

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
26 October 2006 (26.10.2006)

PCT

(10) International Publication Number
WO 2006/113711 A2

(51) International Patent Classification:
H04J 3/02 (2006.01)

(21) International Application Number:
PCT/US2006/014559

(22) International Filing Date: 18 April 2006 (18.04.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/673,431 21 April 2005 (21.04.2005) US

(71) Applicant (for all designated States except US): **QUARTICS, INC.** [US/US]; 1 ADA PKWY, Suite 250, Irvine, California 92618 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **AHMED, Sherjil** [US/US]; 1 ADA PKWY, Suite 250, Irvine, California 92618 (US). **USMAN, Mohammad** [US/US]; 24302 Chrisanta Drive, Mission Viejo, California 92691 (US). **COWAN, Anthony, J.** [US/US]; 4768 Sw 3rd Ave., Ocala,

Florida (US). **ABHISHEK, Joshi** [IN/US]; 331 Oak Glen, Irvine, California 92618 (US). **SIDDIQUI, Mudeem, I.** [PK/US]; 265 Streamwood, Irvine, California 92620 (US). **RAULT, Patrick** [CA/CA]; 66 Cameron Ave., Toronto, M2N1E1 (CA). **SAQIB, Malik** [PK/US]; 1622 Piedmont, Irvine, California 92620 (US).

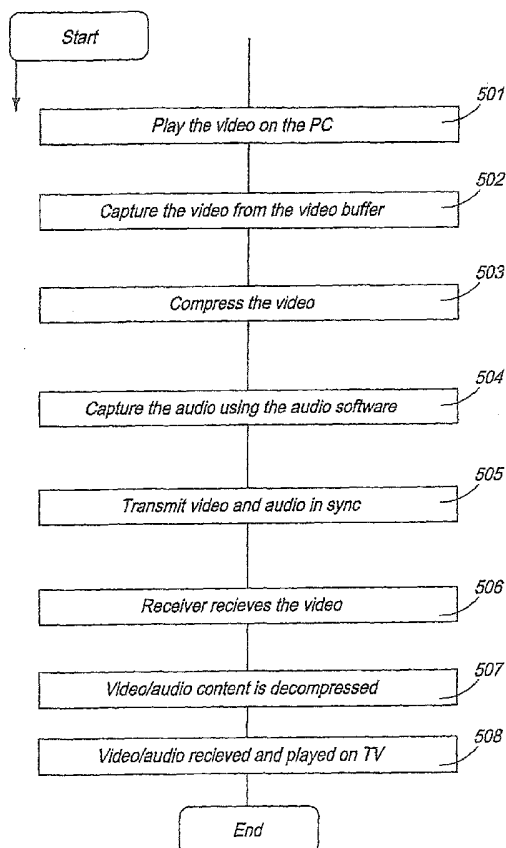
(74) Agent: **ANSARI, Hazim**; PATENTMETRIX, 14252 CULVER DR., Box 914, Irvine, California 92604 (US).

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

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH,

[Continued on next page]

(54) Title: INTEGRATED WIRELESS MULTIMEDIA TRANSMISSION SYSTEM



(57) Abstract: The present invention relates generally to methods and systems for the wireless real time transmission of data from a source to a monitor. The present invention further relates generally to the substantially automatic configuration of wireless devices. In an exemplary embodiment, the present invention is a method of capturing video and audio data from a source and wirelessly transmitting the data comprising the steps of playing the data, capturing the video data using a mirror display driver, capturing the audio data from an input source, compressing the captured audio and video data, and transmitting the compressed audio and video data using a transmitter.

WO 2006/113711 A2



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, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Integrated Wireless Multimedia Transmission System

Cross-Reference

The present invention relies for priority U.S. Provisional
5 Application No. 60/673,431 filed on April 18, 2005.

Field of the Invention

The present invention relates generally to methods and
systems for the wireless real time transmission of data from a
10 source to a monitor. The present invention further relates
generally to the substantially automatic configuration of
wireless devices.

Background of the Invention

15 Individuals use their computing devices, including personal
computers, storage devices, mobile phones, personal data
assistants, and servers, to store, record, transmit, receive,
and playback media, including, but not limited to, graphics,
text, video, images, and audio. Such media may be obtained from
20 many sources, including, but not limited to, the Internet, CDs,
DVDs, other networks, or other storage devices. In particular,
individuals are able to rapidly and massively distribute and
access media through open networks, often without time,
geographic, cost, range of content or other restrictions.
25 However, individuals are often forced to experience the obtained
media on small screens that are not suitable for audiences in
excess of one or two people.

Despite the rapid growth and flexibility of using computing
devices to store, record, transmit, receive, and playback media,
30 a vast majority of individuals throughout the world still use
televisions as the primary means by which they receive
audio/video transmissions. Specifically, over the air,

satellite, and cable transmissions to televisions still represent the dominant means by which audio/video media is communicated to, and experienced by, individuals. Those transmissions, however, are highly restricted in terms of cost, range of content, access time and geography.

Given the ubiquity of individual computing devices being used to store, record, transmit, receive, and playback media, it would be preferred to be able to use those same computing devices, in conjunction with the vast installed base of televisions, to allow individuals to rapidly and flexibly obtain media and, yet, still use their televisions to experience the media.

Prior attempts at enabling the integration of computing devices with televisions have focused on a) transforming the television into a networked computing appliance that directly accesses the Internet to obtain media, b) creating a specialized hardware device that receives media from a computing device, stores it, and, through a wired connection, transfers it to the television, and/or c) integrating into the television a means to accept storage devices, such as memory sticks. However, these conventional approaches suffer from having to substantially modify existing equipment, i.e. replacing existing computing devices and/or televisions, or purchasing expensive new hardware. Additionally, both approaches have typically required the use of multiple physical hard-wired connections to transmit graphics, text, audio, and video. Such physical connections limit the use of devices to a single television, limit the placement of equipment to a particular area in the home, and result in an unsightly web of wires. Finally, the requirement to physically store media to a storage element, such as a memory stick, and then input into the television is not only cumbersome

and inflexible, but highly limited in the amount of data that can be transferred.

There is therefore still a need for methods, devices, and systems that enable individuals to use existing computing
5 devices to receive, transmit, store, and playback media and to use existing televisions to experience the media. There is also a need for a simple, inexpensive way to wireless transmit media from a computing device to a television, thereby transforming the television in a remote monitor. It would also be preferred
10 if numerous diverging standards applicable to text, graphics, video, audio transmission can be managed by a single, universal wireless media transmission system. Finally, there is a need for the convenient, automated configuration of wireless devices.

15 Summary of the Invention

The present invention relates generally to methods and systems for the wireless real time transmission of data from a source to a monitor. The present invention further relates generally to the substantially automatic configuration of
20 wireless devices.

In one embodiment, the present invention is a method of capturing media from a source and wirelessly transmitting said media, comprising the steps of: playing said media, comprising at least audio data and video data, on a computing device;
25 capturing said video data using a mirror display driver; capturing said audio data from an input source; compressing said captured audio and video data; and transmitting said compressed audio and video data using a transmitter.

Optionally, the method further comprises the step of
30 receiving said media at a receiver, decompressing said captured audio and video data, and playing said decompressed audio and video data on a display remote from said source. Optionally,

the transmitter and receiver establish a connection using TCP and the transmitter transmits packets of video data using UDP.

Optionally, the media further comprises graphics and text data and wherein said graphics and text data is captured
5 together with said video data using the mirror display driver. Optionally, the method further comprising the step of processing said video data using a CODEC. Optionally, the CODEC removes temporal redundancy from the video data using a motion
estimation block. Optionally, the CODEC converts a frame of
10 video data into 8*8 blocks or 4*4 blocks of pixels using a DCT transform block. Optionally, the CODEC codes video content into shorter words using a VLC coding circuit. Optionally, the CODEC converts back spatial frequencies of the video data into the pixel domain using an IDCT block. Optionally, the CODEC
15 comprises a rate control mechanism for speeding up the transmission of media.

In another embodiment, the present invention comprises a program stored on a computer-readable substrate for capturing media, comprising at least video data, from a source and
20 wirelessly transmitting said media, comprising a mirror display driver operating in a kernel mode for capturing said video data; a CODEC for processing said video data; and a transmitter for transmitting said processed video data.

Optionally, the program further comprises a virtual display
25 driver. Optionally, the transmitter establishes a connection with a receiver using TCP and the transmitter transmits packets of video data using UDP. Optionally, the media further comprises graphics and text data and said mirror display driver captures graphics and text data together with said video data.
30 Optionally, the CODEC comprises a motion estimation block for removing temporal redundancy from the video data. Optionally, the CODEC comprises a DCT block for converting a frame of video

data into 8*8 or 4*4 blocks of pixels. Optionally, the CODEC comprises a VLC coding circuit for coding video content into shorter words. Optionally, the CODEC comprises an IDCT block for converting back spatial frequencies of the video data into the pixel domain. Optionally, the CODEC comprises a rate control mechanism for speeding up the transmission of media.

These, and other embodiments, will be described in greater clarity in the Detailed Description and with reference to a Brief Description of the Drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages of the present invention will be appreciated, as they become better understood by reference to the following Detailed Description when considered in connection with the accompanying drawings, wherein:

Figure 1 depicts a block diagram of the integrated wireless media transmission system of the present invention;

Figure 2 depicts the components of a transmitter of one embodiment of the present invention;

Figure 3 depicts a plurality of software modules comprising one embodiment of a software implementation of the present invention;

Figure 4 depicts the components of a receiver of one embodiment of the present invention;

Figure 5 is a flowchart depicting an exemplary operation of the present invention;

Figure 6 depicts one embodiment of the TCP/UDP RT hybrid protocol header structures of the present invention;

Figure 7 is a flowchart depicting exemplary functional steps of the TCP/UDP RT transmission protocol of the present invention;

Figure 8 depicts a block diagram of an exemplary codec used in the present invention;

Figure 9 is a functional diagram of an exemplary motion estimation block used in the present invention;

5 Figure 10 depicts one embodiment of the digital signal waveform and the corresponding data transfer;

Figure 11 is a block diagram of an exemplary video processing and selective optimization of the IDCT block of the present invention;

10 Figure 12 is a block diagram depicting the components of the synchronization circuit for synchronizing audio and video data of the present invention;

Figure 13 is a flowchart depicting another embodiment of synchronizing audio and video signals of the present invention;

15 Figure 14 depicts another embodiment of the audio and video synchronizaton circuit of the present invention;

Figure 15 depicts an enterprise configuration for automatically downloading and updating the software of the present invention;

20 Figure 16 is a schematic diagram depicting the communication between a transmitter and plurality of receivers;

Figure 17 depicts a block diagram of a Microsoft Windows framework for developing display drivers;

25 Figure 18 depicts a block diagram of an interaction between a GDI and a display driver; and

Figure 19 depicts a block diagram of a DirectDraw architecture.

Detailed Description

30 The present invention is an integrated wireless system for transmitting media wirelessly from one device to another device in real time. The present invention will be described with

reference to the aforementioned drawings. The embodiments described herein are not intended to be exhaustive or to limit the invention to the precise forms disclosed. They are chosen to explain the invention and its application and to enable others skilled in the art to utilize the invention.

Referring to Figure 1, a computing device 101, such as a conventional personal computer, desktop, laptop, PDA, mobile telephone, gaming station, set-top box, satellite receiver, DVD player, personal video recorder, or any other device, operating the novel systems of the present invention communicates through a wireless network 102 to a remote monitor 103. Preferably, the computing device 101 and remote monitor 103 further comprise a processing system on a chip capable of wirelessly transmitting and receiving graphics, audio, text, and video encoded under a plurality of standards. The remote monitor 103 can be a television, plasma display device, flat panel LCD, HDD, projector or any other electronic display device known in the art capable of rendering graphics, audio and video. The processing system on chip can either be integrated into the remote monitor 103 and computing device 101 or incorporated into a standalone device that is in wired communication with the remote monitor 103 or computing device 101. An exemplary processing system on a chip is described in PCT/US2006/00622, which is also assigned to the owner of the present application, and incorporated herein by reference.

