像块3而非6
主机服务210
编码器2600
过去的编码器状态2611
存储器2610
未经压缩的帧/图像块2605
8
9
10
11

图 26a

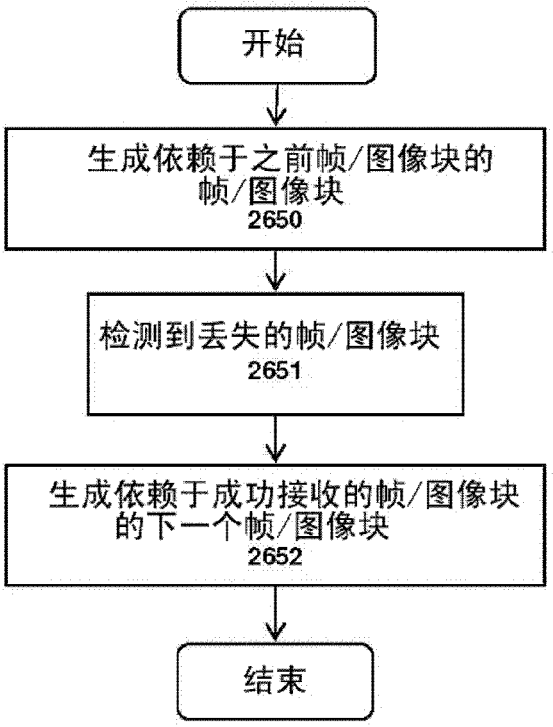

开始
生成依赖于之前帧/图像块的帧/图像块
2650
检测到丢失的帧/图像块
2651
生成依赖于成功接收的帧/图像块的下一个帧/图像块
2652
结束

图 26b

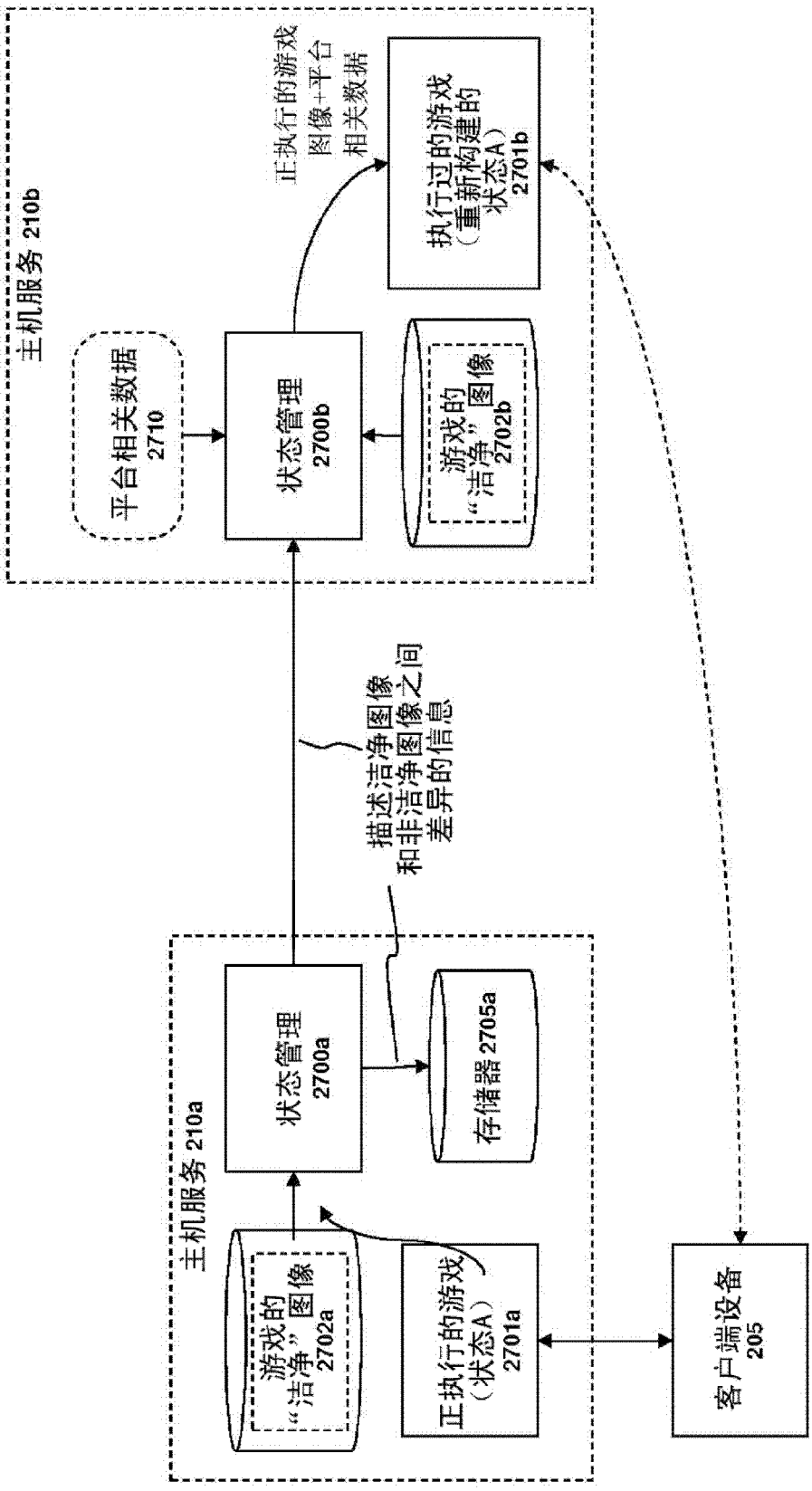

主机服务 210b
平台相关数据 2710
状态管理 2700b
游戏的“洁净”图像 2702b
正执行的游戏图像+平台相关数据
执行过的游戏（重新构建的状态A）2701b
描述洁净图像和非洁净图像之间差异的信息
主机服务 210a
状态管理 2700a
存储器 2705a
游戏的“洁净”图像 2702a
正执行的游戏（状态A）2701a
客户端设备 205

