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

(19) **World Intellectual Property Organization**
International Bureau

(43) **International Publication Date**
**17 June 2004 (17.06.2004)**

**PCT**

(10) **International Publication Number**
**WO 2004/051962 A1**

(51) **International Patent Classification⁷:** **H04L 29/06**, 29/08, G06F 17/60, A63F 13/12, H04N 7/24, 7/173

(21) **International Application Number:** PCT/US2003/035821

(22) **International Filing Date:** 7 November 2003 (07.11.2003)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**
10/306,536 27 November 2002 (27.11.2002) US

(71) **Applicant: HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 20555 S.H. 249, Houston, TX 77070 (US).

(72) **Inventors: YONG, Yan**; 34077 Paseo Padre Parkway #69, Fremont, CA 94555 (US). **SHEN, Bo**; 655 Mariposa Avenue #5, Mountain View, CA 94041 (US). **BASU, Sujoy**; 2326 California Street #8, Mountain View, CA 94040 (US). **KUMAR, Rajendra**; 2019 Kent Drive, Los Altos, CA 94024 (US).

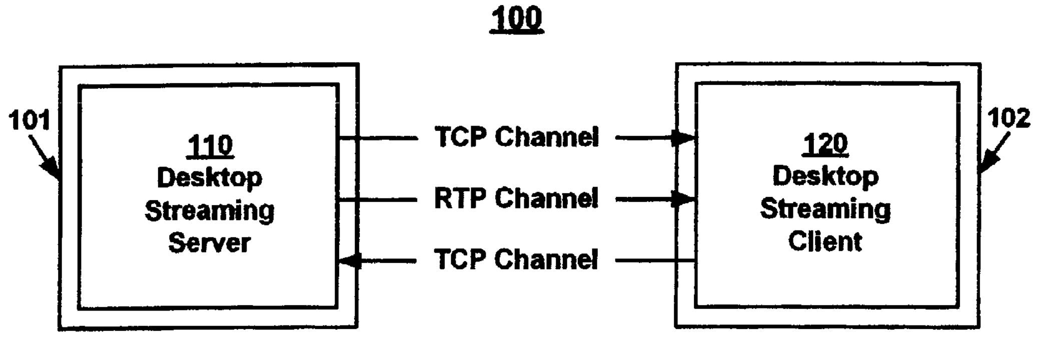
(74) **Agent: LEE, Denise, A.**; Hewlett-Packard Company, Intellectual Property Administration, P O Box 272400, Mail Stop 35, Fort Collins, CO 80527-2400 (US).

(81) **Designated States** *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, 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, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(54) **Title:** METHODS AND SYSTEMS FOR TRANSFERRING EVENTS INCLUDING MULTIMEDIA DATA



(57) **Abstract:** Methods (700) and systems (110, 120) for transferring an event from a server to a remote client are described. An event is received from a driver (710). The event is dispatched into an event queue according to the event type (720). The event is processed according to the event type (730). The processing includes encoding the event when the event comprises multimedia data. The event is transferred to the remote client when triggered (740). The transfer occurs according to a protocol corresponding to the event type.

WO 2004/051962 A1

# METHODS AND SYSTEMS FOR TRANSFERRING EVENTS INCLUDING MULTIMEDIA DATA

## TECHNICAL FIELD

Embodiments of the present invention relate to computer system networks. More specifically, embodiments of the present invention relate to multimedia streaming in a network environment.

## BACKGROUND ART

Computer system networks provide access to local resources (e.g., software applications, data, Web pages, etc., on a server) from a remote client device. Applications such as virtual network computing have extended this idea further. In a virtual computing network, servers supply not only software applications and data but also a desktop environment that can be accessed and controlled from client devices. In essence, a user at a client device is presented with a display (a graphical user interface or desktop) that is generated at a server and then transferred to and reproduced at the client. Accordingly, the amount of state maintained by the client can be reduced, resulting in what is referred to as "thin" client. Also, a single desktop can be simultaneously accessed and viewed from several different clients. This may be particularly useful when implementing a group of cooperating engineering work stations.

Prior art virtual network computing systems typically use protocols such as the Remote Frame Buffer (RFB) protocol to provide client access to the server-generated graphical user interface. The image to be displayed on a computer system monitor is held in a frame buffer on the server. In a limiting case, the entire contents of the frame buffer are periodically transferred to the client device for display. Enhancements to the limiting case include sending only the changes to the frame buffer (and hence to the display) that occurred since the preceding transfer.

Typically, the rate of transfer to the client is about 10-15 frames per second. While this rate is fairly high, it is not considered high enough to satisfactorily transfer video data or multimedia data (video and audio data) from the server to the client. Consequently, prior art systems are limited in this regard.

Accordingly, a more satisfactory way of transferring video/multimedia data from servers to clients in a virtual network computing system is desirable. Embodiments of the present invention provide such an improvement.

## DISCLOSURE OF THE INVENTION

Embodiments of the present invention pertain to methods and systems for transferring an event from a server to a remote client. An event is received from a driver. The event is dispatched into an event queue according to the event type. The event is processed according to the event type. The processing includes encoding the event when the event comprises multimedia data. The event is transferred to the remote client when triggered. The transfer occurs according to a protocol corresponding to the event type.

## BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

FIGURE 1 is a block diagram of a server and a client coupled in a network according to one embodiment of the present invention.

FIGURE 2 is a block diagram of a desktop streaming server in accordance with one embodiment of the present invention.

FIGURE 3 is a block diagram of a driver interface in accordance with one embodiment of the present invention.

FIGURE 4 is a block diagram showing differentiation of window updates according to one embodiment of the present invention.

FIGURE 5 is a block diagram of a desktop streaming engine according to one embodiment of the present invention.

FIGURE 6 is a block diagram of a desktop streaming client according to one embodiment of the present invention.

FIGURE 7 is a flowchart of a method for transferring an event to a remote client according to one embodiment of the present invention.

The drawings referred to in this description should not be understood as being drawn to scale except if specifically noted.

## BEST MODE FOR CARRYING OUT THE INVENTION

Reference will now be made in detail to various embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with these embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. In other instances, well-known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Figure 1 is a block diagram of a server 101 and a client 102 coupled in a network 100 according to one embodiment of the present invention. It is appreciated that network 100 may include elements other than those shown. Network 100 may also include more than one of the various elements shown; for example, there may be multiple clients coupled to server 101, and there may be multiple servers. The functionality of server 101 and client 102 is discussed below (refer at least to Figures 2 and 6); however, it is appreciated that these elements may implement functionality other than that discussed.

Communication may occur directly between server 101 and client 102 of Figure 1, or indirectly through an intermediary device or node (not shown). Also, communication may be wired or wireless, or a combination of wired and wireless. In one embodiment, communication occurs over the World Wide Web (or Internet).