Referring to Figure 2, a computing device 200 of the present invention is depicted. Computing device 200 comprises an operating system 201 capable running the novel software systems of the present invention 202 and a transceiver 203. The operating system 201 can be any operating system including but not limited to MS Windows 2000, MS Windows NT, MS Windows XP, Linux, OS/2, Palm-based operating systems, cell phone operating

systems, iPod operating systems, and MAC OS. The computing device 200 transmits media using appropriate wireless standards for the transmission of graphics, text, video and audio signals, for example, IEEE 802.11a, 802.11g, Bluetooth2.0, HomeRF 2.0, 5 HiperLAN/2, and Ultra Wideband, among others, along with proprietary extensions to any of these standards.

Referring to Figure 3, modules of the novel software system 300 of the present invention is depicted. The software 300 comprises a module for the real-time capture of media 301, a 10 module for managing a buffer for storing the captured media 302, a codec 303 for compressing and decompressing the media, and a module for packaging the processed media for transmission 304. In one embodiment, the computing device receives media from a source, whether it be downloaded from the Internet, real-time 15 streamed from the Internet, transmitted from a cable or satellite station, transferred from a storage device, or any other source. The media is played on the computing device via suitable player installed on the computing device. While the media is played on the computing device, the software module 301 20 captures the data in real time and temporarily stores it in the buffer before transmitting it to the CODEC. The CODEC 303 compresses it and prepares it for transmission.

Referring to Figure 4, a receiver of the present invention is depicted. The receiver 400 comprises a transceiver 401, a 25 CODEC 402, a display device 403 for rendering video and graphics data and an audio device 404 for rendering the audio data. The transceiver 401 receives the compressed media data, preferably through a novel transmission protocol used by the present invention. In one example, the novel transmission protocol is a 30 TCP/UDP hybrid protocol. The TCP/UDP hybrid protocol for the real-time transmission of packets combines the security services of TCP with the simplicity and lower processing requirements of

UDP. The content received by the receiver is then transmitted to the CODEC 402 for decompression. The CODEC decompresses the media and prepares the video and audio signals, which are then transmitted to the display device 403 and speakers 404 for rendering.

Referring to Figure 5, the flowchart depicts an exemplary operation of the integrated wireless system of the present invention. The personal computer plays 501 the media using appropriate media player on its console. Such media player can include players from Apple (iPod), RealNetworks (RealPlayer), Microsoft (Windows Media Player), or any other media player. The software of the present invention captures 502 the real time video directly from the video buffer. The captured video is then compressed 503 using the CODEC. Similarly, the audio is captured 504 using the audio software operating on the computing device and is compressed using the CODEC.

In one embodiment, the software of the present invention captures video through the implementation of software modules comprising a mirror display driver and a virtual display driver. In one embodiment, the mirror display driver and virtual display driver are installed as components in the kernel mode of the operating system running on the computer that hosts the software of the present invention.

A mirror display driver for a virtual device mirrors the operation of a physical display device driver by mirroring the operations of the physical display device driver. In one embodiment, a mirror display driver is used for capturing the contents of a primary display associated with the computer while a virtual display driver is used to capture the contents of an "extended desktop" or a secondary display device associated with the computer.

In use, the operating system renders graphics and video content onto the video memory of a virtual display driver and a mirror display driver. Therefore, any media being played by the computer using, for example, a media player is also rendered on one of these drivers. An application component of the software of the present invention maps the video memory of virtual display driver and mirror display driver in the application space. In this manner, the application of the present inventions obtains a pointer to the video memory. The application of the present invention captures the real-time images projected on the display (and, therefore, the real-time graphics or video content that is being displayed) by copying the memory from the mapped video memory to locally allocated memory.

In one embodiment, the mirror display driver and virtual display driver operate in the kernel space of a Microsoft operating system, such as a Windows 2000/NT compatible operating system. Referring to Figure 17, a Microsoft Windows framework 1700 for developing display drivers is shown. An application 1701 running on the computer issues a call to a graphics display interface, referred to as the Win32 GDI (Graphics Display Interface) 1702. The GDI 1702 issues graphics output requests. These requests are routed to software operating in the kernel space, including a kernel-mode GDI 1705. The kernel-mode GDI 1705 is an intermediary support between a kernel-mode graphics driver 1706 and an application 1701. Kernel-mode GDI 1705 sends these requests to an appropriate miniport 1709 or graphics driver, such as a display driver 1706 or printer driver [not shown].

For every display driver (DDI) there is a corresponding video miniport 1709. The miniport driver 1709 is written for one graphics adapter (or family of adapters). The display driver 1706 can be written for any number of adapters that share a

common drawing interface. This is because the display driver draws, while the miniport driver performs operations such as mode sets and provides information about the hardware to the driver. It is also possible for more than one display driver to
5 work with a particular miniport driver. The active component in this architecture is the Win32-GDI process 1702 and the application 1701. The rest of the components 1705-1710 are called from the Win32-GDI process 1702.

The video miniport driver 1709 generally handles operations
10 that interact with other kernel components 1703. For example, operations such as hardware initialization and memory mapping require action by the NT I/O subsystem. Video miniport driver 1709 responsibilities include resource management, such as hardware configuration, and physical device memory mapping. The
15 video miniport driver 1709 is specific to the video hardware. The display driver 1706 uses the video miniport driver 1709 for operations that are not frequently requested; for example, to manage resources, perform physical device memory mapping, ensure that register outputs occur in close proximity, or respond to
20 interrupts. The video miniport driver 1709 also handles mode set interaction with the graphics card, multiple hardware types (minimizing hardware-type dependency in the display driver), and mapping the video register into the display driver's 1706 address space.

25 There are certain functions that a driver writer should implement in order to write to a miniport. These functions are exported to the video port with which the miniport interacts. The driver writer specifies the absolute addresses of the video memory and registers, present on the video card, in miniport.
30 These addresses are first converted to bus relative addresses and then to virtual addresses in the address space of the calling process.

The display driver's 1706 primary responsibility is rendering. When an application calls a Win32 function with device-independent graphics requests, the Graphics Device Interface (GDI) 1705 interprets these instructions and calls the display driver 1706. The display driver 1706 then translates these requests into commands for the video hardware to draw graphics on the screen.

The display driver 1706 can access the hardware directly. By default, GDI 1705 handles drawing operations on standard format bitmaps, such as on hardware that includes a frame buffer. A display driver 1706 can hook and implement any of the drawing functions for which the hardware offers special support. For less time-critical operations and more complex operations not supported by the graphics adapter, the driver 1706 can push functions back to GDI 1705 and allow GDI 1705 to do the operations. For especially time-critical operations, the display driver 1706 has direct access to video hardware registers. For example, the VGA display driver for x86 systems uses optimized assembly code to implement direct access to hardware registers for some drawing and text operations.

Apart from rendering, display driver 1706 performs other operations such as surface management and palette management. Referring to Figure 18, a plurality of inputs and outputs between the GDI and display driver are shown. In one embodiment, GDI 1801 issues a DrvEnableDriver command 1810 to the display driver 1802. GDI 1801 then issues a DrvEnablePDEV command 1811 to the display driver 1802. GDI 1801 then receives a EngCreatePalette command 1812 from the display driver 1802. GDI 1801 then issues a DrvCompletePDEV command 1813 to the display driver 1802. GDI 1801 then issues a DrvEnableSurface command 1814 to the display driver 1802. GDI 1801 then receives a EngCreateDevicSurface command 1815 from the display driver

1802 and a EngModifySurface command 1816 from the display driver 1802.

Referring to Figure 19, a software architecture 1900 for a graphics generation system is shown. The software architecture 1900 represents Microsoft's DirectDraw, which includes the following components:

1. User-mode DirectDraw that is loaded and called by DirectDraw applications. This component provides hardware emulation, manages the various DirectDraw objects, and provides display memory and display hardware management services.
2. Kernel-mode DirectDraw, the system-supplied graphics engine that is loaded by a kernel-mode display driver. This portion of DirectDraw performs parameter validation for the driver, making it easier to implement more robust drivers. Kernel-mode DirectDraw also handles synchronization with GDI and all cross-process states.
3. The DirectDraw portion of the display driver, which, along with the rest of the display driver, is implemented by graphics card hardware vendors. Other portions of the display driver handle GDI and other non-DirectDraw related calls.

When DirectDraw 1900 is invoked, it accesses the graphics card directly through the DirectDraw driver 1902. DirectDraw 1900 calls the DirectDraw driver 1902 for supported hardware functions, or the hardware emulation layer (HEL) 1903 for functions that must be emulated in software. GDI 1905 calls are sent to the driver.

At initialization time and during mode changes, the display driver returns capability bits to DirectDraw 1900. This enables DirectDraw 1900 to access information about the available driver

functions, their addresses, and the capabilities of the display card and driver (such as stretching, transparent bits, display pitch, and other advanced characteristics). Once DirectDraw 1900 has this information, it can use the DirectDraw driver to access the display card directly, without making GDI calls or using the GDI specific portions of the display driver. In order to access the video buffer directly from the application, it is necessary to map the video memory into the virtual address space of the calling process.

10 In one embodiment, the virtual display driver and mirror display driver are derived from the architecture of a normal display driver and include a miniport driver and corresponding display driver. In conventional display drivers, there is a physical device, either attached to PCI bus or AGP slot. Video memory and registers are physically present on the video card, which are mapped in the address space of the GDI process or the capturing application using DirectDraw. In the present embodiment, however, there is no physical video memory. The operating system assumes the existence of a physical device (referred to as a virtual device) and its memory by allocating memory in the main memory, representing video memory and registers. When the miniport of the present invention is loaded, a chunk of memory, such as 2.5 MB, is reserved from the non-paged pool memory. This memory serves as video memory. This memory is then mapped in the virtual address space of the GDI process (application in case of a graphics draw operation). When the display driver of the present invention requests a pointer to the memory, the miniport returns a pointer to the video memory reserved in the RAM. It is therefore transparent to the GDI and display device interface (DDI) (or application in case of direct draw) whether the video memory is on a RAM or a video card. DDI or GDI perform the rendering on this memory location.

The miniport of the present invention also allocates a separate memory for overlays. Certain applications and video players like Power DVD, Win DVD etc uses overlay memory for video rendering.

In one conventional embodiment, rendering is performed by the DDI and GDI. GDI provides the generic device independent rendering operations while DDI performs the device specific operation. The display architecture layers GDI over DDI and provides a facility that DDI can delegate it's responsibilities to GDI. In an embodiment of the present invention, because there is no physical device, there are no device specific operations. Therefore, the display driver of the present invention delegates the rendering operations to GDI. DDI provides GDI with the video memory pointer and GDI perform the rendering based on the request received from the Win32 GDI process. Similarly, in the case where the present invention is compatible with DirectDraw, the rendering operations are delegated to the HEL (Hardware emulation layer) by DDI.

In one embodiment, the present invention comprises a mirror driver which, when loaded, attaches itself to a primary display driver. Therefore, all the rendering calls to the primary display driver are also routed to the mirror driver and whatever data is rendered on the video memory of the primary display driver is also rendered on the video memory of the mirror driver. In this manner, the mirror driver is used for computer display duplication.

In one embodiment, the present invention comprises a virtual driver which, when loaded, operates as an extended virtual driver. When the virtual driver is installed, it is shown as a secondary driver in the display properties of the computer and the user has the option on extend the display on to this display driver.

In one embodiment, the mirror driver and virtual driver support the following resolutions: 640 * 480, 800 * 600, 1024 * 768, and 1280 * 1024. For each of these resolutions, the drivers support 8, 16, 24, 32 bit color depths and 60 and 75 Hz refresh rates. Rendering on the overlay surface is done in YUV 420 format.