图 27a

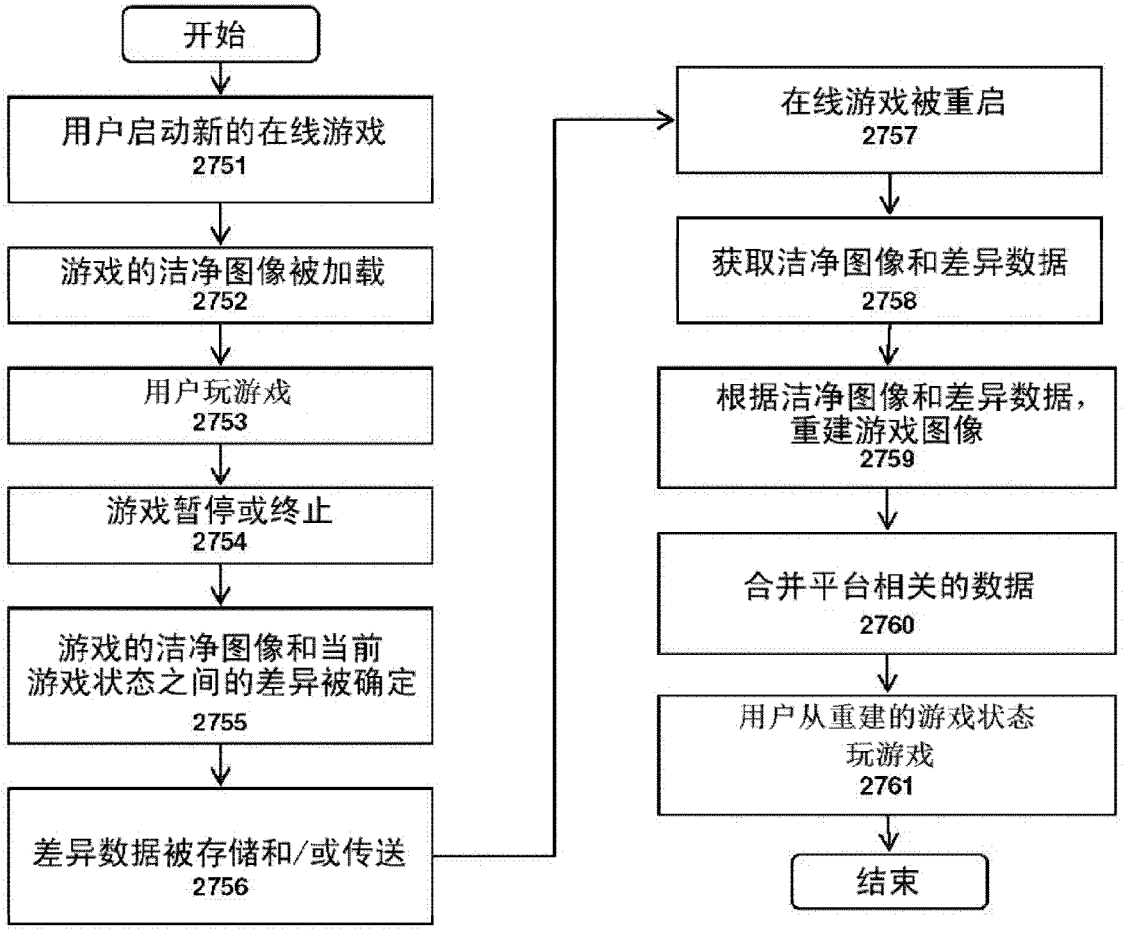

开始
用户启动新的在线游戏
2751
游戏的洁净图像被加载
2752
用户玩游戏
2753
游戏暂停或终止
2754
游戏的洁净图像和当前游戏状态之间的差异被确定
2755
差异数据被存储和/或传送
2756
在线游戏被重启
2757
获取洁净图像和差异数据
2758
根据洁净图像和差异数据，重建游戏图像
2759
合并平台相关的数据
2760
用户从重建的游戏状态玩游戏
2761
结束

图 27b

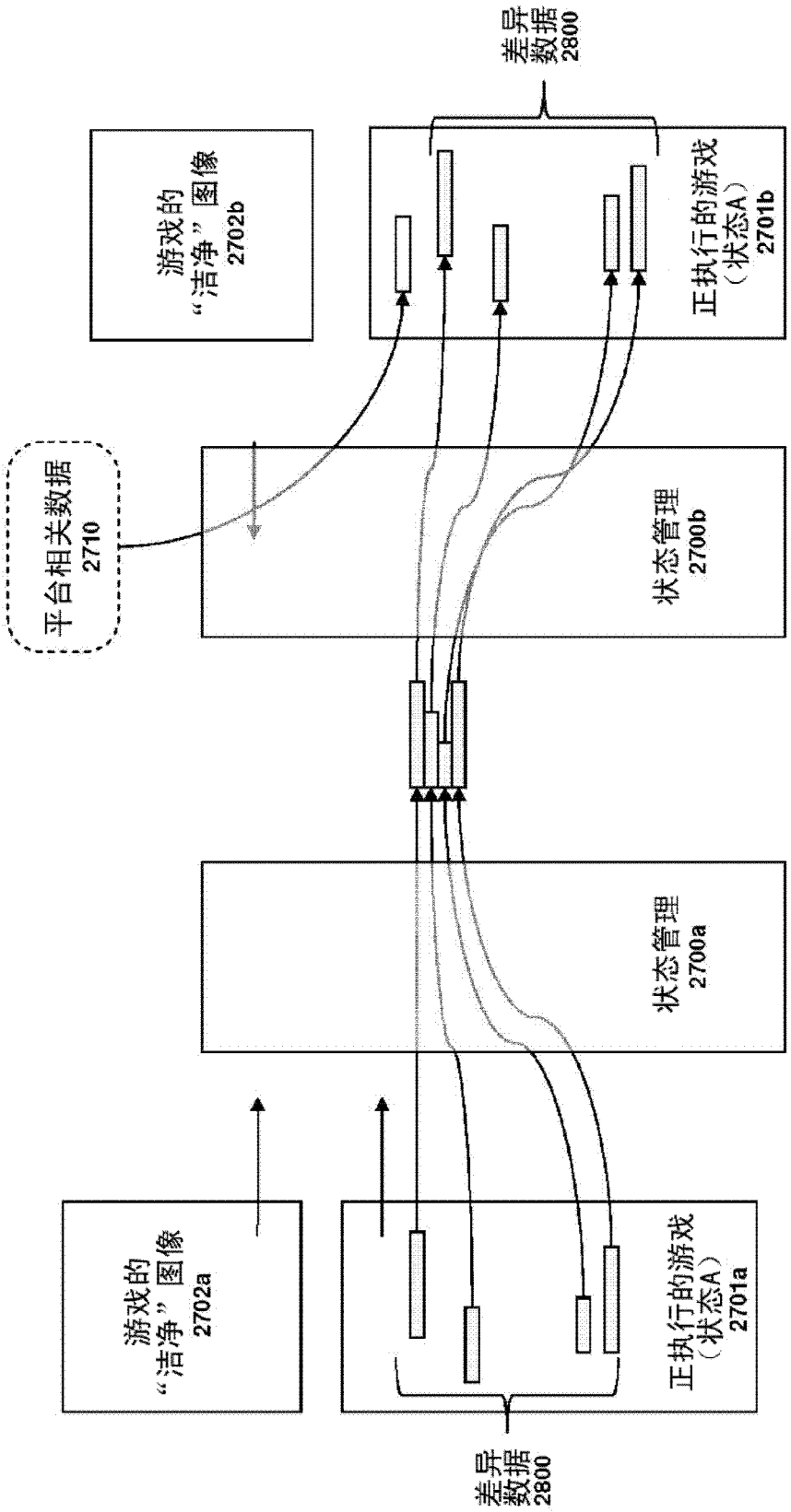

差异数据 2800
游戏的“洁净”图像 2702b
正执行的游戏（状态A） 2701b
平台相关数据 2710
状态管理 2700b
状态管理 2700a
游戏的“洁净”图像 2702a
正执行的游戏（状态A） 2701a
差异数据 2800