In one embodiment, server 101 and client 102 are computer systems that provide processing capability and memory capacity. In addition, according to the embodiments of the present invention, server 101 implements a desktop streaming server 110, and client 102 implements a desktop streaming client 120. In this embodiment, desktop streaming server 110 and desktop streaming client 120 allow the server 101 to supply a display (e.g., a graphical user interface, and in particular, a "desktop" display) to the client 102. As will be seen, both server-side control and client-side control are allowed, either concurrently or exclusively.

According to the embodiments of the present invention, "events" are transferred from server 101 to client 102, and vice versa, using desktop streaming server 110 and desktop streaming client 120, respectively. As used herein, an event is generally an output of a software driver, such as a keyboard driver, a mouse driver, a video driver or an audio driver. Thus, events include detectable actions or occurrences such as clicking a mouse button or pressing a key on a keyboard. Events can also refer to actions or occurrences that can cause a change in the display (graphical user interface) to be displayed by the client 102. For example, a change in the display caused by scrolling through a text (word processing) document, or by changing from one page to another page in the document, can constitute an event. A movement of a window in the display can also constitute an event. A moving video display can also constitute an event. An audio sample can also constitute an event.

To summarize, as used herein, an event is an output of a driver used with either server 101 or client 102; an event may cause a change in the display on either server 101 or client 102, and/or may cause an audio output at the client 102. Events can be generally categorized as: control events, and data events. Control events carry control information to change the execution behaviors of other components, locally or remotely. Data events carry content data to inform other components of content changes.

For simplicity of discussion, the term "mouse" is used herein; however, it is understood that other means of cursor control may be used instead of, or in combination with, a mouse. There is a myriad of cursor control mechanisms known in the art, and each of these may be used in accordance with the present invention. The term "keyboard" is also used herein. Again, there is a myriad of mechanisms known in the art that provide the functionality of a keyboard, including voice recognition systems, and each of these may be used in accordance with the present invention. Thus, although embodiments of the present invention are described for a mouse and keyboard, the present invention is not so limited. Instead, the focus should be on the term "event" and how that term is used herein, as described above. In that context, keyboard and mouse events are simply examples of events, while the embodiments of the present invention are well suited to use with virtually all types of events.

In one embodiment, on the server side, the events received from the drivers are dispatched into event queues according to the event type. The events are then processed according to their event type; for example, a multimedia event may be encoded. The events are then transferred from server 101 to client 102 according to a protocol that corresponds to the event type. In one embodiment, events from keyboard and mouse drivers are transferred to client 102 using a protocol such as Transmission Control Protocol (TCP), while events from video and audio drivers are transferred to client 102 using a protocol such as Real Time Protocol (RTP). These processes are described in further detail in conjunction with Figure 2, below.

TCP and RTP are known in the art. TCP provides a highly reliable protocol, while RTP provides the timing capability useful for synchronizing the video and audio portions of a multimedia data object. Thus, TCP can be used to precisely transfer control events (e.g., keyboard and mouse events) as well as other high precision events, such as changes in graphics applications, and RTP can be used for streaming frame buffer (video buffer) and sound buffer updates. Other protocols that can provide these or similar capabilities may be used instead.

On the client side, in one embodiment, events are received from keyboard and mouse drivers and transferred to server 101 of Figure 1 using a protocol such as TCP. This process is described further in conjunction with Figure 6, below. In the present embodiment, the relationship between server 101 and client 102 does not have the client sending events from audio and video drivers to the server; however, the present invention is not so limited.

Figure 2 is a block diagram of a desktop streaming server 110 in accordance with one embodiment of the present invention. Desktop streaming server 110 is implemented in server 101 of Figure 1 as hardware, software, firmware, or a combination thereof. As illustrated by Figure 2, desktop streaming server 110 includes a number of elements that are rendered separately for clarity of illustration and discussion; however, it is understood that these elements may not exist as separate entities within desktop streaming server 110. In general, according to the various embodiments of the present invention, desktop streaming server 110 provides the capability and functionality provided by the various elements of Figure 2. In addition, desktop streaming server 110 may provide other capabilities and functionalities in addition to those described herein.

In the present embodiment, a driver interface 210 receives input (events) from a number of drivers such as, but not limited to, keyboard driver 201, mouse driver 202, video driver 203, and/or audio driver 204. Thus, in addition to intercepting more conventional events such as window changes, keyboard input and cursor movement, driver interface 210 also samples video data from the video driver 203 and audio data from the audio driver 204.

Generally, the drivers 201-204 are components of the underlying operating system. For example, the keyboard driver for a Windows environment may be different than the keyboard driver for a Linux environment. According to the present embodiment, driver interface 210 provides the capability to communicate with the various types of drivers while providing a fixed interface to the downstream portions of desktop streaming server 110 (e.g., a fixed interface to event filter 220).

Figure 3 is a block diagram of a driver interface 210 in accordance with one embodiment of the present invention. In this embodiment, driver interface 210 includes a number of interface modules, exemplified by interfaces 301 and 302, so that there is a module suitable for interfacing with each driver. This is not to say that there is an interface module per driver, although that may be the case.

In the present embodiment, driver interface 210 also includes an application program interface (API) 305 that is common to the interfaces 301 and 302, etc. The API 305 provides a single interface to the remainder of desktop streaming server 110 despite the myriad of possible drivers, and thus insulates the desktop streaming server 110 from the underlying operating system.

Returning to Figure 2, in the present embodiment, the events from driver interface 210 are provided to (received by) event filter 220. In this embodiment, event filter 220 identifies the event according to event type, and dispatches the event into a respective event queue according to event type. The event queues are used to buffer similar types of events that can be transferred to the desktop streaming client 120 (Figure 1) using a protocol appropriate to the event type (e.g., TCP or RTP) after processing (if any) appropriate to the event type.

In the present embodiment, the event filter 220 of Figure 2 categorizes events into the following types: 1) control events that are generated by keyboard input or cursor (e.g., mouse) movement, for example; 2) window movement events that are generated by window movements in which window content is unchanged; 3) window updates that are generated by a change in window content; and 4) audio samples that are captured from a change in the sound buffer of the audio driver 204. In this embodiment, control events are forwarded to the control event queue 231; window movements are forwarded to the window movement queue 232; window updates are forwarded to the windows update queue 233; and audio samples are forwarded to the audio sample queue 234.

In the present embodiment, control events in the control event queue 231 record the coordinate changes of the mouse or cursor. For example, for a particular cursor movement, only the start and end coordinates of the movement may be transferred to the desktop streaming client 120 of Figure 1. Control events can be further optimized by the desktop streaming engine 250.

Continuing with reference to Figure 2, in the present embodiment, the window movement queue 232 records the coordinate changes of a window that is being moved (without the content of the window being changed). In those cases in which the window content has been previously transferred to desktop streaming client 120 of Figure 1, only the changes in the coordinates of the window may be transferred to the client. The client can then render the desktop display using the previously received window content and the new window coordinates.