In one embodiment, a software library is used to support the capturing of a computer display using the mirror or virtual device drivers. The library maps the video memory allocated in the mirror and virtual device drivers in the application space when it is initialized. In the capture function, the library copies the mapped video buffer in the application buffer. In this manner, the application has a copy of the computer display at that particular instance.

For capturing the overlay surfaces, the library maps the video buffer in the application space. In addition, a pointer is also mapped in the application space which holds the address of the overlay surface that was last rendered. This pointer is updated in the driver. The library obtains a notification from the virtual display driver when rendering on the overlay memory starts. The display driver informs the capture library of the color key value. After copying the main video memory, a software module, CAPI, copies the last overlay surface rendered using the pointer which was mapped from the driver space. It does the YUV to RGB conversion and pastes the RGB data, after stretching to the required dimensions, on the rectangular area of the main video memory where the color key value is present. The color key value is a special value which is pasted on the main video memory by the GDI to represent the region on which the data rendered on the overlay should be copied. In use on computers operating current Windows/NT operating systems, overlays only apply to the extended virtual device driver and not the mirror

driver because, when the mirror driver is attached, DirectDraw is automatically disabled.

While the video and graphics capture method and system has been specifically described in relation to Microsoft operating systems, it should be appreciated that a similar mirror display driver and virtual display driver approach can be used with computers operating other operating systems.

In one embodiment, audio is captured using through an interface used by conventional computer-based audio players to play audio data. In one embodiment, audio is captured using Microsoft Windows Multimedia API, which is a software module compatible with Microsoft Windows and NT operating systems. A Microsoft Windows Multimedia Library provides an interface to the applications to play audio data on an audio device using waveOut calls. Similarly, it also provides interfaces to record audio data from an audio device. The source for recording device can be line In, microphone, or any other source designation. The applications can specify the format (sampling frequency, bits per sample) in which it wants to record the data. An exemplary set of steps for audio capture in a Windows/NT compatible operating system computing environment are as follows.

1. An application opens the audio device using waveInOpen() function. It specifies the audio format in which to record, the size of audio data to capture at a time and callback function to call when the specified size to audio data is available

2. The application passes a number of empty audio buffers to the windows audio subsystem using waveInAddBuffer() call.

3. To specify start of capture the application calls waveInStart()

4. When the specified size of audio data is available, the Windows audio subsystem calls the callback function through

which it passes the audio data to the application in one of the audio buffers which were passed by the application.

5 5. The application copies the audio data into its local buffer and, if it needs to continue capturing again, passes the empty audio buffer to the Windows audio subsystem through waveInAddBuffer()

6. When the application needs to stop capturing, the application calls waveInClose()

10 In one embodiment, a stereo mix option is selected in a media playback application and audio is captured in the process. Audio devices typically have the capability to route audio, being played on an output pin, back to an input pin. While named differently on different systems, it is generally referred to as a "stereo mix". If the stereo mix option is selected in the playback option, and audio is recorded from the default audio device using waveIn call, then everything that is being played on the system can be recorded. i.e the audio being played on the system can be captured. It should be appreciated that the specific approach is dependent on the capabilities of the particular audio device being used and that one of ordinary skill in the art would know how to capture the audio stream in accordance with the above teaching. It should also be appreciated that, to prevent the concurrent playback of audio from the computer and the remote device, the local audio (on the computer) should be muted, provided that such muting does not also mute the audio routing to the input pin.

15 20 25

30 In another embodiment, a virtual audio driver, referred to as a virtual audio cable (VAC), is installed as a normal audio driver that can be selected as a default playback and/or recording device. A feature of VAC is that, by default, it routes all the audio going to its audio output pin to its input pin. Therefore, if VAC is selected as a default playback device,

then all the audio being played on the system would go to the output pin of VAC and hence to its input pin. If any application captures audio from the input pin of VAC using the appropriate interface, such as the waveIn API, then it would be able to capture everything that is being played on that system. In order to capture audio using VAC, it would have to be selected as a default audio device. Once VAC is selected as a default audio device, then the audio on the local speaker would not be heard. Where

The media is then transmitted 505 simultaneously in a synchronization manner wirelessly to a receiver, as previously described. The receiver, which is in data communication with the remote monitoring device receives 506 the compressed media data. The media data is then uncompressed 507 using the CODEC. The data is then finally rendered 508 on the display device.

To transmit the media, any transmission protocol may be employed. However, it is preferred to transmit separate video and audio data streams, in accordance with a hybrid TCP/UDP protocol, that are synchronized using a clock or counter. Specifically, a clock or counter sequences forward to provide a reference against which each data stream is timed.

Referring to Figure 6, an embodiment of the abovementioned TCP/UDP hybrid protocol is depicted. The TCP/UDP hybrid protocol 600 comprises of a TCP packet header 601 of size equivalent to 20 TCP, 20 IP and a physical layer header and UDP packet header 602 of size equivalent to 8 TCP, 20 IP and a physical layer header.

Figure 7 is a flow diagram that depicts the functional steps of the TCP/UDP real-time (RT) transmission protocol implemented in the present invention. The transmitter and receiver, as previously described, establish 701 connection using TCP and the transmitter sends 702 all the reference frames

using TCP. Thereafter, the transmitter uses 703 the same TCP port, which was used to establish connection in step 701, to send rest of the real-time packets but switches 704 to the UDP as transport protocol. While transmitting real-time packets
5 using UDP, the transmitter further checks for the presence of an RT packet that is overdue for transmission. The transmitter discards 705 the overdue frame at the transmitter itself between IP and MAC. However an overdue reference frame/packet is always sent. Thus, the TCP/UDP protocol significantly reduces
10 collisions while substantially improving the performance of RT traffic and network throughput.

The TCP/UDP protocol is additionally adapted to use ACK spoofing as a congestion-signaling method for RT transmission over wireless networks. Sending RT traffic over wireless
15 networks can be sluggish. One of the reasons for this is that after transmission of every block of data TCP conventionally requires the reception of an ACK signal from the destination/receiver before resuming the transmission of the next block or frame of data. In IP networks, specifically
20 wireless, there remain high probabilities of the ACK signals getting lost due to network congestion, particularly so in RT traffic. Thus, since TCP does both flow control and congestion control, this congestion control causes breakage of connection over wireless networks owing to scenarios such as non-receipt of
25 ACK signals from the receiver.

To manage breakage of connection, the present invention, in one embodiment, uses ACK spoofing for RT traffic sent over networks. By implementing ACK spoofing, if the receiver does not receive any ACK within a certain period of time, the transmitter
30 generates a false ACK for the TCP, so that it resumes sending process. In an alternate embodiment, in the event of poor quality of transmission due to congestion and reduced network

throughput, the connection between the transmitter and receiver is broken and a new TCP connection is opened to the same receiver. This results in clearing congestion problems associated with the previous connection. It should be appreciated that this transmission method is just one of several transmission methods that could be used and is intended to describe an exemplary operation.

Referring to Figure 8, the block diagram depicts the components of the CODEC of the integrated wireless system. The CODEC 800 comprise a motion estimation block 801 which removes the temporal redundancy from the streaming content, a DCT block 802 which converts the frame into 8*8 blocks of pixels to perform DCT, a VLC coding circuit 803 which further codes the content into shorter words, an IDCT block 804 converts back the spatial frequencies to the pixel domain, and a rate control mechanism 805 for speeding up the transmission of media.

The motion estimation block 801 is used to compress the video by exploiting the temporal redundancy between the adjacent frames of the video. The algorithm used in the motion estimation is preferably a full search algorithm, where each block of the reference frame is compared with the current frame to obtain the best matching block. The full search algorithm, as the term suggests, takes every point of a search region as a checking point, and compares all pixels between the blocks corresponding to all checking points of the reference frame and the block of the current frame. Then the best checking point is determined to obtain a motion vector value.

For example, Figure 9 depicts the functional steps of the one embodiment of the motion estimation block. The checking points A and A1 shown in the figure respectively correspond to the blocks 902 and 904 in a reference frame. If the checking point A is moved left and downward by one pixel, it becomes the

checking point A1. In this way, when the block 902 is shifted left and downward by one pixel, it results in the block 904.

The comparison technique is performed by computing the difference in the image information of all corresponding pixels and then summing the absolute values of the differences in the image information. Finally, the sum of absolute difference (SAD) is performed. Then, among all checking points, the checking point with the lowest SAD is determined to be the best checking point. The block that corresponds to the best checking point is the block of the reference frame, which matches best with the block of the current frame that is to be encoded. And these two blocks obtain a motion vector.

Referring back to Figure 8, once the motion estimation is carried out the picture is coded using a discrete cosine transform (DCT) via the DCT block 802. The DCT coding scheme transforms pixels (or error terms) into a set of coefficients corresponding to the amplitudes of specific cosine basis functions. The discrete cosine transform (DCT) is typically regarded as the most effective transform coding technique for video compression and is applied to the sampled data, such as digital image data, rather than to a continuous waveform.

Usage of the DCT for image compression is advantageous because the transform converts N (point) highly correlated input spatial vectors in the form of rows and columns of pixels into N point DCT coefficient vectors including rows and columns of DCT coefficients in which high frequency coefficients are typically zero-valued. Energy of a spatial vector, which is defined by the squared values of each element of the vector, is preserved by the DCT transform so that all energy of a typical, low-frequency and highly-correlated spatial image is compacted into the lowest frequency DCT coefficients. Furthermore, the human psycho visual system is less sensitive to high frequency signals so that a

reduction in precision in the expression of high frequency DCT coefficients results in a minimal reduction in perceived image quality. In one embodiment 8*8 block resulting from the DCT block is divided by a quantizing matrix to reduce the magnitude of the DCT coefficients. In such a case, the information associated to the highest frequencies less visible to human sight tends to be removed. The result is reordered and sent to the variable length-coding block 803.

Variable length coding (VLC) block 803 is a statistical coding block that assigns codewords to the values to be encoded. Values of high frequency of occurrence are assigned short codewords, and those of infrequent occurrence are assigned long codewords. On an average, the more frequent shorter codewords dominate so that the code string is shorter than the original data. VLC coding, which generates a code made up of DCT coefficient value levels and run lengths of the number of pixels between nonzero DCT coefficients, generates a highly compressed code when the number of zero-valued DCT coefficients is greatest. The data obtained from the VLC coding block is transferred to the transmitter at an appropriate bit rate. The amount of data transferred per second is known as bit rate.

Figure 10 depicts the exemplary digital signal waveform and data transfer. The vertical axis 1001 represents voltage and the horizontal axis 1002 represents time. The digital waveform has a pulse width of N and a period (or cycle) of $2N$ where N represents the bit time of the pulse (i.e., the time during which information is transferred). The pulse width, N , may be in any units of time such as nanoseconds, microseconds, picoseconds, etc. The maximum data rate that may be transmitted in this manner is $1/N$ transfers per second, or one bit of data per half cycle (the quantity of time labeled N). The fundamental frequency of the digital waveform is $1/2N$ hertz. In one

embodiment, simplified rate control is employed which increases the bit rate of the data by 50% compared to MPEG2 using the method described above. Consequently in less time there is large chunk of data being transferred to the transmitter making the process real time.

The compressed data is then transmitted, in accordance with the above-described transmission protocol, and wirelessly received by the receiver. To provide motion video capability, compressed video information must be quickly and efficiently decoded. The aspect of the decoding process, which is used in the preferred embodiment, is inverse discrete cosine transformation (IDCT). Inverse discrete cosine transform (IDCT) converts the transform-domain data back to spatial-domain form. A commonly used two-dimensional data block size is 8*8 pixels, which furnishes a good compromise between coding efficiency and hardware complexity. The inverse DCT circuit performs an inverse digital cosine transform on the decoded video signal on a block-by-block basis to provide a decompressed video signal.