图 28

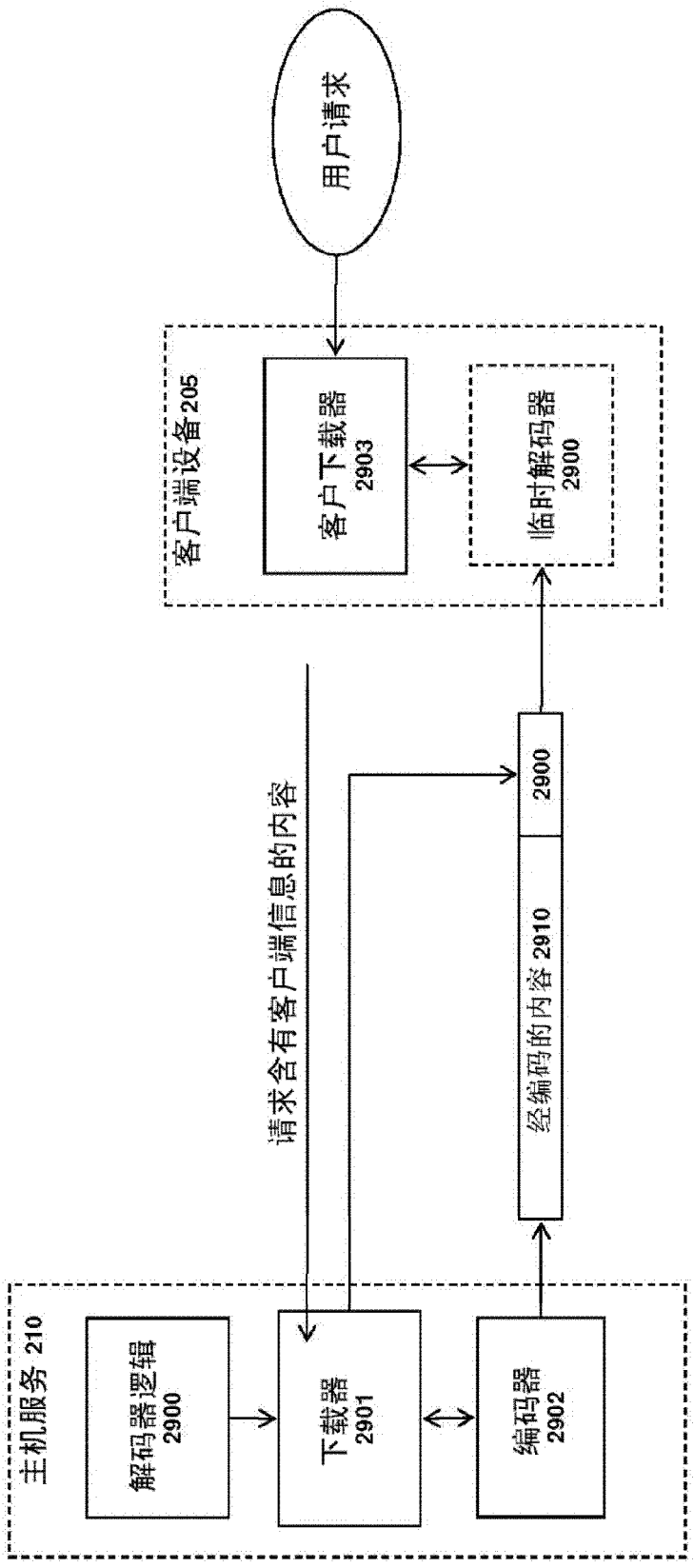

用户请求
客户端设备 205
客户下载器 2903
临时解码器 2900
请求含有客户端信息的内容
2900
经编码的内容 2910
主机服务 210
解码器逻辑 2900
下载器 2901
编码器 2902