According to the present embodiment, the window updates queue 233 of Figure 2 records the updates of window content. These updates can be captured from the video frame buffer. In one embodiment, in order to facilitate selection of an appropriate encoder and transfer protocol, window updates are categorized into three types: 1) graphics window updates; 2) video window updates; and 3) other window updates. Here, graphics is used to refer to relatively static images (e.g., images having relatively low update rates in comparison to video); video refers to moving images having a relatively high update rate; and other window updates refers to relatively static window content such as the text in a word processing document. Generally speaking, the window updates can be separated into categories according to the complexity of the image and the update frequency.

Figure 4 is a block diagram showing differentiation of window updates according to one embodiment of the present invention. In this embodiment, window updates differentiator 410 filters the windows into the aforementioned categories. The window updates differentiator 410 can accomplish this using information such as the display application type, and the coordinates and screen sizes of the windows. The display application type can be acquired from the driver interface 210 (Figure 2) when, for example, a new window is created, a Uniform Resource Locator (URL) for a Web site is clicked, or an application is executed. The coordinates and screen sizes help to locate which part of an update to the video frame buffer belongs to which window.

Graphics window updates are forwarded to the graphics window updates queue 421; video window updates are forwarded to the video window updates queue 422; and other window updates are forwarded to the other window updates queue 423.

Returning to Figure 2, in the present embodiment, the transfer of events from each of the queues 231-234 to the desktop streaming engine 250 is triggered by a respective regulator 241, 242, 243 and 244. Similarly, in the embodiment of Figure 4, the queues 421-423 are under control of a respective regulator 431, 432 and 433.

In one embodiment, the regulators 241-244 of Figure 2 (and the regulators 431-433 of Figure 4) trigger (impel) events from their respective queues based on a threshold being reached or a time interval having passed. The threshold is based on the number of events in a respective event queue. When the number of events in the queue reaches a predefined threshold, some portion of the events or all of the events can be transferred from the queue. The threshold mechanism provides the capability to handle a burst of events, in order to prevent the queue from accumulating too many events before the events are transferred in response to a time-out signal. The threshold may be different for each of the queues 231-234 and 421-423. The threshold may also be changed by the desktop streaming engine 250 based on monitoring of the channels to the desktop streaming client 120 (Figure 1), or based on feedback from the desktop streaming client 120.

Otherwise, according to the present embodiment, the transfer of events from a respective event queue will occur after a specified time interval has passed. Thus, in one embodiment, the regulators 241-244 of Figure 2 and the regulators 431-433 of Figure 4 can include a timer. The time interval may be different for each of the queues 231-234 (Figure 2) and 421-423 (Figure 4). The time interval may also be changed by the desktop streaming engine 250 (Figure 2) based on monitoring of the channels to the desktop streaming client 120 (Figure 1), or based on feedback from the desktop streaming client 120. This is described further in conjunction with Figures 5 and 6, below.

Figure 5 is a block diagram of a desktop streaming engine 250 according to one embodiment of the present invention. As illustrated by Figure 5, desktop streaming engine 250 includes a number of elements that are rendered separately for clarity of illustration and discussion; however, it is understood that these elements may not exist as separate entities within desktop streaming engine 250. In general, according to the various embodiments of the present invention, desktop streaming engine 250 provides the capability and functionality provided by the various elements of Figure 5. In addition, desktop streaming engine 250 may provide other capabilities and functionalities in addition to those described herein.

In the present embodiment, scheduler 510 is coupled to the regulators 580 (e.g., regulators 241-244 and 431-433 of Figures 2 and 4, respectively), which in turn are coupled to the queues 590 (e.g., queues 231-234 and 421-423 of Figures 2 and 4, respectively). Scheduler 510 is also coupled to a monitor 530. In one embodiment, the monitor 530 periodically measures the available bandwidth of the channels 571, 572 and 573. Although only three channels are shown, it is appreciated that there may be some number of channels other than three. In another embodiment, the monitor 530 also collects information from desktop streaming client 120 (Figure 1) with regard to the workload of the client as well as the attributes of the client device (e.g., the device's display capabilities such as screen size and resolution, the device's processing capabilities such as processor speed, the device's available memory capacity, and the like). Based on the information from monitor 530, scheduler 510 can set or adjust the trigger (e.g., the time interval, threshold or other mechanism) that is used to initiate the transfer of events from the various event queues. As mentioned above, different time intervals, thresholds, or the like can be specified for each of the queues.

The schedule for transferring events in the control event queue 231 and the window movement queue 232 (Figure 2) is relatively straightforward because these types of events require less communication overhead and therefore can be transferred almost if not immediately. The schedule for transferring events from the windows update queue 233 and the audio samples queue 234 can be adapted according to the information from monitor 530 (and also from monitor 630 of the client; see Figure 6), so as to provide a satisfactory display at client 102 (Figure 1).

Continuing with reference to Figure 5, and with reference also to Figures 2 and 4, transfer engine 540 forwards the events from event queues 590 to the appropriate channel based on the event type. For example, in one embodiment, events from control event queue 231, window movement queue 232, graphics window updates queue 421, and other window updates queue 423 can be forwarded to a TCP channel, while events from audio samples queue 234 and video window updates queue 422 can be forwarded to an RTP channel.

With reference to Figure 5, in the present embodiment, video and/or audio content may be encoded (compressed) using encoder 550. The degree of encoding is typically a function of the attributes of the client 102 (Figure 1), but the degree of encoding may also be influenced by the bandwidth of the connection (channel) between the client 102 and the server 101 (Figure 1).

In the present embodiment, the synchronizer 560 of Figure 5 synchronizes the audio with other related media parts. The media parts related to the audio can be text, graphics or video. In one embodiment, the audio and related portions are sent over separate channels (e.g., RTP channels). In that embodiment, synchronizer 560 requests scheduler 510 to obtain data from the corresponding samples queue (the audio samples queue 234, the window updates queue 233, and/or the window movement queue 232 of Figure 2) in a synchronized fashion. For example, if one second's worth of audio packets is acquired, one second's worth of video packets should also be acquired. This type of approach helps ensure that the audio and video packets will arrive at the client 102 (Figure 1) concurrently or within a short interval of each other. A small buffer may be utilized by client 102 to overcome any disparity in arrival times.

In another embodiment, with reference to Figure 5, the synchronization is done by encoder 550, regardless of whether or not the media parts (e.g, video data) are encoded. In this embodiment, encoder 550 can multiplex the audio and related media parts, and subsequently transfer the data to communication manager 570, which can then transfer the multiplexed packets over one channel (e.g., an RTP channel). A decoder on client 102 (Figure 1; also see Figure 6, below) can play the multiplexed data in a synchronized fashion. For example, within one second of actual time, the decoder can decode one second's worth of audio data, which can be place in an audio buffer. During the time remaining in the one second of actual time, the decoder can decode as many video frames as possible. In this manner, the desktop streaming server 110 of Figure 2 is able to take advantage of resources available on the client side.