Referring to Figure 11, a diagram of the processing and selective optimization of the IDCT block is depicted. The circuit 1100 includes a preprocess DCT coefficient block (hereinafter PDCT) 1101, an evaluate coefficients block 1102, a select IDCT block 1103, a compute IDCT block 1104, a monitor frame rate block 1105 and an adjust IDCT parameters block 1106. In operation, the wirelessly transmitted media, received from the transmitter, includes various coded DCT coefficients, which are routed to the PDCT block 1101. The PDCT block 1101 selectively sets various DCT coefficients to a zero value to increase processing speed of the inverse discrete cosine transform procedure with a slight reduction or no reduction in video quality. The DCT coefficient-evaluating block 1102 then receives the preprocessed DCT coefficient from the PDCT 1101.

The evaluating circuit 1102 examines the coefficients in a DCT coefficient block before computation of the inverse discrete cosine transform operation. Based on the number of non-zero coefficients, an inverse discrete cosine transform (IDCT) selection circuit 1103 selects an optimal IDCT procedure for processing of the coefficients. The computation of the coefficients is done by the compute IDCT block 1104. In one embodiment, several inverse discrete cosine transform (IDCT) engines are available for selective activation by the selection circuit 1103. Typically, the inverse discrete cosine transformed coefficients are combined with other data prior to display. The monitor frame rate block 1105 thereafter determines an appropriate frame rate of the video system, for example by reading a system clock register (not shown) and comparing the elapsed time with a prestored frame interval corresponding to a desired frame rate. The adjust IDCT parameter block 1106 then adjusts parameters including the non-zero coefficient threshold, frequency and magnitude according to the desired or fitting frame rate.

The abovementioned IDCT block computes an inverse discrete cosine transform in accordance with the appropriate selected IDCT method. For example, an 8*8 forward discrete cosine transform (DCT) is defined by the following equation:

$$X(u, v) = C(u)C(v) \sum_{i=0}^7 \sum_{j=0}^7 x(i, j) \cos\left(\frac{x(i) + 1/2}{16}\right) \cos\left(\frac{x(j) + 1/2}{16}\right)$$

where $x(i, j)$ is a pixel value in an 8*8 image block in spatial domains i and j , and $X(u, v)$ is a transformed coefficient in an 8*8 transform block in transform domains u, v . $C(0)$ is $1/\sqrt{2}$ and $C(u) = C(v) = 1$.

An inverse discrete cosine transform (IDCT) is defined by the following equation:

$$f(u, v) = (1/4) \sum_{n=0}^7 \sum_{m=0}^7 C(n)C(m)X(n, m) \cos\left(\frac{\pi(2n+1)u}{16}\right) \cos\left(\frac{\pi(2m+1)v}{16}\right).$$

An 8*8 IDCT is considered to be a combination of a set of 64 orthogonal DCT basis matrices, one basis matrix for each two-dimensional frequency (v, u). Furthermore, each basis matrix is considered to be the two-dimensional IDCT transform of each single transform coefficient set to one. Since there are 64 transform coefficients in an 8*8 IDCT, there are 64 basis matrices. The IDCT kernel K(v, u), also called a DCT basis matrix, represents a transform coefficient at frequency (v, u) according to the equation:

$$K(v, u) = .nu.(u) .nu.(v) \cos\left(\frac{(2m+1) \cdot \pi \cdot u}{16}\right) \cos\left(\frac{(2n+1) \cdot \pi \cdot v}{16}\right),$$

where .nu.(u) and .nu.(v) are normalization coefficients defined as .nu.(u)=1/.sqrt(8) for u=0 and .nu.(u)=1/2 for u>0. The IDCT is computed by scaling each kernel by the transform coefficient at that location and summing the scaled kernels. The spatial domain matrix S is obtained using the equation, as follows

$$S = \sum_v \sum_u F_{v,u} K(v, u).$$

It should be appreciated that a 4*4 transform block could be used as well.

As previously discussed, while the various media streams may be multiplexed and transmitted in a single stream, it is preferred to transmit the media data streams separately in a synchronized manner. Referring to Figure 12, the block diagram depicts the components of the synchronization circuit for synchronizing media data of the integrated wireless system. The synchronization circuit 1200 comprises a buffer 1201 having the video and audio media, first socket 1202 for transmitting video and second socket 1203 for transmitting audio, first counter 1204 and second counter 1205 at the transmitter 1206 and first receiver 1207 for video data, second receiver 1208 for audio data, first counter 1209, second counter 1210, mixer 1211 and a buffer 1212 at receiver end 1213.

Operationally, the buffered audio and video data 1201 at the transmitter 1206 after compression is transmitted separately on the first socket 1202 and the second socket 1203. The counters 1204, 1205 add an identical sequence number both to the video and audio data prior to transmission. In one embodiment, the audio data is preferably routed via User Datagram Protocol (UDP) whereas the video data via Transmission Controlled Protocol (TCP). At the receiver end 1213, the UDP protocol and the TCP protocol implemented by the audio receiver block 1208 and the video receiver block 1207 receives the audio and video signals. The counters 1209, 1210 determine the sequence number from the audio and video signals and provide it to the mixer 1211 to enable the accurate mixing of signals. The mixed data is buffered 1212 and then rendered by the remote monitor.

Referring to Figure 13, the flowchart depicts another embodiment of synchronizing audio and video signals of the integrated wireless system of the present invention. Initially, the receiver receives 1301 a stream of encoded video data and encoded audio data wirelessly. The receiver then ascertains 1302

the time required to process the video portion and the audio portion of the encoded stream. After that, the receiver determines 1303 the difference in time to process the video portion of the encoded stream as compared to the audio portion of the encoded stream. The receiver subsequently establishes 1304 which processing time is greater (i.e., the video processing time or the audio processing time).

If the audio processing time is greater, the video presentation is delayed 1305 by the difference determined, thereby synchronizing the decoded video data with the decoded audio data. However, if the video processing time is greater, the audio presentation is not delayed and played at its constant rate 1306. Video presentation tries to catch up the audio presentation by discarding video frames after regular intervals. The data is then finally rendered 1307 on the remote monitor. Therefore, audio "leads" video meaning that the video synchronizes itself with the audio.

In a particular embodiment, the decoded video data is substantially synchronized with the decoded audio data. Substantially synchronized means, that while there may be a slight, theoretically measurable difference between the presentation of the video data and the presentation of the corresponding audio data, such a small difference in the presentation of the audio and video data is not likely to be perceived by a user watching and listening to the presented video and audio data.

A typical transport stream is received at a substantially constant rate. In this situation, the delay that is applied to the video presentation or the audio presentation is not likely to change frequently. Thus, the aforementioned procedure may be performed periodically (e.g., every few seconds or every 30 received video frames) to be sure that the delay currently being

applied to the video presentation or the audio presentation is still within a particular threshold (e.g., not visually or audibly perceptible). Alternatively, the procedure may be performed for each new frame of video data received from the transport stream.

Referring to Figure 14, another embodiment of the audio and video synchronizaton circuit is depicted. The synchronization circuit 1400 at the transmitter end 1401 comprises buffer 1402 having media data, multiplexer 1403 for combining the media data signals, such as graphics, text, audio, and video signals, and a clock 1404 for providing the timestamps to the media content for synchronization. At the receiver end 1405 the demultiplexer 1406 using clock 1407 devolves the data stream into the individual media data streams. The timestamps provided by the clocks help synchronize the audio and video at the receiver end. The clock is set at the same frequency as that of receiver. The audio and video, which is demultiplexed, is routed to the speakers 1408 and display device 1409 for rendering.

In one embodiment, the present invention provides a system and method of automatically downloading, installing, and updating the novel software of the present invention on the computing device or remote monitor. No software CD is required to install software programs on the remote monitor, the receiver in the remote monitor, the computing device, or the transmitter in the computing device. As an example, a personal computer communicating to a wireless projector is provided, although the description is generic and will apply to any combination of computing device and remote monitor. It is assumed that both the personal computer and wireless projector are in data communication with a processing system on chip, as previously described.

On start up, the wireless projector (WP-AP) runs a script to configure itself as an access point. The WP-AP sets the SSID as QWPxxxxxx where 'xxxxxx' is lower 6 bytes of AP's MAC Address. The WP-AP sets its IP Address as 10.0.0.1. WP-AP
5 starts an HTTP server. WP-AP starts the DHCP server, with following settings in the configuration file

Start Address: 10.0.0.3

End Address: 10.0.0.254

DNS: 10.0.0.1

10 Default Gateway: 10.0.0.1

[Second and Third Octet of the Addresses are configurable]

The WP-AP starts a small DNS server, configured to reply
10.0.0.1 (i.e. WP-AP's address) for any DNS query. The IP
Address in the response will be changed if the WP-AP's IP
15 Address is changed. The default page of HTTP server has a small
software program, such as a Java Applet, that conducts the
automatic software update. The error pages of the HTTP server
redirect to the default page, making sure that the default page
is always accessed upon any kind of HTTP request. This may
20 happen if the default page on the browser has some directory
specified as well, e.g.
<http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&msnhome>

The WP-AP, through its system on chip and transceiver,
communicates its presence as an access point. The user's
25 computing device has a transceiver capable of wirelessly
transmitting and receiving information in accordance with known
wireless transmission protocols and standards. The user's
computing device recognizes the presence of the wireless
projector, as an access point, and the user instructs the
30 computing device to join the access point through graphical user
interfaces that are well known to persons of ordinary skill in
the art.

After joining the wireless projector's access point, the user opens a web browser application on the computing device and types into a dialog box and any URL, or permits the browser to revert to a default URL. The opening of the web browser
5 accesses the default page of WP-AP HTTP server and results in the initiation of the software program (e.g. Java Applet).

In one embodiment, the software program checks if the user's browser supports it in order to conduct an automatic software update. The rest of the example will be described in
10 relation to Java but it should be appreciated that any software programming language could be used.

If Java is supported by the browser, the applet will check if the software and drivers necessary to implement the media transmission methods described herein are already installed. If
15 already present, then the Java Applet compares the versions and automatically initiates installation if the computing device software versions are older than the versions on the remote monitor.

If Java is not supported by the browser, the user's web
20 page is redirected to an installation executable, prompting the user to save it or run it. The page will also display instructions of how to save and run the installation. The installation program also checks if the user has already installed the software and whether the version needs to be
25 upgraded or not. In this case user will be advised to Install Java.

In a first embodiment, the start address for WP-AP's DNS server is 10.0.0.2. WP-AP runs the DHCP client for its Ethernet connection and obtains IP, Gateway, Subnet and DNS addresses
30 from the DHCP Server on the local area network. If the DHCP is disabled then it uses static values. The installation program installs the application, uninstaller, and drivers. The

application is launched automatically. On connection, the application obtains the DNS address of WP-AP's Ethernet port, and sets it on the local machine. After the connection is established, WP-AP enables IP Forwarding and sets the firewall
5 such that it only forwards packets from the connected application to the Ethernet and vice versa. These settings enable the user to access the Ethernet local area network of WP-AP and access the Internet. The firewall makes sure that only the user with his/her application connected to the WP-AP can
10 access LAN/Ethernet. On disconnection, WP-AP disables IP Forwarding and restores the firewall settings. The application running on the user system sets the DNS setting to 10.0.0.1. On the application exit, the DNS setting is set to DHCP.

In another embodiment, during installation, the user is
15 prompted to select if the computing device will act as a gateway or not. Depending on the response, the appropriate drivers, software, and scripts are installed.

Referring now to Figure 15, another exemplary configuration for automatically downloading and updating the software of the
20 present invention is shown. The wireless projector access point has a pre-assigned IP Address of 10.A.B.1 and the gateway system has as a pre-assigned IP Address of 10.A.B.2 where A and B octets can be changed by the user.

The WP-AP is booted. The user's computing device scans for
25 available wireless networks and selects QWPxxxxxx. The computing device's wireless configuration should have automatic TCP/IP configuration enabled, i.e. 'Obtain an IP address automatically' and 'Obtain DNS server address automatically' options should be checked. The computing device will automatically get an IP
30 address from 10.0.0.3 to 10.0.0.254. The default gateway and DNS will be set as 10.0.0.1.