图 29

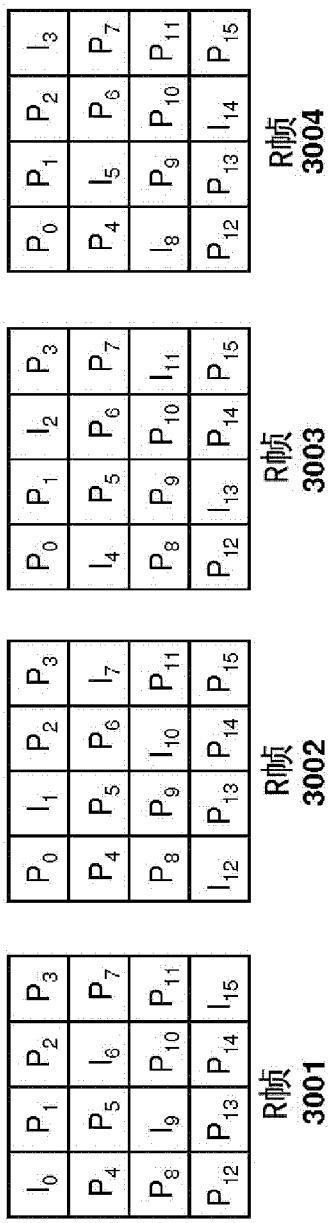

R帧 3001
R帧 3002
R帧 3003
R帧 3004

图 30

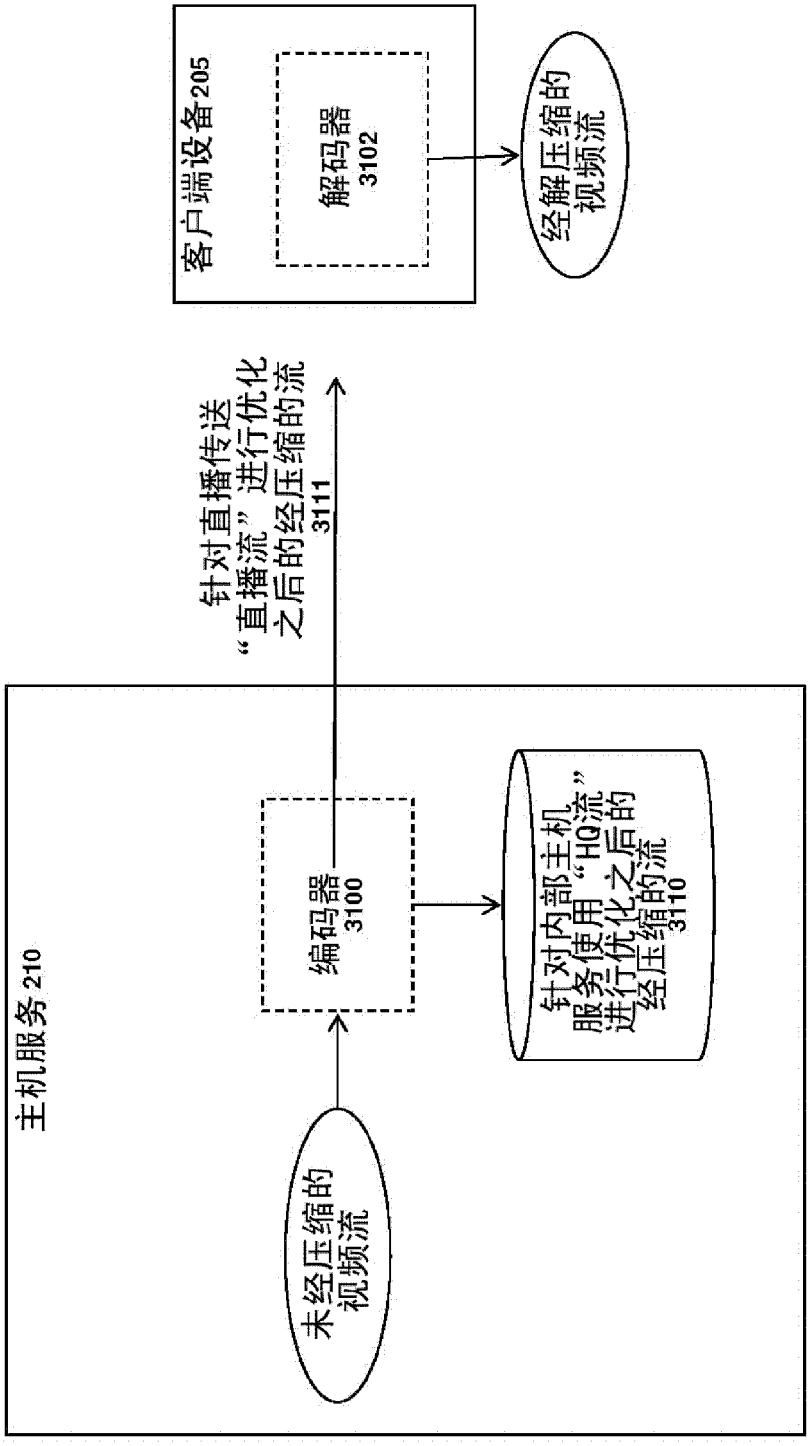

客户端设备205
解码器
3102
经解压缩的
视频流
针对直播传送
“直播流”进行优化
之后的经压缩的流
3111
主机服务 210
编码器
3100
针对内部主机
服务使用“HQ流”
进行优化之后的
经压缩的流
3110
未经压缩的
视频流

图 31a

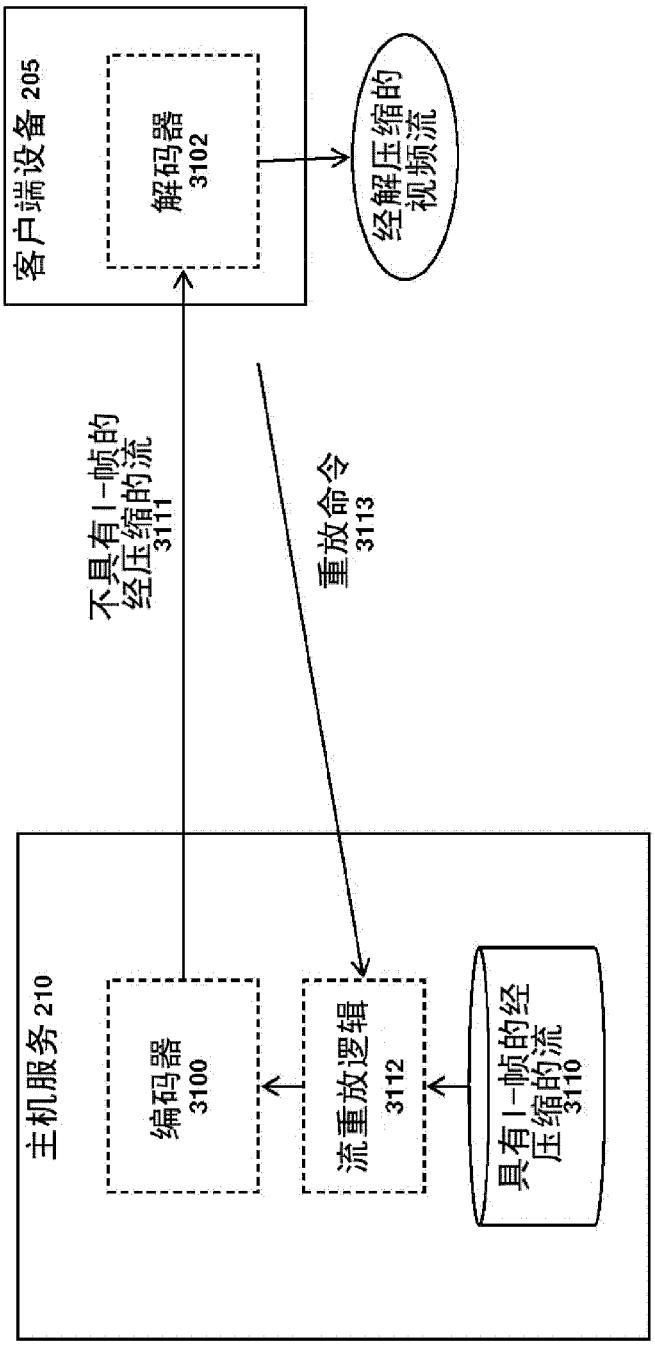

客户端设备 205
解码器
3102
经解压缩的
视频流
不具有I-帧的
经压缩的流
3111
重放命令
3113
主机服务 210
编码器
3100
流重放逻辑
3112
具有I-帧的经
压缩的流
3110