Continuing with reference to Figure 5, in the present embodiment, communication manager 570 manages the communication channels 571-573. Also in the present embodiment, client event receiver 520 is for receiving control events (e.g., keyboard and mouse events) from desktop streaming client 120 (Figure 1). These events are forwarded by client event receiver 520 to driver interface 210 that, in one embodiment, forwards them to video driver 203 (Figure 2).

In summary, in one embodiment, the desktop streaming server 110 (Figure 2) intercepts events from the various drivers, filters those events and dispatches the events to the appropriate event queue based on event type, optionally encodes selected video data and audio data and streams that data to a client over a channel such as an RTP channel, and reacts to events received from the client.

Figure 6 is a block diagram of a desktop streaming client 120 according to one embodiment of the present invention. As illustrated by Figure 6, desktop streaming client 120 includes a number of elements that are rendered separately for clarity of illustration and discussion; however, it is understood that these elements may not exist as separate entities within desktop streaming client 120. In general, according to the various embodiments of the present invention, desktop streaming client 120 provides the capability and functionality provided by the various elements of Figure 6. In addition, desktop streaming client 120 may provide other capabilities and functionalities in addition to those described herein.

In the present embodiment, desktop streaming client 120 includes an agent 610 (which may also be referred to as a daemon). In this embodiment, the agent 610 establishes and maintains the channel connections with the server 101, specifically to the desktop streaming server 110 (Figure 1). As mentioned, these channels may utilize the TCP and RTP protocols. Agent 610 also forwards data received over the channels to processing engine 620. Agent 610 can also respond to inquiries from the desktop streaming server 110 regarding available channel bandwidth. Furthermore, at the beginning of a session, agent 610 can inform desktop streaming server 110 of the client's decoding capabilities as well as other client attributes, such as processor speed, and display screen size and resolution.

In the present embodiment, agent 610 of Figure 6 is coupled to a monitor 630, and can provide information from monitor 630 to the desktop streaming server 110 of Figure 1. In this embodiment, monitor 630 can monitor the workload of the client 102 (Figure 1), and can also provide information regarding available memory capacity. Also, monitor 630 can provide feedback regarding the efficiency in which the display is generated based on the events received from the desktop streaming server 110, so that the streaming rate, for example, can be adjusted accordingly. Thus, for example, the information from monitor 630 (and also from agent 610) can be used by the scheduler 510 of Figure 5 to set or adjust the trigger (e.g., the time interval, threshold or other mechanism) that is used to initiate the transfer of events from the various event queues of the desktop streaming server 110.

With reference to Figure 6, in the present embodiment, agent 610 is also coupled to processing engine 620. Events received by agent 610 are forwarded to processing engine 620. In one embodiment, processing engine 620 includes a decoder 622 for decoding (decompressing) data (events) if the data are encoded. The events from processing engine 620 are forwarded to driver interface 640, which functions in a manner similar to that described above for driver interface 210 (Figure 2) to forward the events to video driver 663 and/or audio driver 664.

In the present embodiment, driver interface 640 also receives events from keyboard driver 661 and mouse driver 662 of the client device. These events are forwarded to client event processor 650, which optimizes these events by eliminating non-critical events and then forwards the critical events to the desktop streaming server 110 (Figure 2) via agent 610. That is, for example, for a particular cursor movement, client event processor 650 may eliminate intermediate coordinates associated with the cursor movement, identifying and keeping only the start and end coordinates of the movement for transfer to the desktop streaming server 110.

Thus, in summary, in one embodiment, the desktop streaming client 120 (Figure 6) decodes events transferred from the server (when the events are encoded), updates the display based on the events, provides audio playback when the events include audio samples, responds to inquiries from the desktop streaming server 110 (Figure 2), periodically provides monitoring information to the desktop streaming server 110, and intercepts control events and sends them to the desktop streaming server 110.

Figure 7 is a flowchart 700 of a method for transferring an event from a server to a client according to one embodiment of the present invention. Although specific steps are disclosed in flowchart 700, such steps are exemplary. That is, embodiments of the present invention are well suited to performing various other steps or variations of the steps recited in flowchart 700. It is appreciated that the steps in flowchart 700 may be performed in an order different than presented, and that not all of the steps in flowchart 700 may be performed. All of, or a portion of, the methods described by flowchart 700 may be implemented using computer-readable and computer-executable instructions which reside, for example, in computer-usable media of a computer system or like device. Generally, flowchart 700 is implemented by devices such as server 101 of Figure 1.

In step 710 of Figure 7, in the present embodiment, an event is received from a driver. The event may be a keyboard event, a cursor control (e.g, mouse) event, a video event, or an audio event.

In step 720, in the present embodiment, the event is dispatched to an event queue according to the event type. That is, events of the same type are forwarded to a particular queue. In one embodiment, the events are filtered to identify event type.

In step 730, in the present embodiment, the event is processed according to the event type. For example, when the event includes multimedia data (e.g., video data and/or audio data), the event may be encoded. For cursor movement, the coordinates of the points between the start and end points may be eliminated, with only the coordinates of the start and end point retained.

In step 740, in the present embodiment, the event is transferred to the client when triggered. The transfer to the client occurs according to a protocol that corresponds to the type of event. In one embodiment, event transfer is triggered by the passage of a specified time interval. In another embodiment, event transfer is triggered when the number of events in the event queue reaches a specified threshold.

Embodiments of the present invention are thus described. While the present invention has been described in particular embodiments, it should be appreciated that the present invention should not be construed as limited by such embodiments, but rather construed according to the following claims.

CLAIMS

What is claimed is:

1. A method (700) for transferring an event from a server to a remote client, said method comprising:

(710) receiving an event from a driver;

(720) dispatching said event into an event queue according to event type;

(730) processing said event according to said event type, wherein said processing comprises encoding said event when said event comprises multimedia data; and

(740) transferring said event to said remote client when triggered, wherein said transferring occurs according to a protocol corresponding to said event type.

2. The method of Claim 1 wherein said event is a keyboard event, a mouse event, a video event or an audio event and wherein said driver is a keyboard driver, a mouse driver, a video driver or an audio driver.

3. The method of Claim 2 wherein said protocol is Transport Control Protocol (TCP) for keyboard, mouse and graphics events and Real Time Protocol (RTP) for video and audio events.

4. The method of Claim 1 wherein said dispatching comprises:

(720) filtering said event to identify said event type.

5. The method of Claim 1 wherein said processing of multimedia data further comprises:

(730) synchronizing an audio portion and a related media portion of said multimedia data during said encoding, said related media portion selected from the group consisting of text, graphics and video.

6. The method of Claim 1 wherein said transferring is triggered by a timer.

7. The method of Claim 1 wherein said transferring is triggered when the number of events in said event queue reaches a threshold.

8. The method of Claim 1 further comprising:

(520) receiving events from said remote client.