The user opens the browser, and, if Java is supported, the automatic software update begins. If Java is not supported, the user will be prompted to save the installation and will have to run it manually. If the computing device will not act as a gateway to a network, such as the Internet, during the installation, the user selects 'No' to the Gateway option.

The installation runs a script to set the DNS as 10.0.0.2. So that next DNS query gets appropriately directed. An application link is created on the desktop. The user runs the application that starts transmitting the exact contents of the user's screen to the Projector. If required, user can now change the WP-AP configuration (SSID, Channel, IP Address Settings: second and third octet can be changed of 10.0.0.x.).

If the computing device will act as a gateway to a network, such as the Internet, during the installation, the user selects 'Yes' to the Gateway option when prompted. The installation then enables Internet sharing (IP Forwarding) on the Ethernet interface (sharing is an option in the properties of network interface in both Windows 2000 and Windows XP), sets the system's wireless interface IP as 10.0.0.2, sets the system's wireless interface netmask as 255.255.255.0, and sets the system's wireless interface gateway as 10.0.0.1. An application link is created on the desktop. The user runs the application that starts transmitting the exact contents of the user's screen to the Projector. If required, user can now change the WP-AP configuration (SSID, Channel, IP Address Settings: second and third octet can be changed of 10.0.0.x.).

It should be appreciated that the present invention enables the real-time transmission of media from a computing device to one or more remote monitoring devices or other computing devices. Referring to Figure 16, another arrangement of the integrated wireless multimedia system of the present invention

is depicted. In this particular embodiment the communication between the transmitter 1601 and the plurality of receivers 1602, 1603, 1604 is depicted. The transmitter 1601 wirelessly transmits the media to a receiver integrated into, or in data communication with, multiple devices 1601, 1602, and 1603 for real-time rendering. In an alternate embodiment the abovementioned software can also be used in both the mirror capture mode and the extended mode. In mirror capture mode, the real time streaming of the content takes place with the identical content being displayed both at the transmitter and the receiver end. However, in an extended mode, user can work on some other application at the transmitter side and the transmission can continue as a backend process.

The above examples are merely illustrative of the many applications of the system of present invention. Although only a few embodiments of the present invention have been described herein, it should be understood that the present invention might be embodied in many other specific forms without departing from the spirit or scope of the invention. For example, other configurations of transmitter, network and receiver could be used while staying within the scope and intent of the present invention. Therefore, the present examples and embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope of the appended claims.

CLAIMS

We claim:

- 5 1. A method of capturing media from a source and wirelessly transmitting said media, comprising the steps of:
 playing said media, comprising at least audio data and
 video data, on a computing device;
 capturing said video data using a mirror display
10 driver;
 capturing said audio data from an input source;
 compressing said captured audio and video data; and
 transmitting said compressed audio and video data
 using a transmitter.
- 15 2. The method of claim 1 further comprising the step of receiving said media at a receiver, decompressing said captured audio and video data, and playing said decompressed audio and video data on a display remote from
20 said source.
3. A method of claim 2 wherein said transmitter and receiver establish a connection using TCP and the transmitter transmits packets of video data using UDP.
- 25 4. The method of claim 1 wherein said media further comprises graphics and text data and wherein said graphics and text data is captured together with said video data using the mirror display driver.
- 30 5. The method of claim 1 further comprising the step of processing said video data using a CODEC.

6. The method of claim 1 wherein said CODEC removes temporal redundancy from the video data using a motion estimation block.

5

7. The method of claim 1 wherein said CODEC converts a frame of video data into 8*8 blocks of pixels using a DCT block.

10

8. The method of claim 1 wherein said CODEC codes video content into shorter words using a VLC coding circuit.

9. ~~The method of claim 1 wherein said CODEC converts back~~ spatial frequencies of the video data into the pixel domain using an IDCT block.

15

10. The method of claim 1 wherein said CODEC comprises a rate control mechanism for speeding up the transmission of media.

20

11. A program stored on a computer-readable substrate for capturing media, comprising at least video data, from a source and wirelessly transmitting said media, comprising:
a mirror display driver operating in a kernel mode for capturing said video data;
a CODEC for processing said video data; and
a transmitter for transmitting said processed video data.

25

12. The program of claim 1 further comprising a virtual display driver.

30

13. The program of claim 1 wherein said transmitter establishes a connection with a receiver using TCP and the transmitter transmits packets of video data using UDP.

5 14. The program of claim 1 wherein said media further comprises graphics and text data and said mirror display driver captures graphics and text data together with said video data.

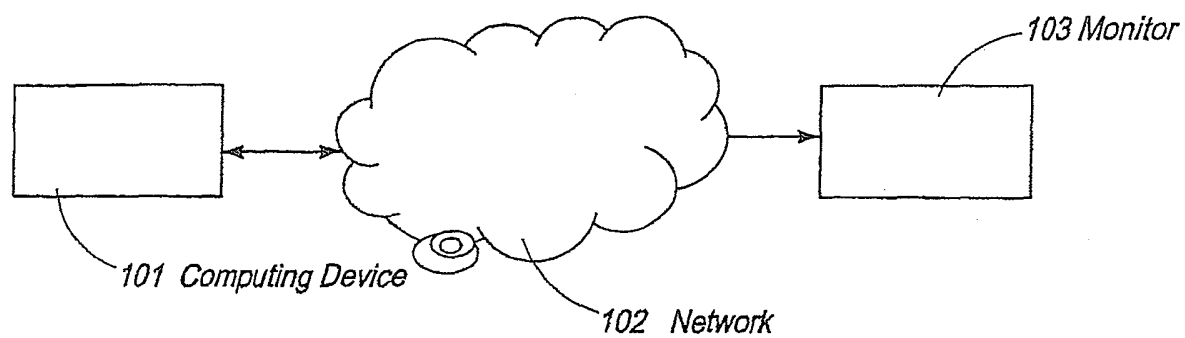
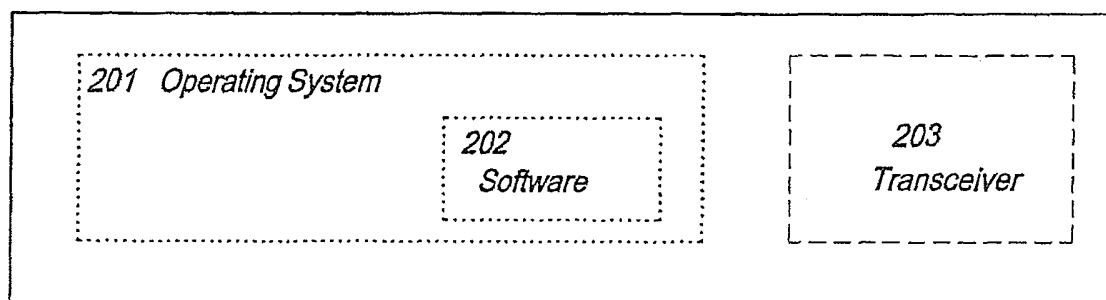
10 15. The program of claim 1 wherein said CODEC comprises a motion estimation block for removing temporal redundancy from the video data.

15 16. The program of claim 1 wherein said CODEC comprises a DCT block for converting a frame of video data into 8*8 blocks of pixels.

20 17. The program of claim 1 wherein said CODEC comprises a VLC coding circuit for coding video content into shorter words.

25 18. The program of claim 1 wherein said CODEC comprises an IDCT block for converting back spatial frequencies of the video data into the pixel domain.

30 19. The program of claim 1 wherein said CODEC comprises a rate control mechanism for speeding up the transmission of media.