图 31b

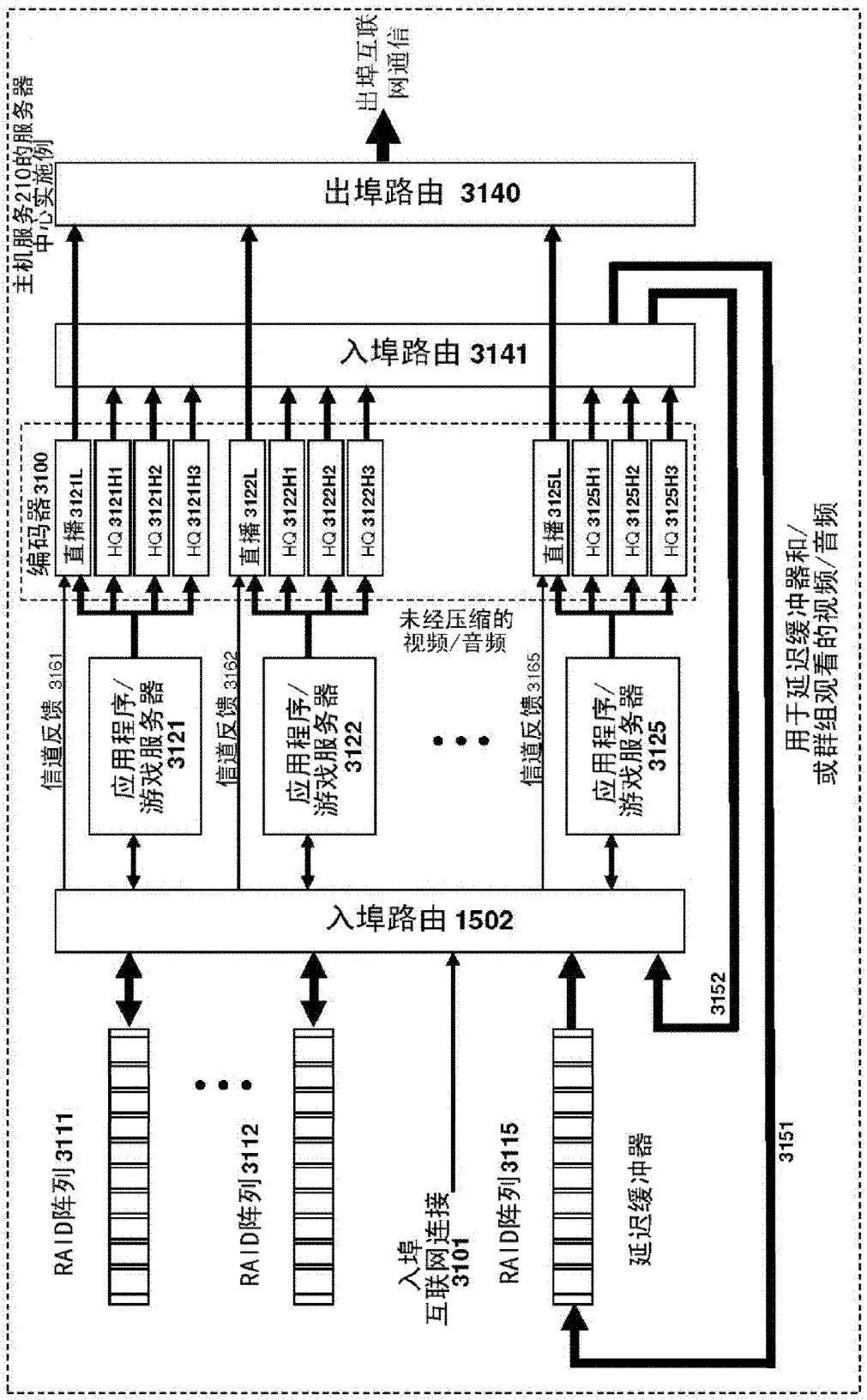

主机服务210的服务器中心实施例
出埠互联网通信
出埠路由 3140
入埠路由 3141
编码器3100
直播3121L
HQ 3121H1
HQ 3121H2
HQ 3121H3
直播3122L
HQ 3122H1
HQ 3122H2
HQ 3122H3
直播3125L
HQ 3125H1
HQ 3125H2
HQ 3125H3
未经压缩的视频/音频
信道反馈 3161
信道反馈 3162
信道反馈 3165
应用程序/游戏服务器 3121
应用程序/游戏服务器 3122
应用程序/游戏服务器 3125
入埠路由 1502
3152
用于延迟缓冲器和/或群组观看的视频/音频
RAID阵列3111
RAID阵列3112
入埠互联网连接 3101
RAID阵列3115
延迟缓冲器
3151