9. A system (110) for transferring an event to a remote client, said system comprising:

a driver interface (210) for receiving different types of events from a plurality of drivers (201-204);

an event filter (220) coupled to said driver interface, said event filter for identifying an event by event type and for placing said event into an event queue (231-234, 421-423) according to said event type;

a regulator (241-244, 431-433) coupled to said event queue, said regulator for impelling said event from said event queue to a processing engine that encodes said event when said event comprises multimedia data; and

a channel (100) coupled to said processing engine, said channel for transferring said event to said remote client according to a protocol corresponding to said event type.

10. A system (120) for receiving events from a remote server, said system comprising:

an agent (610) for managing a plurality of channels used for receiving different types of events from said remote server, each channel associated with one of a plurality of different protocols, wherein one of said channels is for streaming multimedia data;

a processing engine (620) coupled to said channels, said processing engine for decoding an event when said event comprises encoded multimedia data; and

a driver interface (640) coupled to said processing engine, said driver interface for forwarding said events to audio and video drivers according to event type, wherein said events are displayable on a display device.

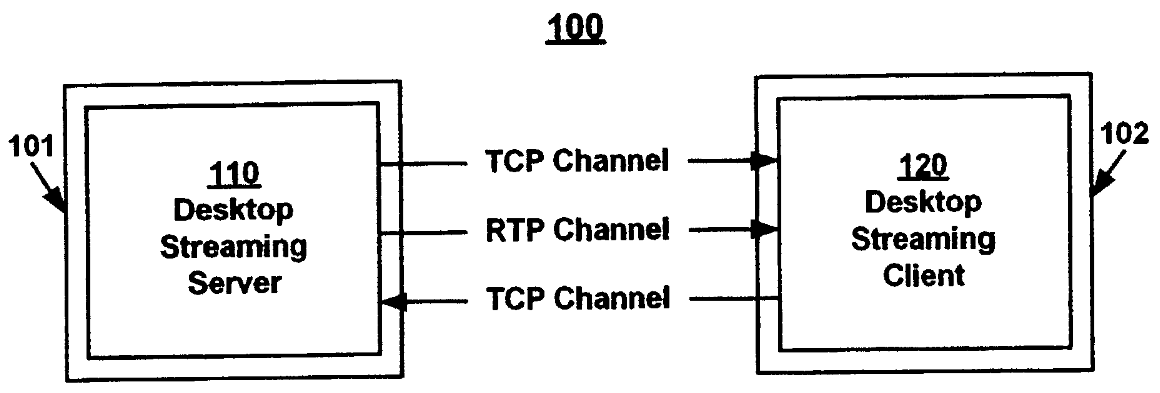

100
101
110
Desktop
Streaming
Server
TCP Channel
RTP Channel
TCP Channel
120
Desktop
Streaming
Client
102

FIG. 1

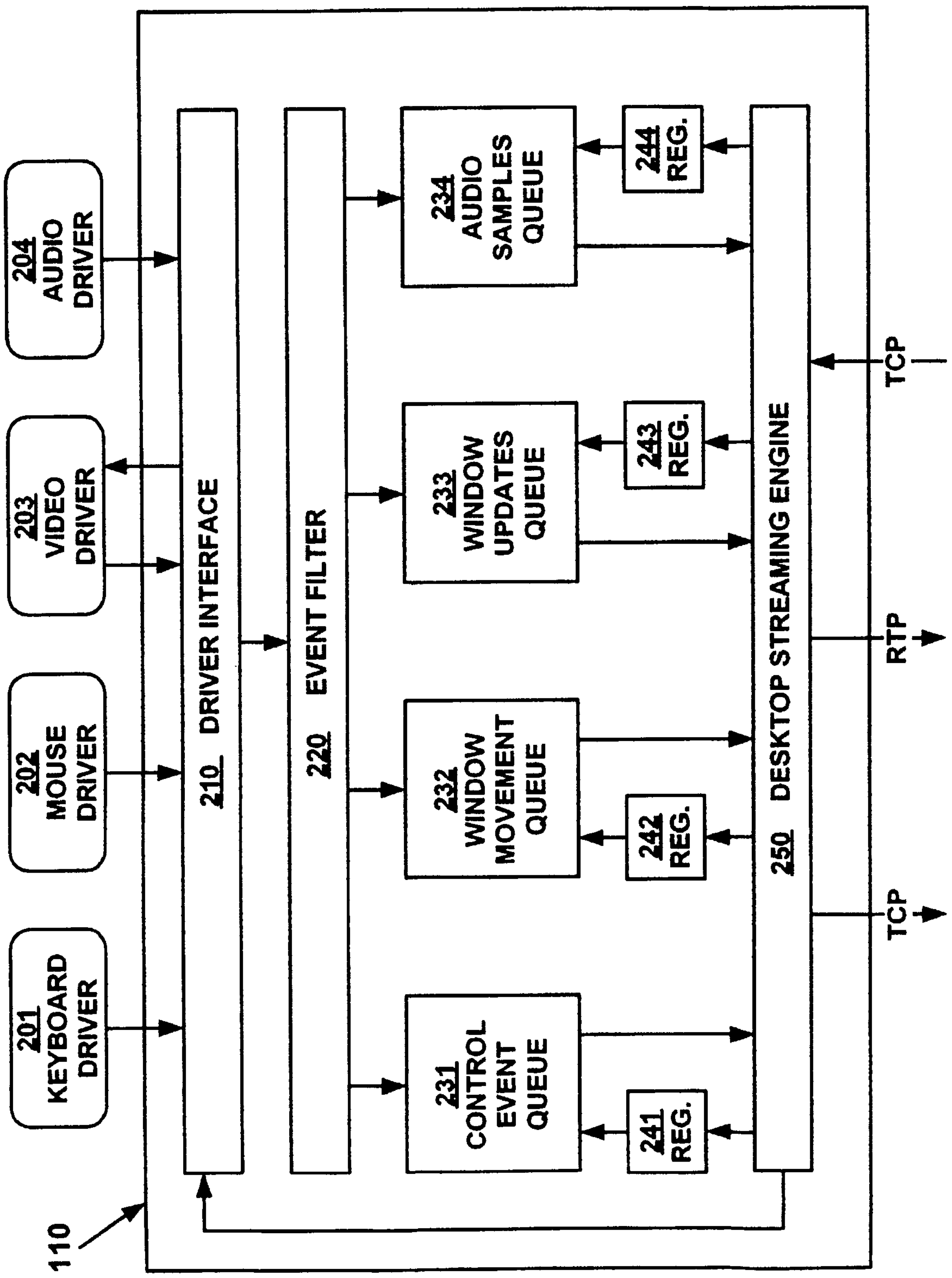

110
201 KEYBOARD DRIVER
202 MOUSE DRIVER
203 VIDEO DRIVER
204 AUDIO DRIVER
210 DRIVER INTERFACE
220 EVENT FILTER
231 CONTROL EVENT QUEUE
232 WINDOW MOVEMENT QUEUE
233 WINDOW UPDATES QUEUE
234 AUDIO SAMPLES QUEUE
241 REG.
242 REG.
243 REG.
244 REG.
250 DESKTOP STREAMING ENGINE
TCP
RTP
TCP