*FIG. 1**200 Computing Device**FIG. 2*

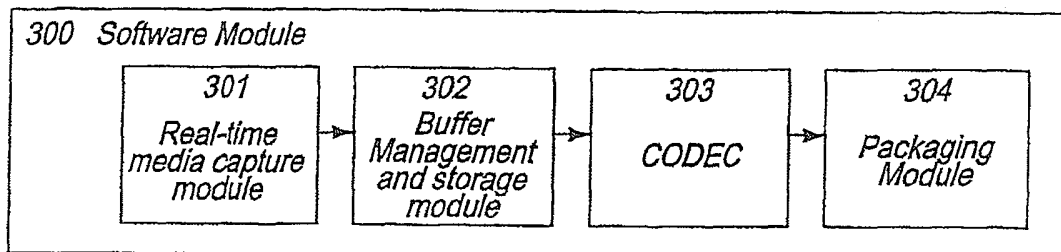


FIG. 3

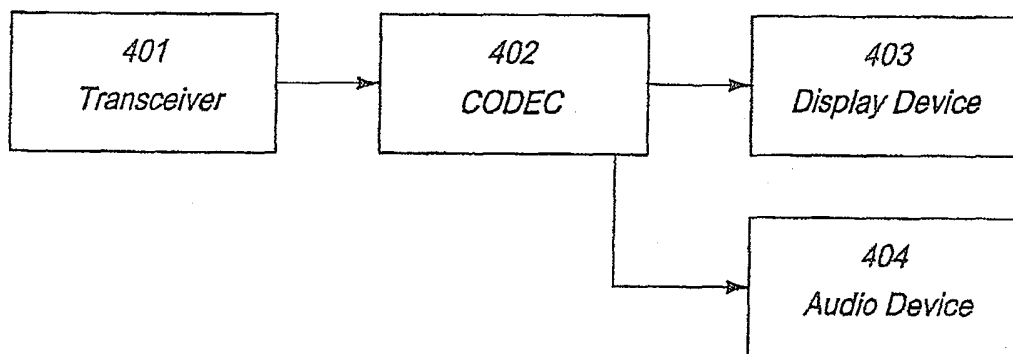
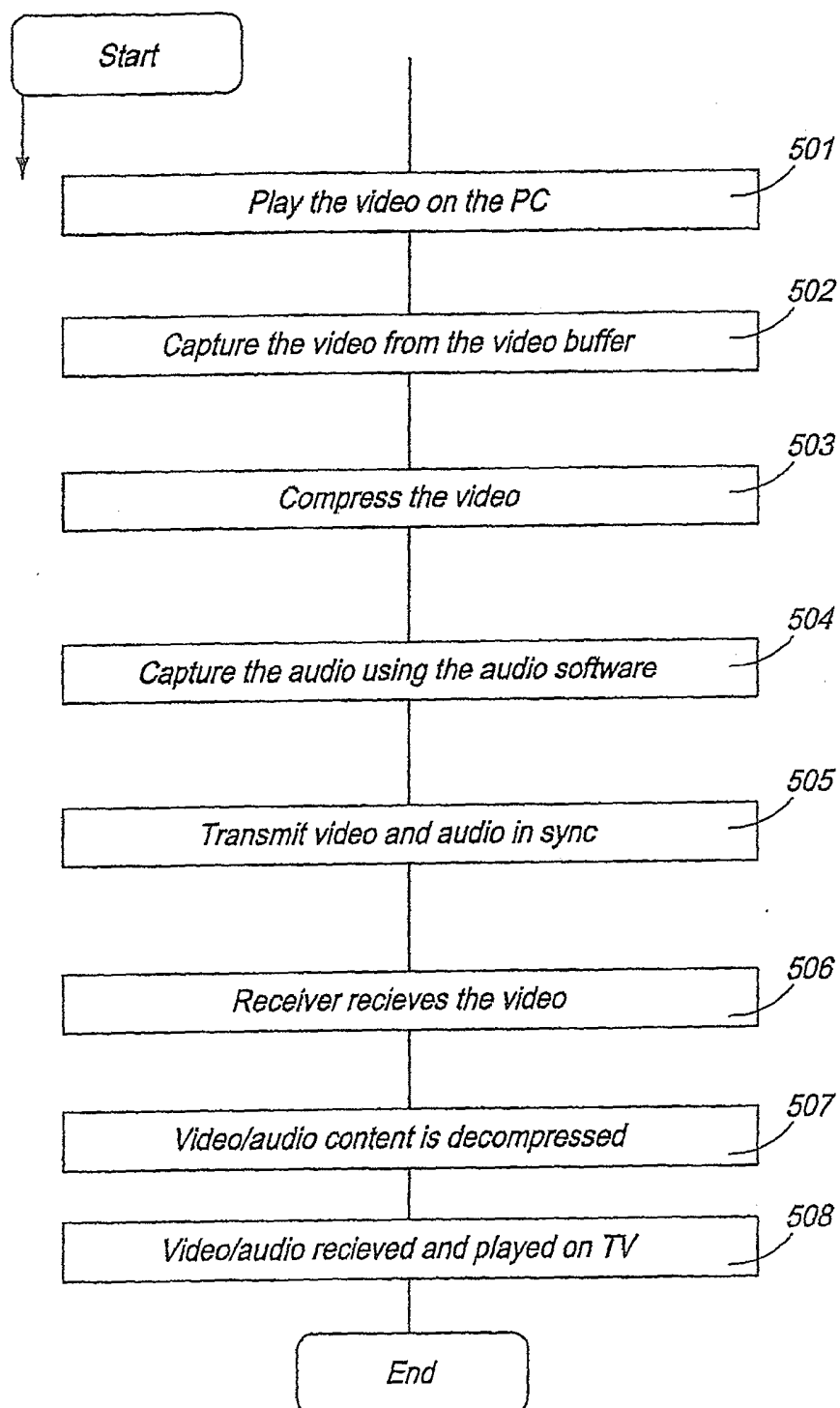
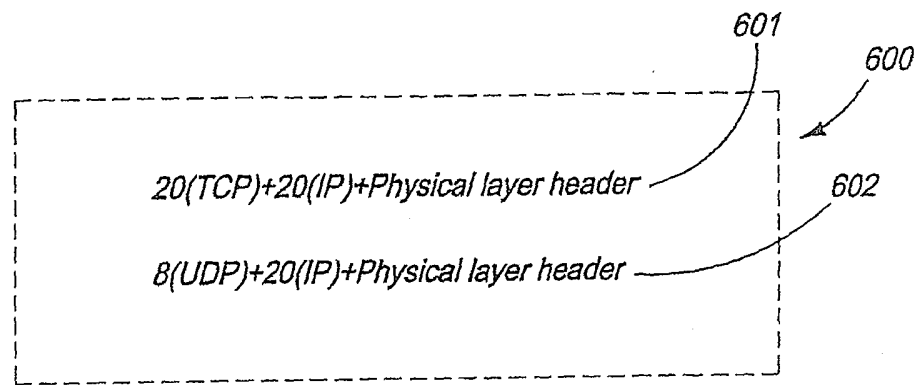
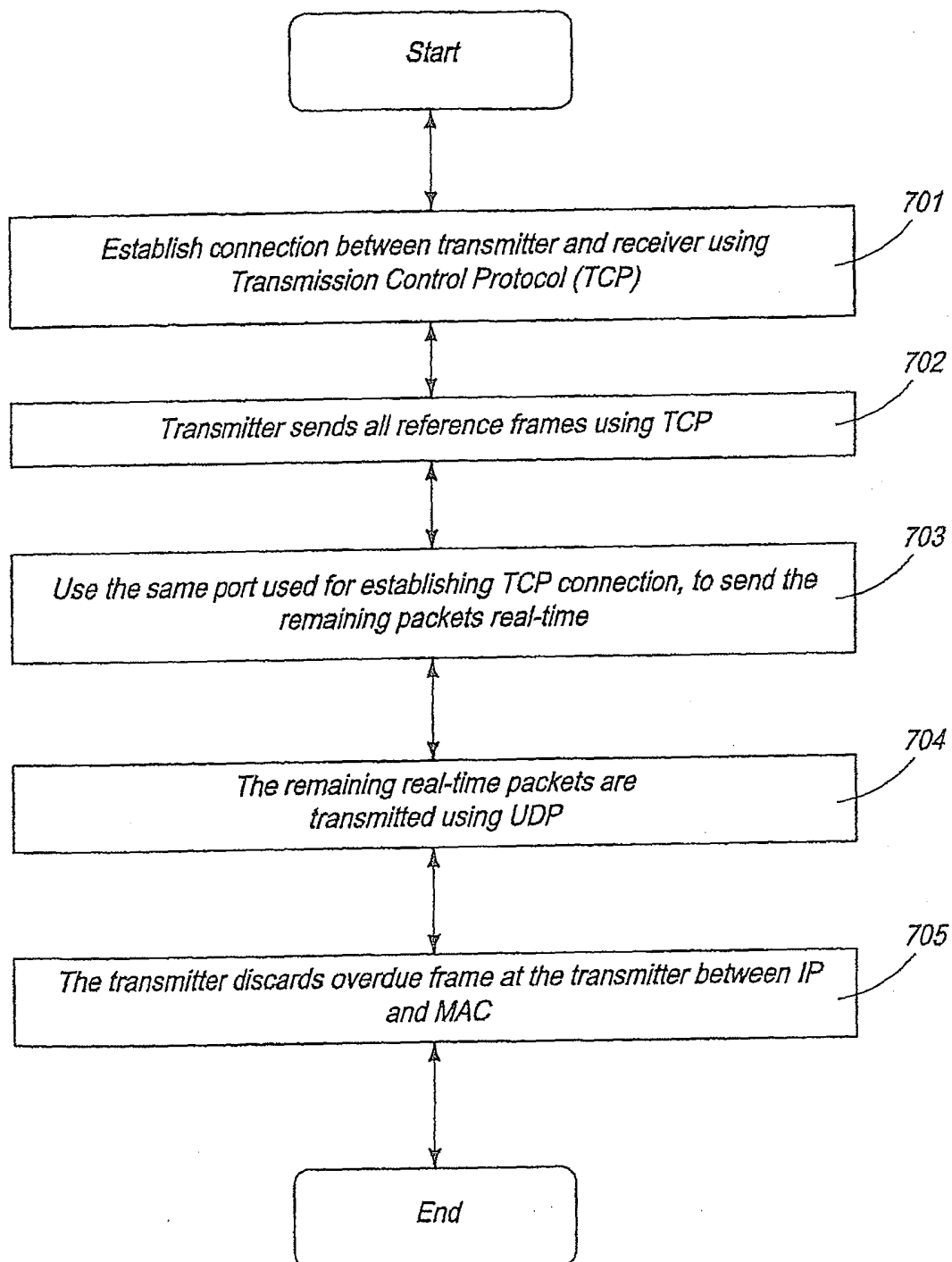
400 Receiver