图 31c

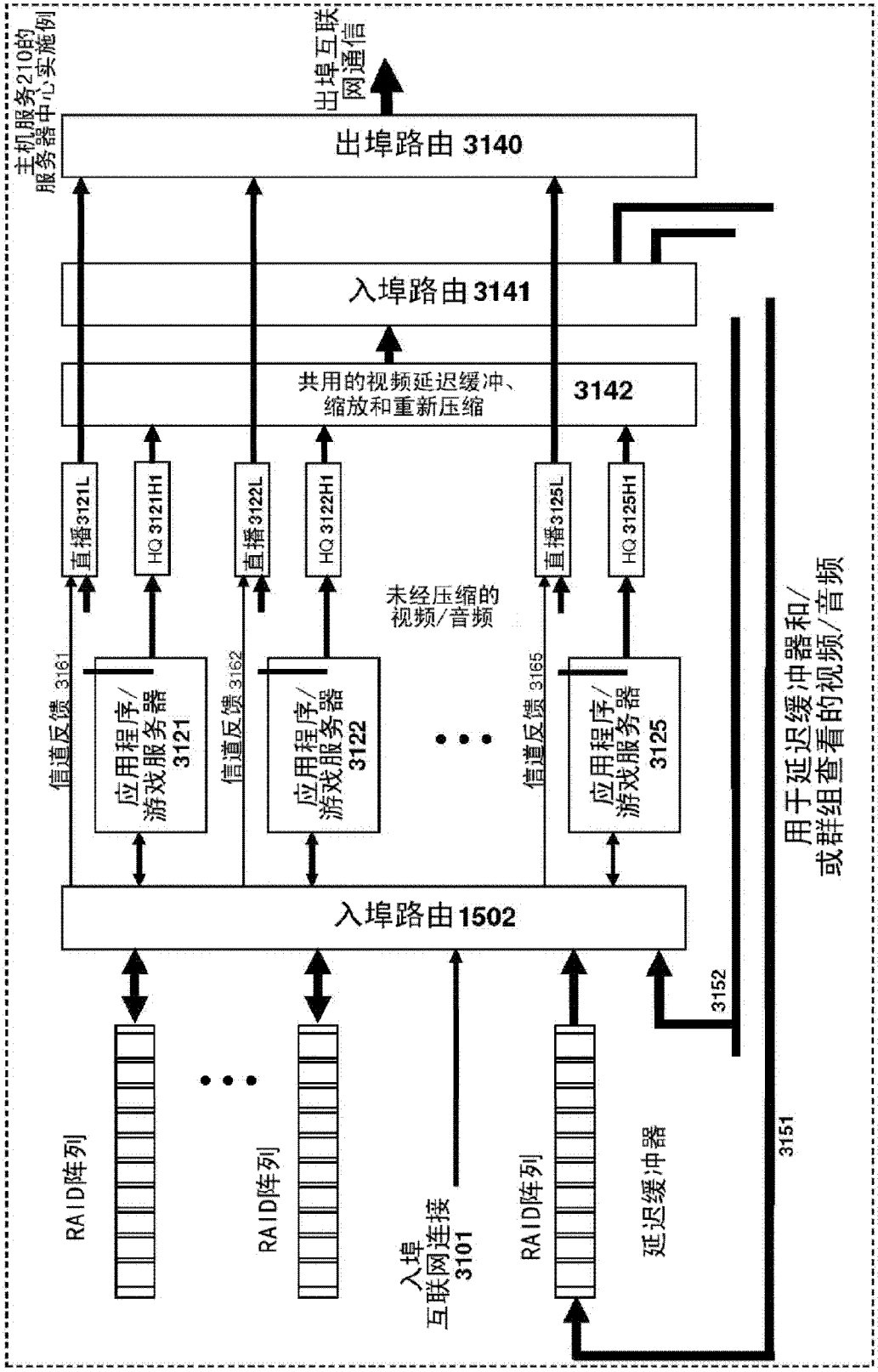

主机服务210的服务器中心实施例
出埠互联网通信
出埠路由 3140
入埠路由3141
共用的视频延迟缓冲、缩放和重新压缩
3142
直播3121L
HQ 3121H1
直播3122L
HQ 3122H1
直播3125L
HQ 3125H1
未经压缩的视频/音频
信道反馈 3161
信道反馈 3162
信道反馈 3165
应用程序/游戏服务器 3121
应用程序/游戏服务器 3122
应用程序/游戏服务器 3125
入埠路由1502
3152
用于延迟缓冲器和/或群组查看的视频/音频
3151
RAID阵列
RAID阵列
入埠互联网连接 3101
RAID阵列
延迟缓冲器

图 31d

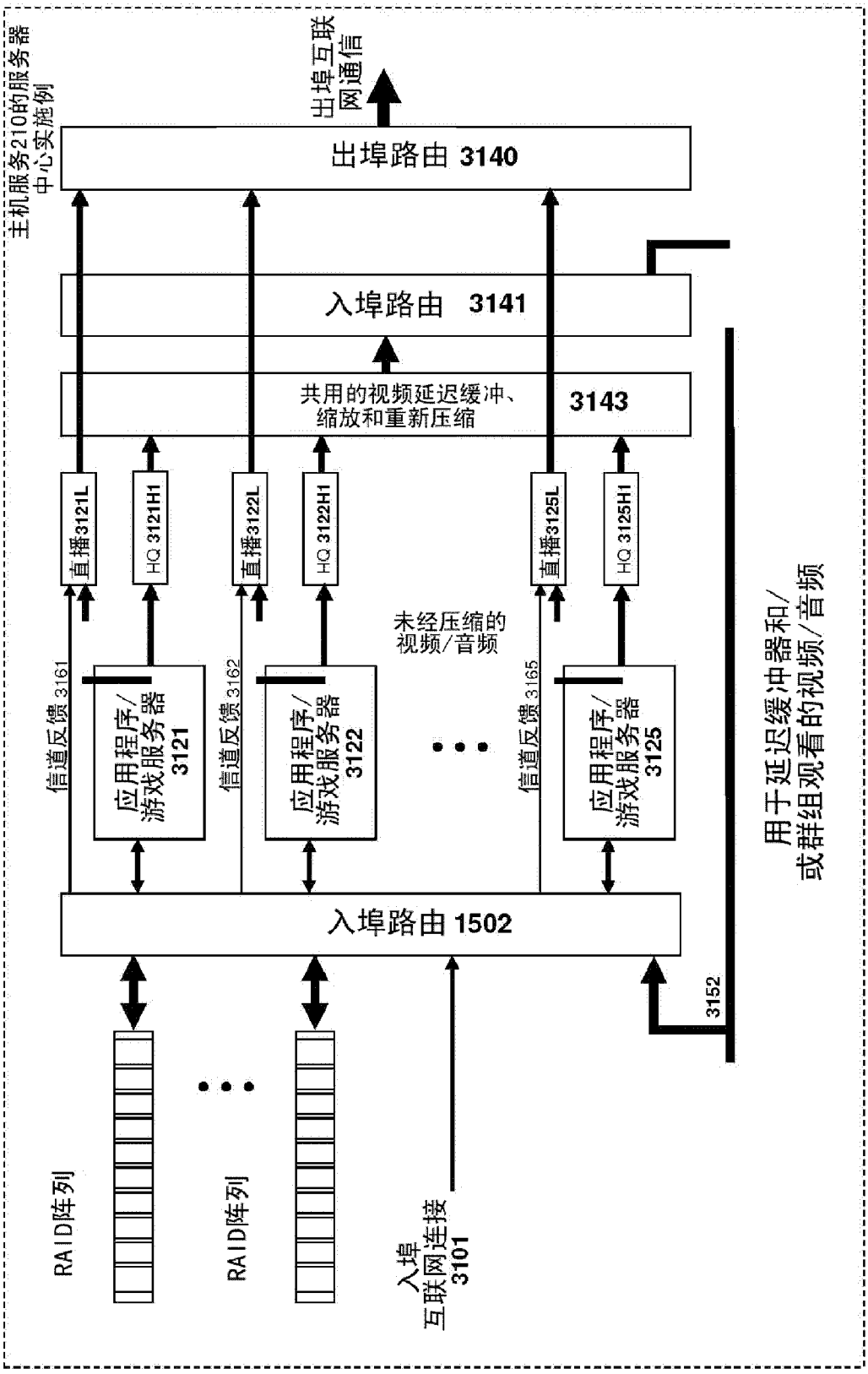

主机服务210的服务器中心实施例
出埠互联网通信
出埠路由 3140
入埠路由 3141
共用的视频延迟缓冲、缩放和重新压缩
3143
直播3121L
HQ 3121H1
直播3122L
HQ 3122H1
直播3125L
HQ 3125H1
未经压缩的视频/音频
信道反馈 3161
信道反馈 3162
信道反馈 3165
应用程序/游戏服务器 3121
应用程序/游戏服务器 3122
应用程序/游戏服务器 3125
入埠路由 1502
3152
用于延迟缓冲器和/或群组观看的视频/音频
RAID阵列
RAID阵列
入埠互联网连接 3101