FIG. 2

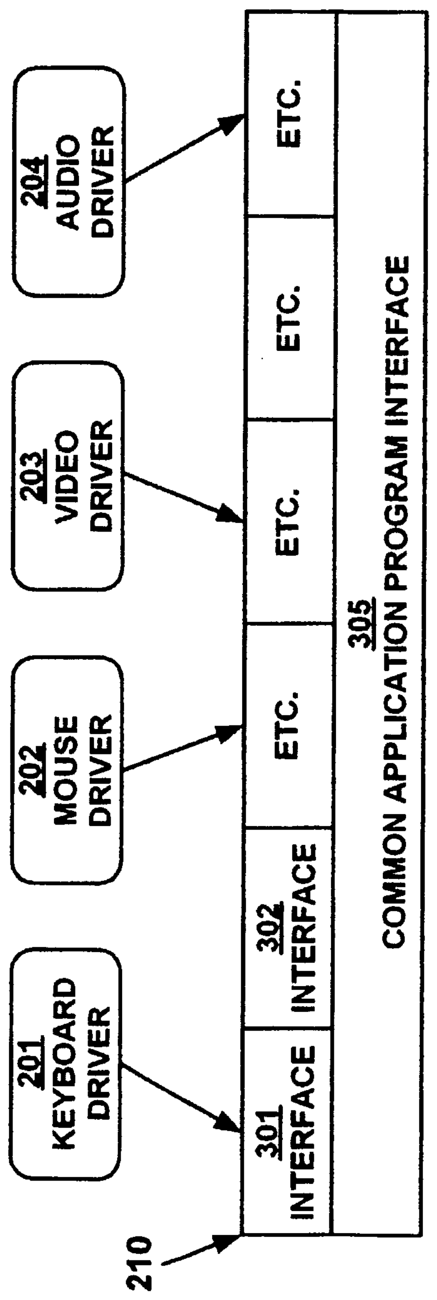

201 KEYBOARD DRIVER
202 MOUSE DRIVER
203 VIDEO DRIVER
204 AUDIO DRIVER
210
301 INTERFACE
302 INTERFACE
ETC.
ETC.
ETC.
ETC.
305 COMMON APPLICATION PROGRAM INTERFACE

FIG. 3

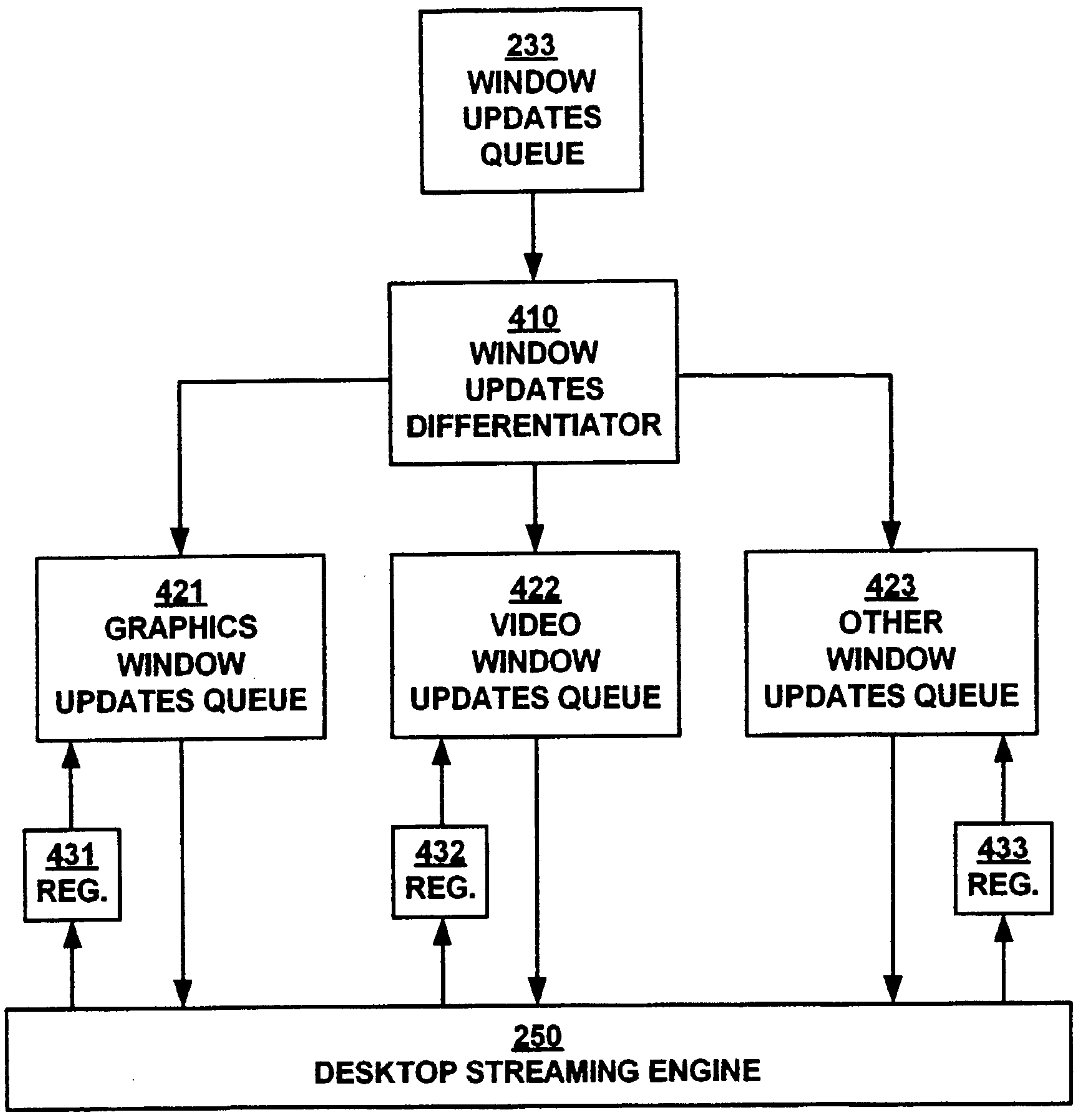
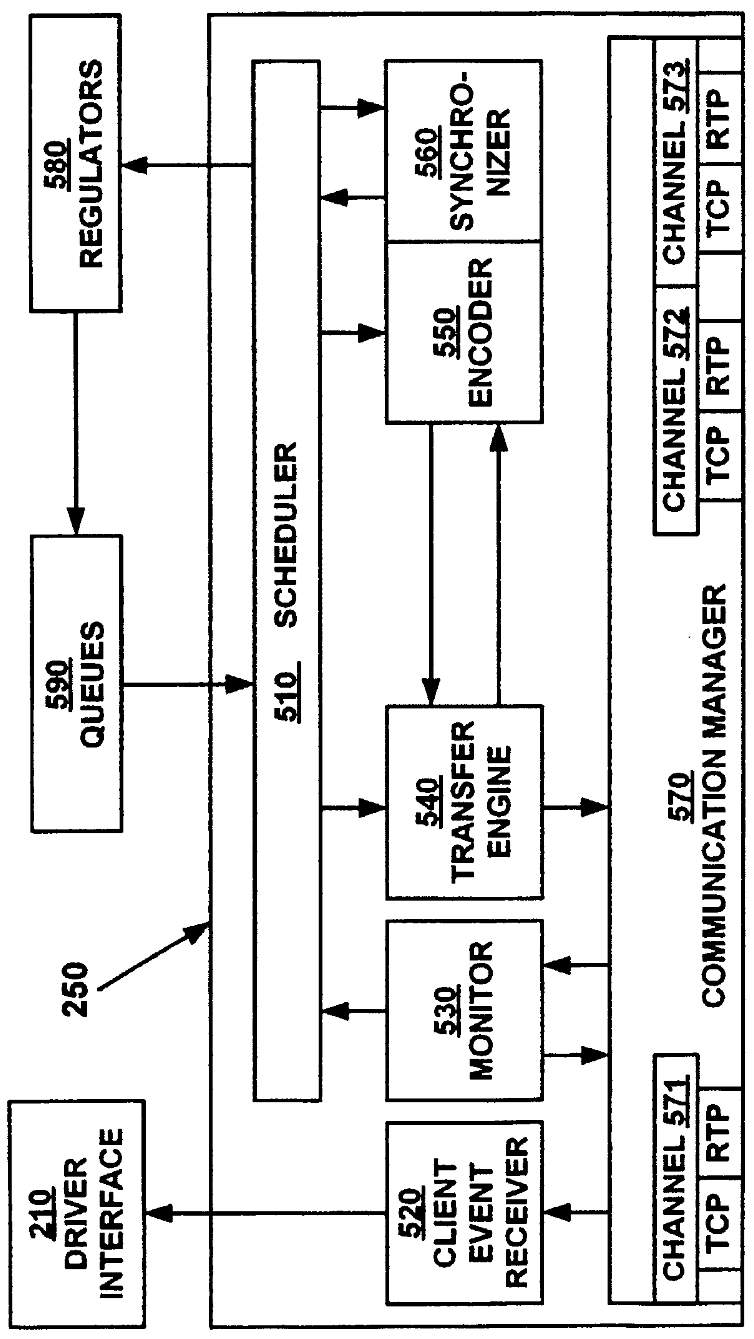