FIG. 4

**FIG. 5**

**FIG. 6**

**FIG. 7**

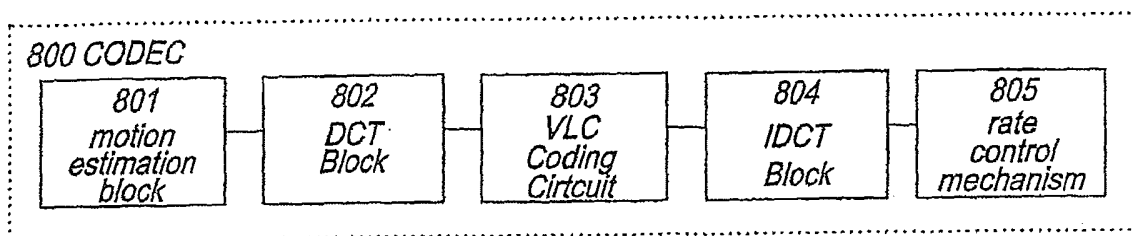


FIG. 8

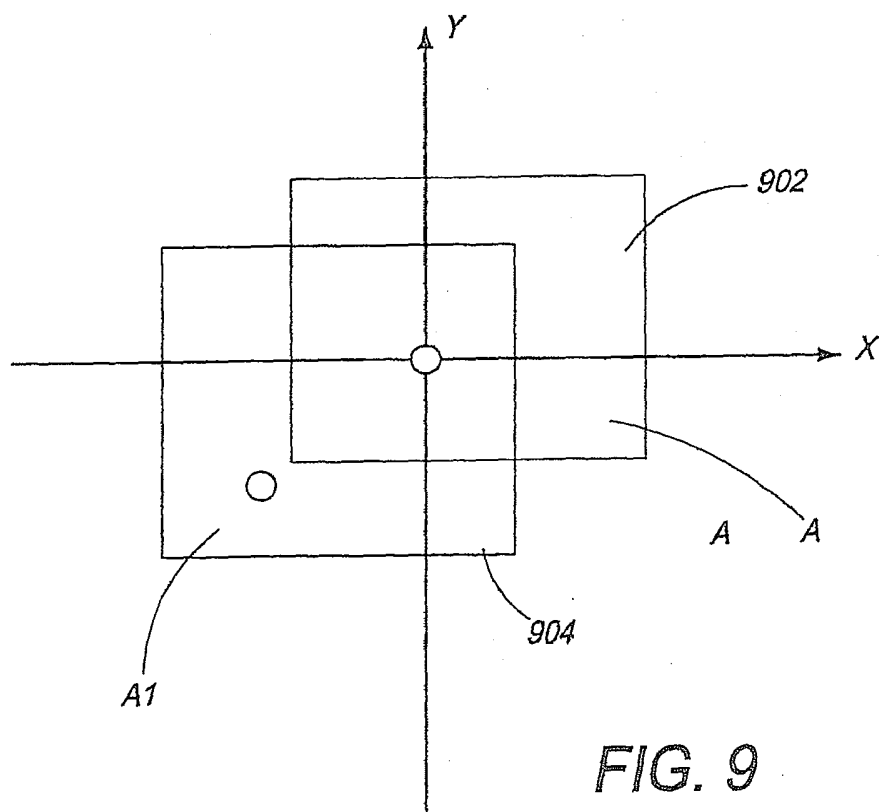


FIG. 9

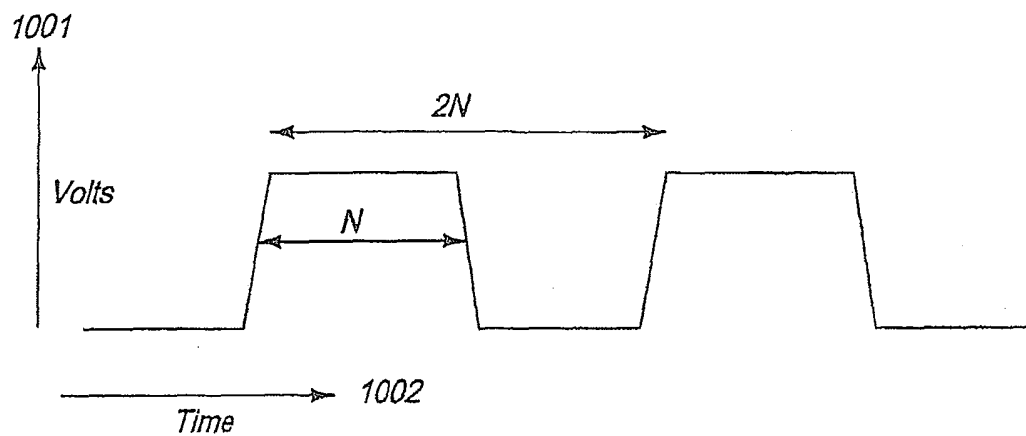


FIG. 10

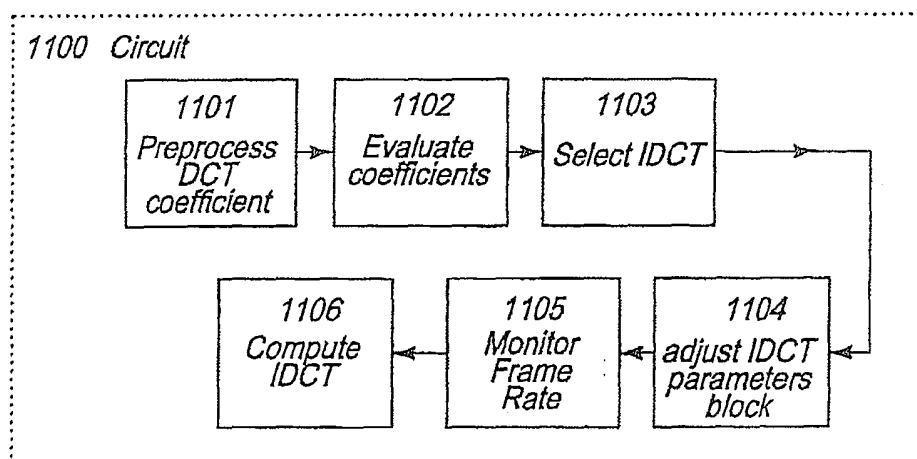
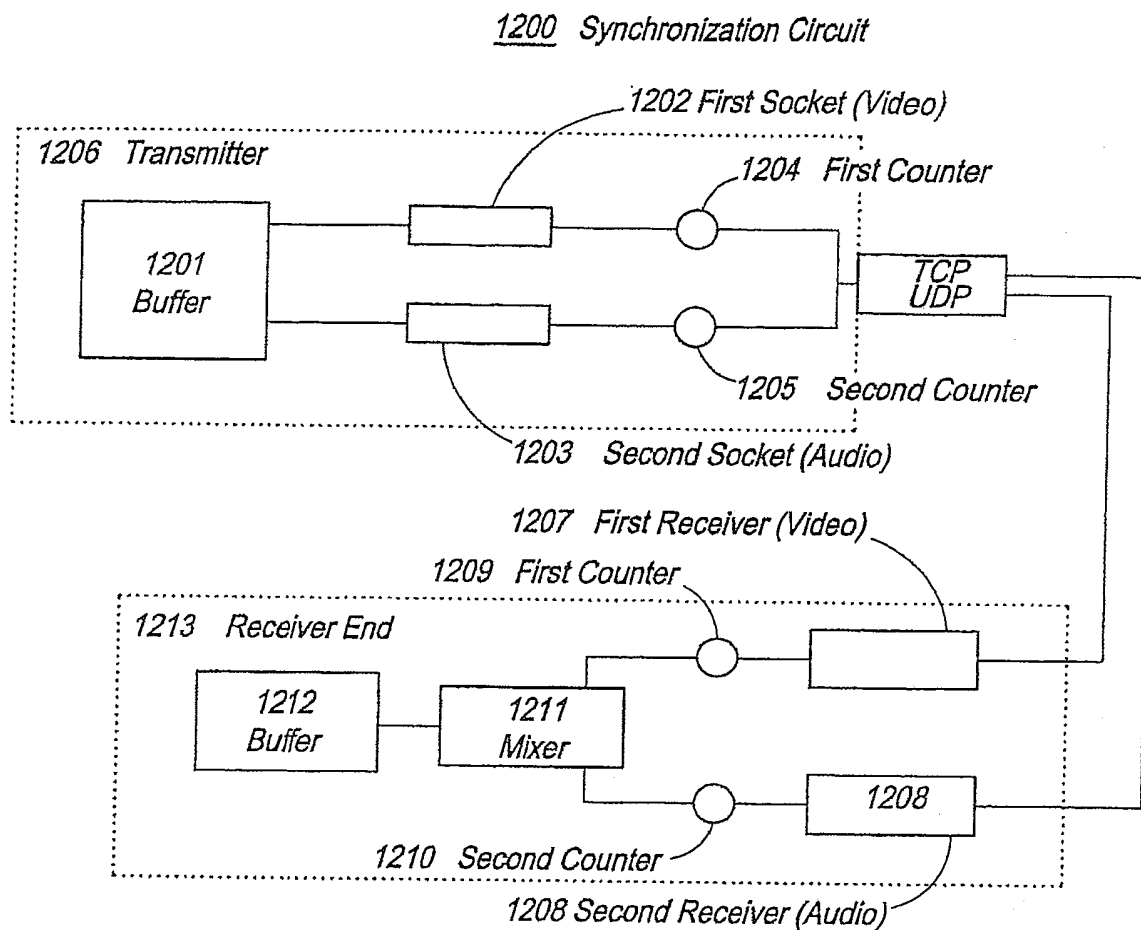


FIG. 11

**FIG. 12**

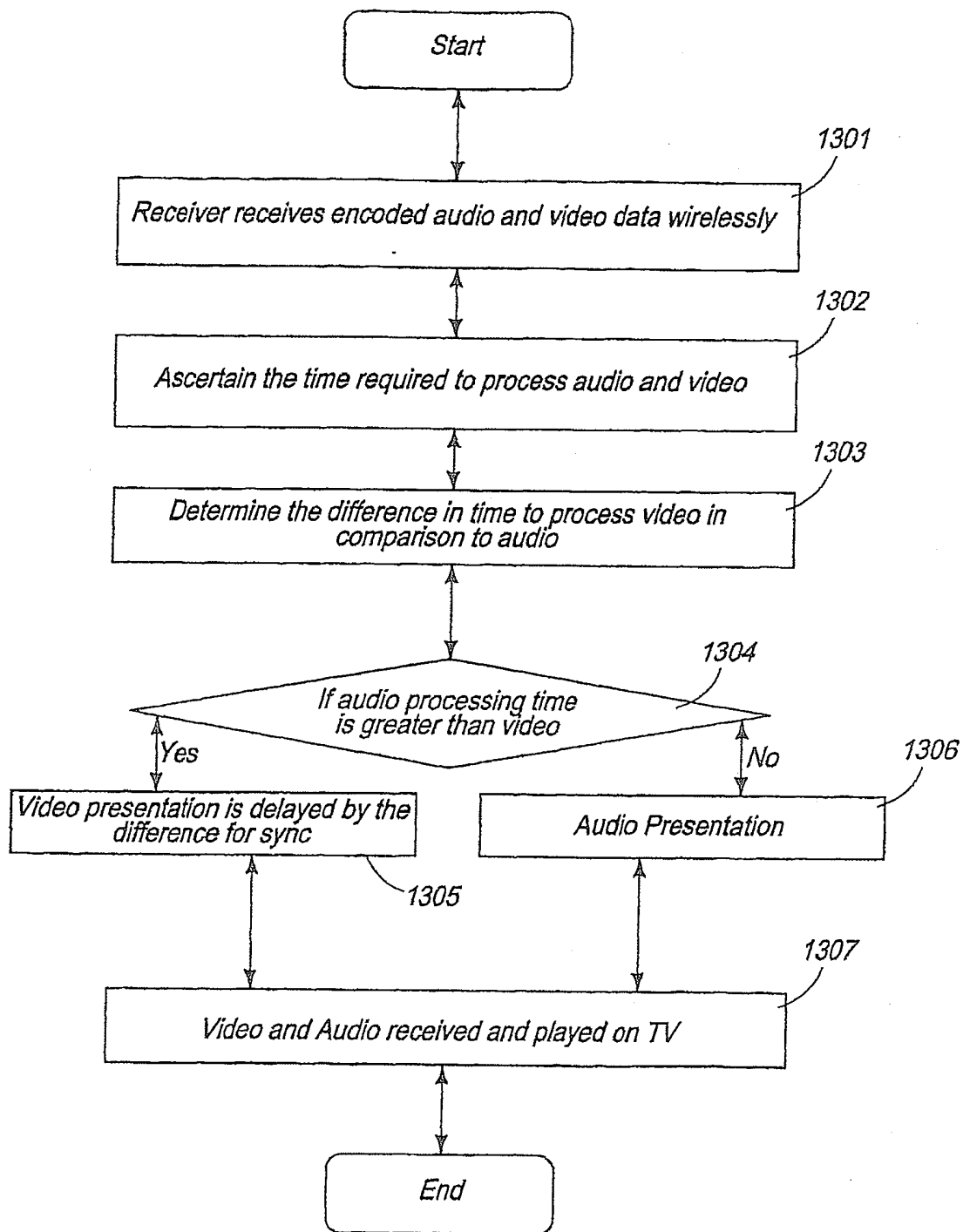
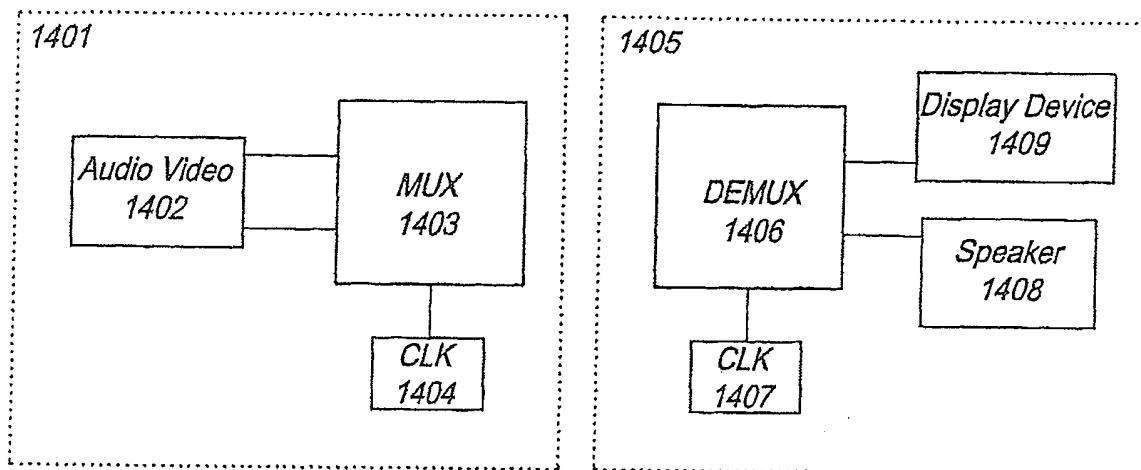