图 31e

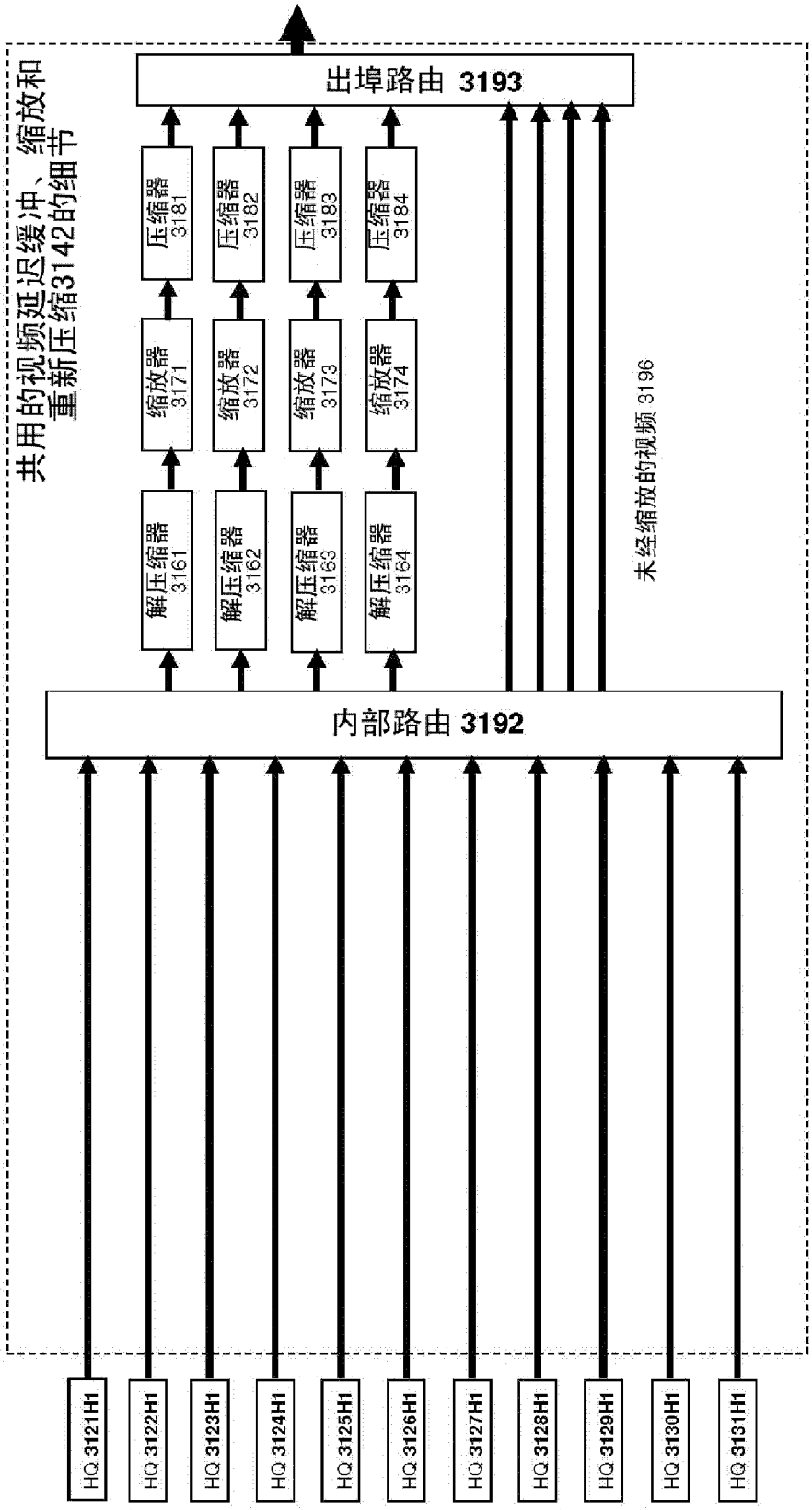

共用的视频延迟缓冲、缩放和
重新压缩3142的细节
出埠路由 3193
压缩器 3181
压缩器 3182
压缩器 3183
压缩器 3184
缩放器 3171
缩放器 3172
缩放器 3173
缩放器 3174
解压缩器 3161
解压缩器 3162
解压缩器 3163
解压缩器 3164
未经缩放的视频 3196
内部路由 3192
HQ 3121H1
HQ 3122H1
HQ 3123H1
HQ 3124H1
HQ 3125H1
HQ 3126H1
HQ 3127H1
HQ 3128H1
HQ 3129H1
HQ 3130H1
HQ 3131H1