233
WINDOW UPDATES QUEUE
410
WINDOW UPDATES DIFFERENTIATOR
421
GRAPHICS WINDOW UPDATES QUEUE
422
VIDEO WINDOW UPDATES QUEUE
423
OTHER WINDOW UPDATES QUEUE
431
REG.
432
REG.
433
REG.
250
DESKTOP STREAMING ENGINE

FIG. 4



580 REGULATORS
590 QUEUES
210 DRIVER INTERFACE
250
510 SCHEDULER
560 SYNCHRO-NIZER
550 ENCODER
540 TRANSFER ENGINE
530 MONITOR
520 CLIENT EVENT RECEIVER
570 COMMUNICATION MANAGER
CHANNEL 573
CHANNEL 572
CHANNEL 571
TCP
RTP
TCP
RTP
TCP
RTP

FIG. 5

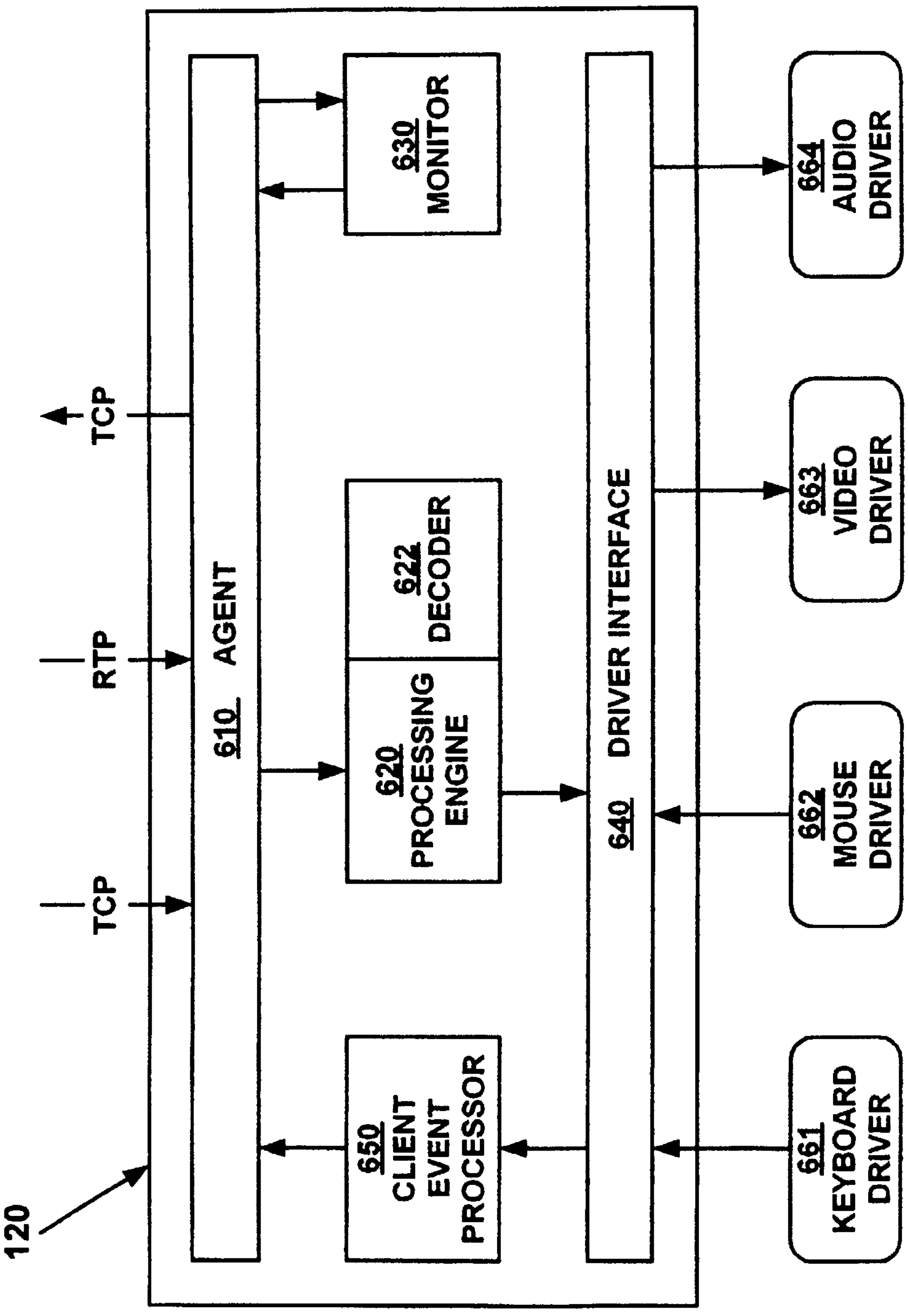

120
TCP
RTP
TCP
610 AGENT
630 MONITOR
622 DECODER
620 PROCESSING ENGINE
650 CLIENT EVENT PROCESSOR
640 DRIVER INTERFACE
664 AUDIO DRIVER
663 VIDEO DRIVER
662 MOUSE DRIVER
661 KEYBOARD DRIVER