FIG. 13

1400**FIG. 14**

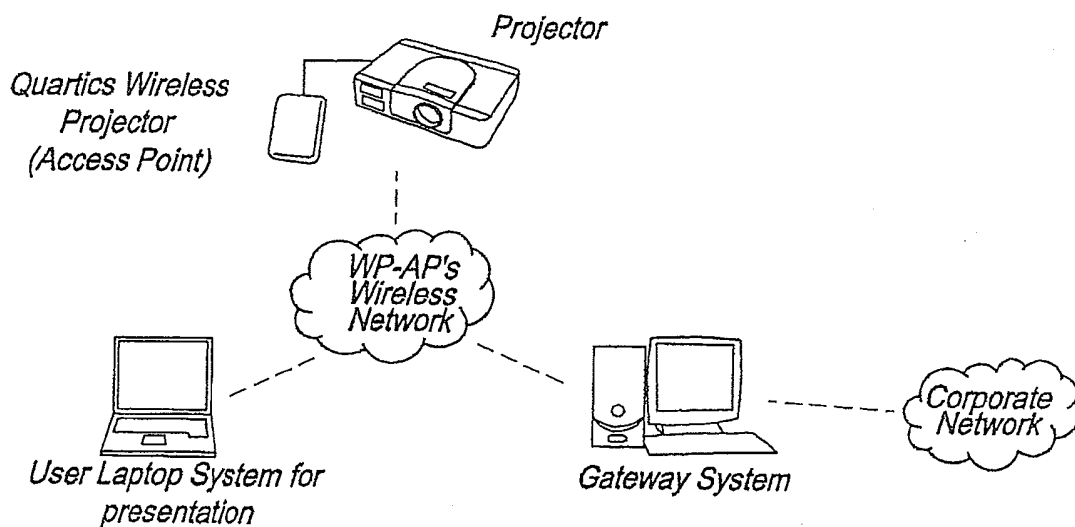


FIG. 15

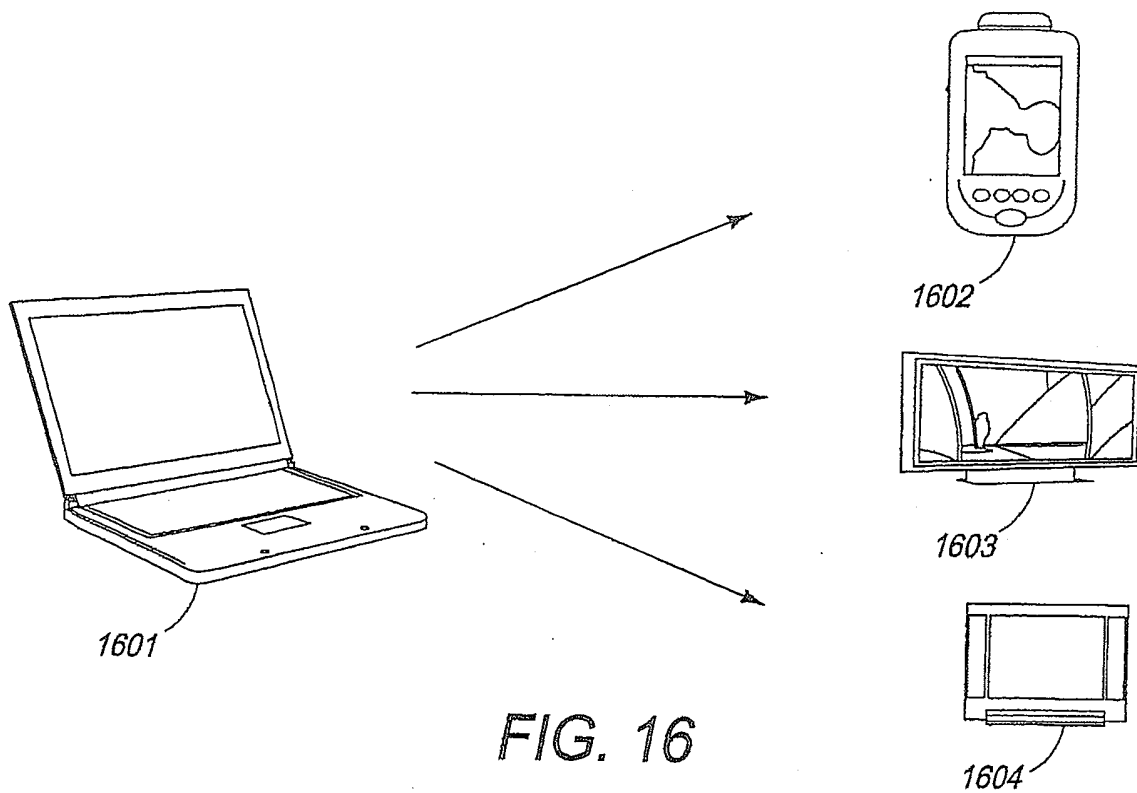
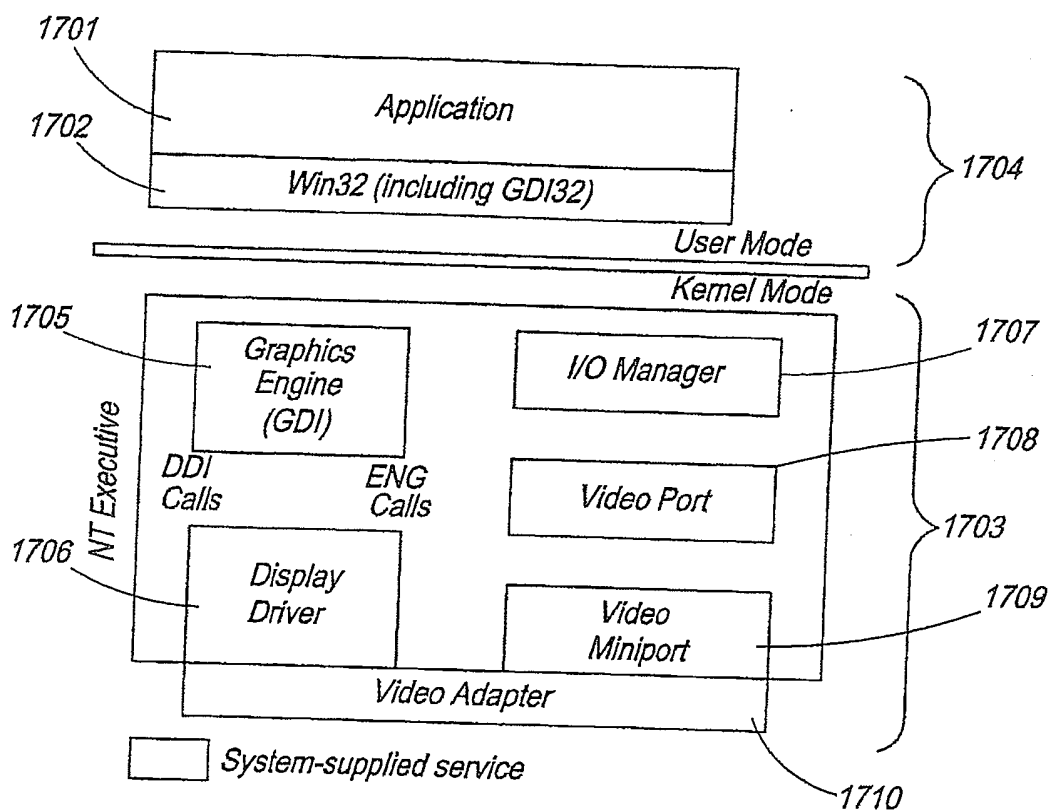


FIG. 16

1700**FIG. 17**

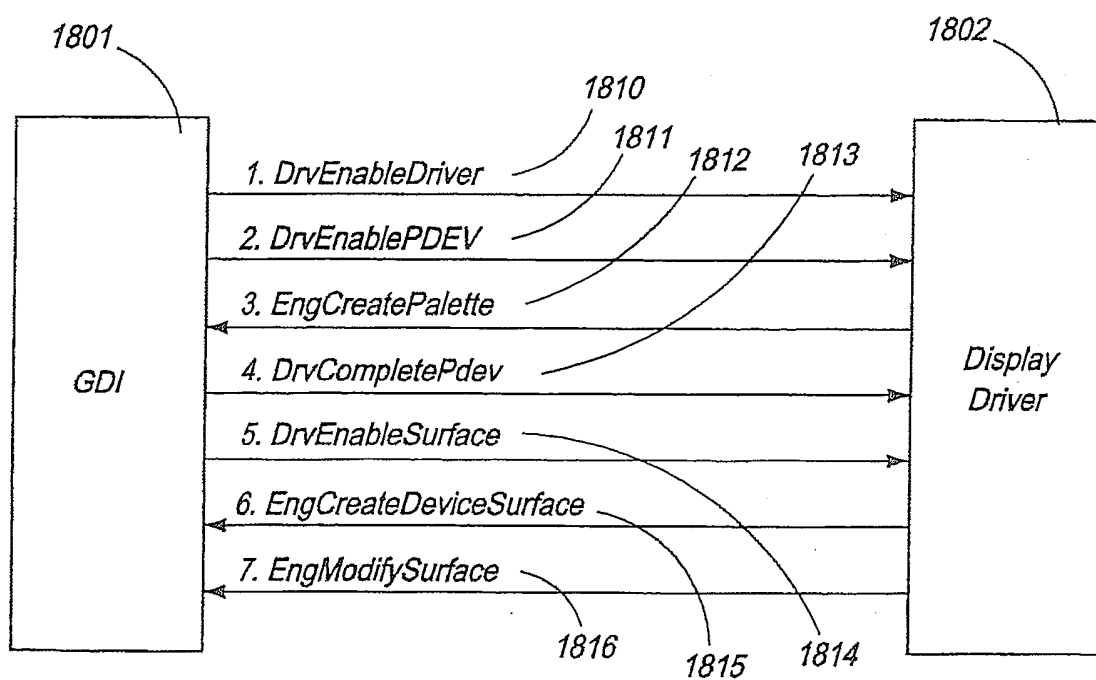


FIG. 18

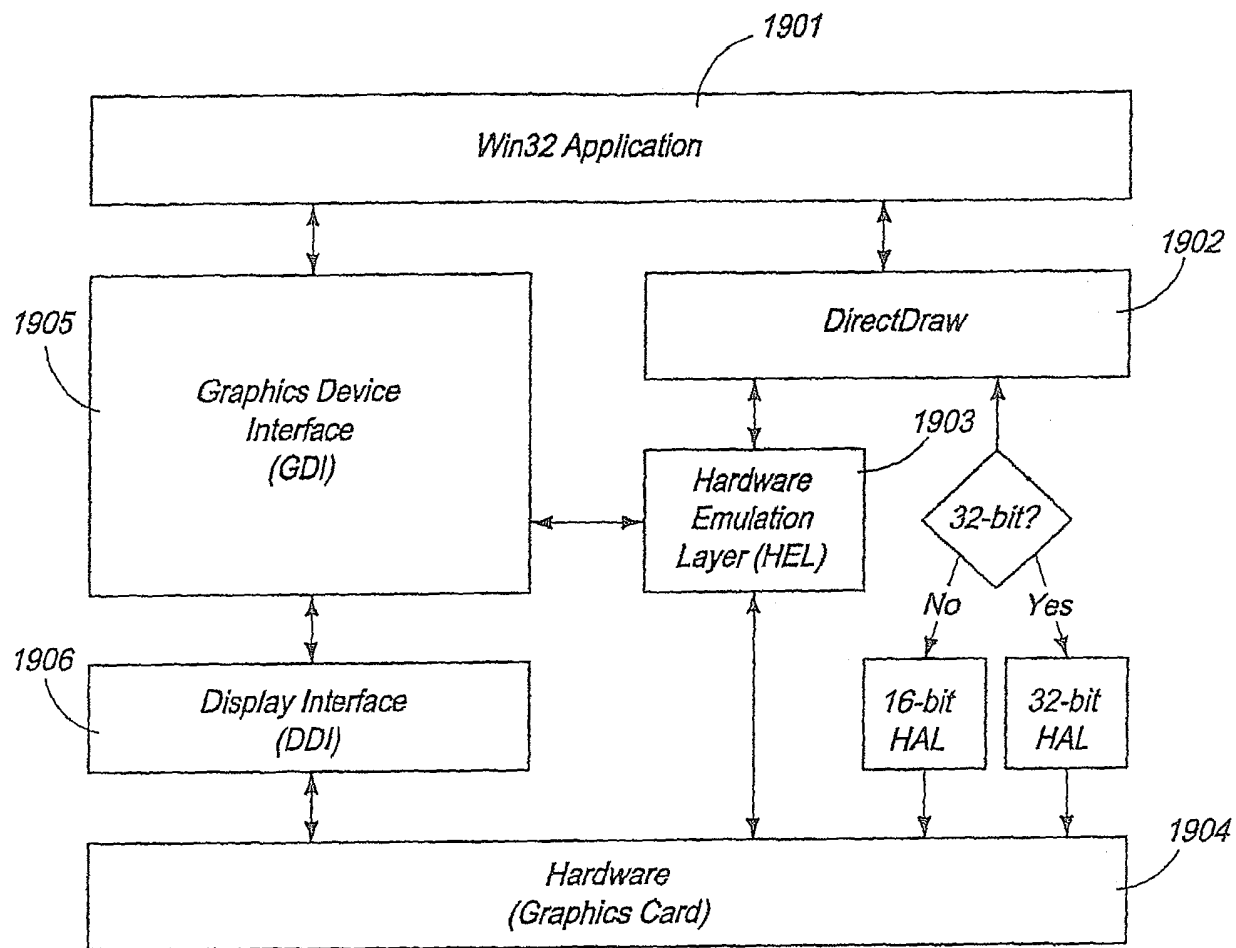


FIG. 19