图 31f

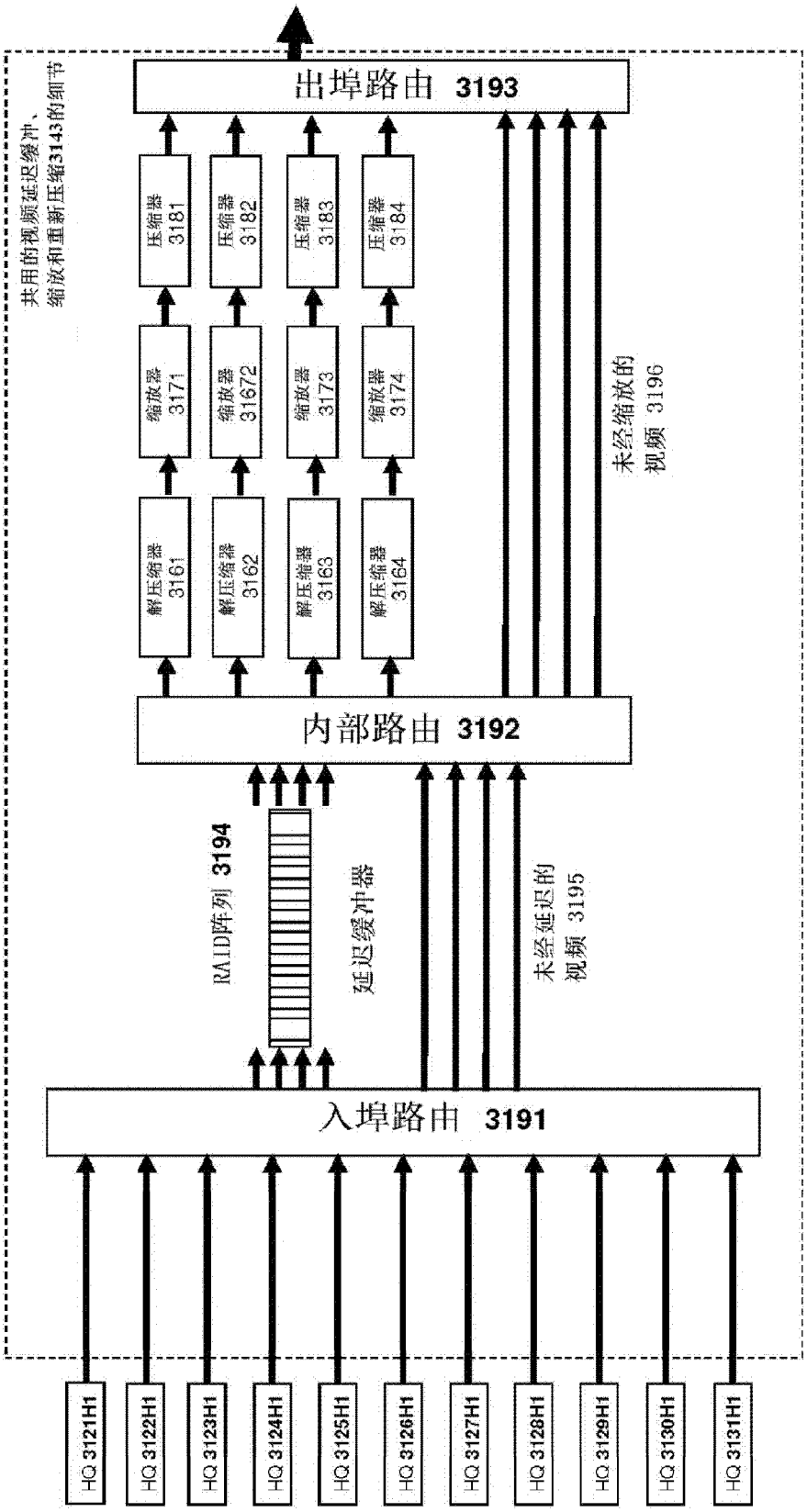

共用的视频延迟缓冲、
缩放和重新压缩3143的细节
出埠路由 3193
压缩器 3181
压缩器 3182
压缩器 3183
压缩器 3184
缩放器 3171
缩放器 31672
缩放器 3173
缩放器 3174
解压缩器 3161
解压缩器 3162
解压缩器 3163
解压缩器 3164
未经缩放的
视频 3196
内部路由 3192
RAID阵列 3194
延迟缓冲器
未经延迟的
视频 3195
入埠路由 3191
HQ 3121H1
HQ 3122H1
HQ 3123H1
HQ 3124H1
HQ 3125H1
HQ 3126H1
HQ 3127H1
HQ 3128H1
HQ 3129H1
HQ 3130H1
HQ 3131H1

图 31g

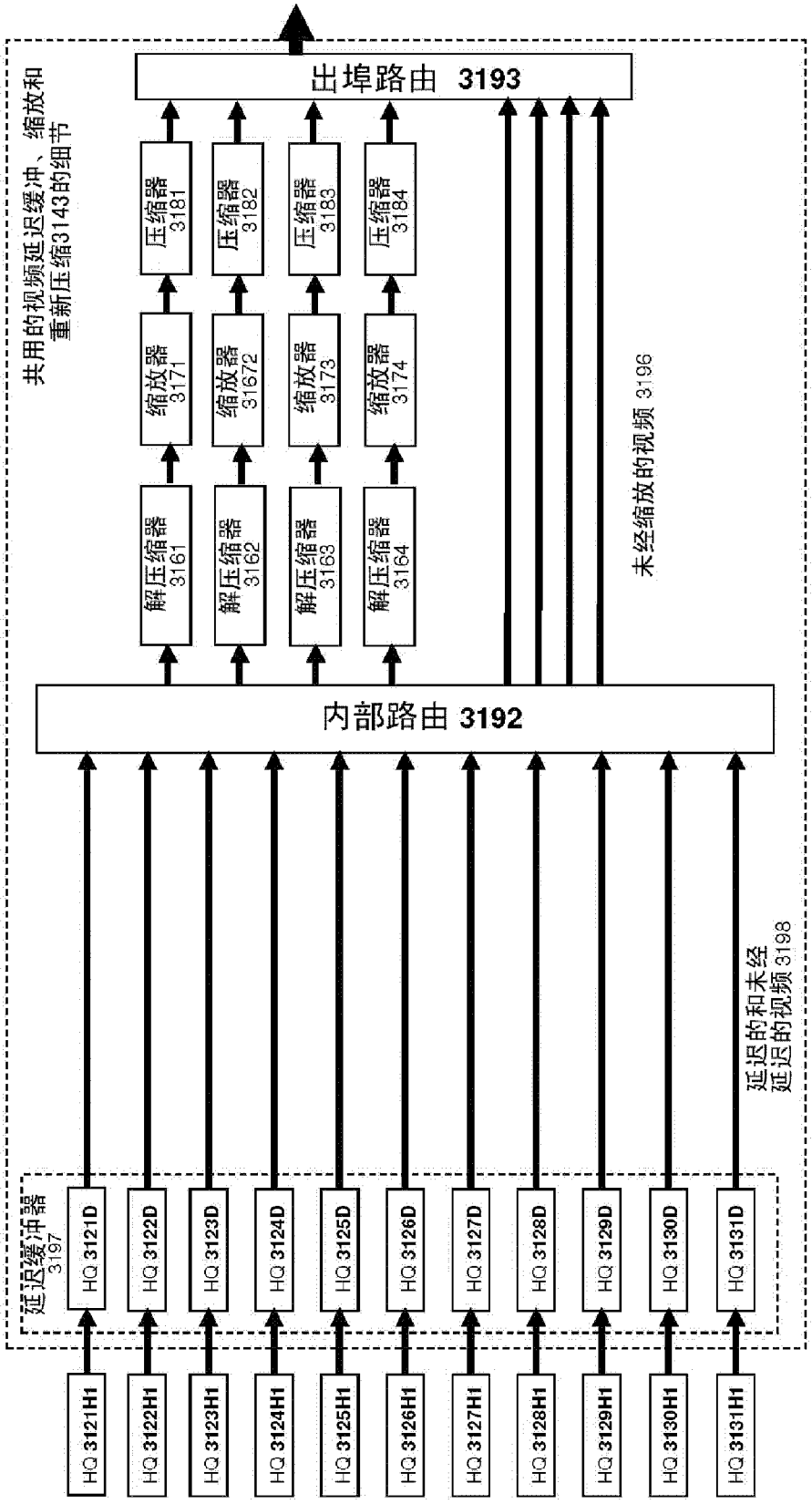

出埠路由 3193
共用的视频延迟缓冲、缩放和重新压缩3143的细节
压缩器 3181
压缩器 3182
压缩器 3183
压缩器 3184
缩放器 3171
缩放器 31672
缩放器 3173
缩放器 3174
解压缩器 3161
解压缩器 3162
解压缩器 3163
解压缩器 3164
未经缩放的视频 3196
内部路由 3192
延迟缓冲器 3197
HQ 3121D
HQ 3122D
HQ 3123D
HQ 3124D
HQ 3125D
HQ 3126D
HQ 3127D
HQ 3128D
HQ 3129D
HQ 3130D
HQ 3131D
延迟的和未经延迟的视频 3198
HQ 3121H1
HQ 3122H1
HQ 3123H1
HQ 3124H1
HQ 3125H1
HQ 3126H1
HQ 3127H1
HQ 3128H1
HQ 3129H1
HQ 3130H1
HQ 3131H1

图 31h