FIG. 6

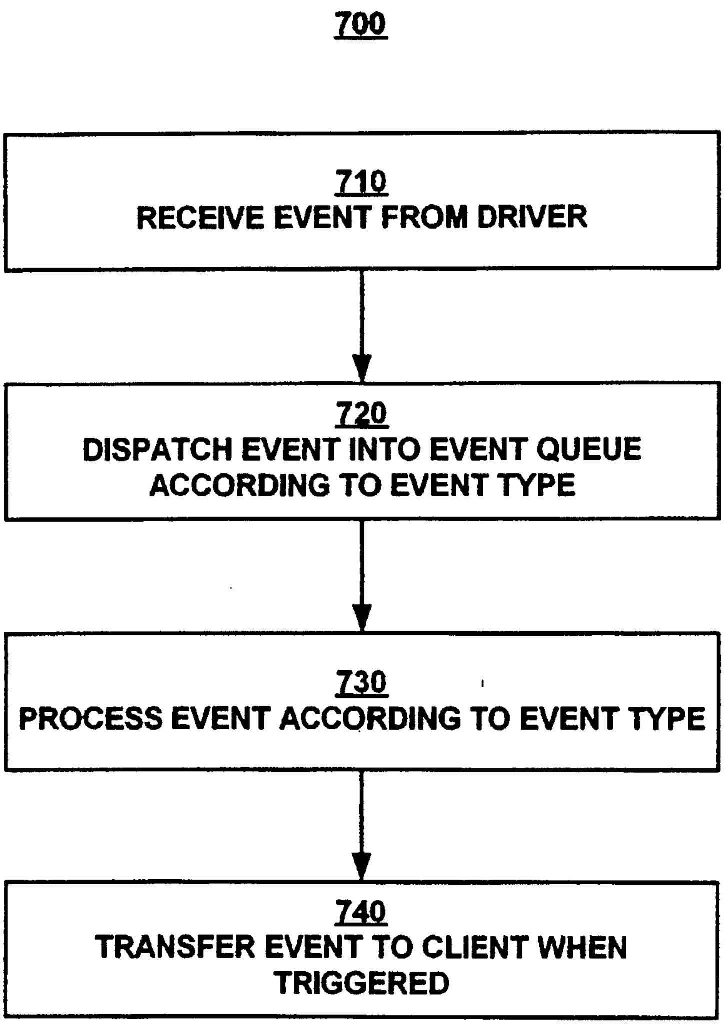

700
710
RECEIVE EVENT FROM DRIVER
720
DISPATCH EVENT INTO EVENT QUEUE ACCORDING TO EVENT TYPE
730
PROCESS EVENT ACCORDING TO EVENT TYPE
740
TRANSFER EVENT TO CLIENT WHEN TRIGGERED

FIG. 7

# INTERNATIONAL SEARCH REPORT

Inte[illegible]al Application No
PCT/US 03/35821

**A. CLASSIFICATION OF SUBJECT MATTER**
IPC 7 H04L29/06 H04L29/08 G06F17/60 A63F13/12 H04N7/24
H04N7/173

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)
IPC 7 H04L G06F A63F H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

COMPENDEX, EPO-Internal, INSPEC, PAJ, IBM-TDB, WPI Data

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category ° | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| A | CHIA-CHEN KUO ET AL: "Design and implementation of a network application architecture for thin clients" PROCEEDINGS OF THE 26TH. ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE. COMPSAC 2002. OXFORD, ENGLAND, AUG. 26 - 29, 2002, ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, LOS ALAMITOS, CA: IEEE COMP. SOC, US, vol. CONF. 26, 26 August 2002 (2002-08-26), pages 193-198, XP010611116 ISBN: 0-7695-1727-7 the whole document<br>---<br>-/-- | 1-10 |

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° 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 document 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<br>23 April 2004 | Date of mailing of the international search report<br>29/04/2004 |
| Name and mailing address of the ISA<br>European Patent Office, P.B. 5818 Patentlaan 2<br>NL – 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,<br>Fax: (+31-70) 340-3016 | Authorized officer<br>Niculiu, R |

# INTERNATIONAL SEARCH REPORT

In[illegible]nal Application No
PCT/US 03/35821

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

| Category ° | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| A | US 6 166 734 A (NAHI PAUL B ET AL)<br>26 December 2000 (2000-12-26)<br>abstract<br>column 9, line 62 -column 11, line 28<br>column 12, line 46 -column 14, line 7<br>column 15, line 14 -column 18, line 65<br>column 20, line 8 - line 44<br>claims 1,5,10,22<br>--- | 1-10 |
| A | WO 00/39678 A (CITRIX SYSTEMS INC)<br>6 July 2000 (2000-07-06)<br>abstract<br>page 8, line 12 -page 9, line 2<br>page 11, line 1 -page 14, line 7<br>page 16, line 13 -page 19, line 2<br>page 26, line 14 -page 30, line 12<br>--- | 1-10 |
| A | US 6 259 443 B1 (WILLIAMS JR HENRY R)<br>10 July 2001 (2001-07-10)<br>abstract<br>column 4, line 1 -column 6, line 65<br>column 8, line 8 -column 107<br>column 12, line 7 - line 41<br>column 13, line 17 -column 14, line 5<br>----- | 1-10 |

# INTERNATIONAL SEARCH REPORT

Inte[illegible]al Application No
PCT/US 03/35821

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| US 6166734 A | 26-12-2000 | NONE | |
| WO 0039678 A | 06-07-2000 | US 2002103884 A1 | 01-08-2002 |
| | | AT 252744 T | 15-11-2003 |
| | | AU 2212900 A | 31-07-2000 |
| | | AU 765088 B2 | 11-09-2003 |
| | | AU 6555299 A | 06-07-2000 |
| | | CA 2293127 A1 | 29-06-2000 |
| | | DE 69912317 D1 | 27-11-2003 |
| | | DK 1141828 T3 | 23-02-2004 |
| | | EP 1411429 A2 | 21-04-2004 |
| | | EP 1141828 A1 | 10-10-2001 |
| | | GB 2349488 A | 01-11-2000 |
| | | JP 2002533830 T | 08-10-2002 |
| | | WO 0039678 A1 | 06-07-2000 |
| US 6259443 B1 | 10-07-2001 | NONE | |