Title: APPARATUSES, METHODS AND SYSTEMS FOR AN INTERACTIVE PROXIMITY DISPLAY TETHER (IPDT) teaches an Interactive Proximity Display Tether (IPDT), which provides an interactive communications and display tether between source and target devices. The IPDT allows users to employ source mobile devices on the large display of target devices. For example, a user may employ a source device, such as the iPhone, and launch a IPDT implemented video game to employ larger display devices as targets, which allows for greater utility in operating various applications, and particularly in game environments. Devices like the iPhone, which allow use of accelerometers whereby the entire device may act as an input (e.g., game) controller are enhanced with IPDT by allowing a user to both control inputs to the application (e.g., control the game) while still being able to fully view the results of the application.
APPARATUSES, METHODS AND SYSTEMS FOR AN INTERACTIVE PROXIMITY DISPLAY TETHER

RELATED APPLICATIONS

Applicant hereby claims priority under 35 USC §119 for United States provisional patent application serial no. 61/108,565 filed on October 27, 2008, entitled "APPARATUSES, METHODS AND SYSTEMS FOR AN INTERACTIVE PROXIMITY DISPLAY TETHER," attorney docket no. 19626-002PV, and for United States provisional patent application serial no. 61/116,175 filed on November 19, 2008, entitled "APPARATUSES, METHODS AND SYSTEMS FOR AN INTERACTIVE PROXIMITY DISPLAY TETHER," attorney docket no. 19626-002PV1, and for United States provisional patent application serial no. 61/151,832 filed on February 11, 2009, entitled "APPARATUSES, METHODS AND SYSTEMS FOR REMOTE INTERACTIVE REALTIME CO-PLAY," attorney docket no. 19626-004PV.

The entire contents of the aforementioned applications are herein expressly incorporated by reference.

FIELD

The present invention is directed generally to an apparatuses, methods, and systems of interactive display, and more particularly, to APPARATUSES, METHODS AND SYSTEMS FOR AN INTERACTIVE PROXIMITY DISPLAY TETHER (hereinafter "IPDT").
BACKGROUND

[0004] Portable game environments such as the Nintendo Gameboy, Nintendo DS, Sony PSP, and mobile communications handsets such as the iPhone, Palm, Blackberry and/or other devices allow users to engage in gaming in a mobile fashion. These devices typically have screens of portable dimension and allow users to engage in gaming on the go.

SUMMARY

[0005] The Interactive Proximity Display Tether (IPDT) allows users to employ source devices on larger displays of target devices. The IPDT is particularly useful in allowing source devices, such as the iPhone, to employ larger display devices as targets, which allows for greater utility in operating various applications, and particularly in game environments. Devices like the iPhone, which allow use of accelerometers whereby the entire device may act as an input (e.g., game) controller are enhanced with IPDT by allowing a user to both control inputs to the application (e.g., control the game) while still being able to fully view the results of the application. For example, when playing a game of virtual golf, a user may simulate a golf swing with the iPhone and the iPhone may be used as an input device, however, the user will not be able to view the swing on the iPhone during the swing. The IPDT allows smaller mobile devices, such as, but not limited to the Apple iPhone, and or/the like, to cast its display to other devices such as large screen computers, laptops, set top boxes, cable boxes, and televisions. Interactive gaming among multiple users with mobile devices such as the iPhone, similarly benefits
from the ability to cast and interact via the IPDT. Similarly, business and personal productivity applications, such as slide show presentations, benefit from applications being enhanced with the IPDT, thereby allowing a user to employ almost any conveniently accessible target device as an extension of their source device.
BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The accompanying appendices and drawings illustrate various non-limiting, example, inventive aspects in accordance with the present disclosure:

[0007] FIGURE 1 is of a block diagram illustrating an overview of an implementation of data flows between an interactive proximity display tether with realtime co-play (hereinafter "IPDT") system and affiliated entities in one embodiment of the IPDT operation;

[0008] FIGURE 2 provides an implementation of IPDT system components in one embodiment of IPDT operation;

[0009] FIGURES 3A-3F provides logic flow diagrams and examples of screenshots illustrating aspects of interactive proximity display tethering within embodiments of the IPDT operation;

[0010] FIGURE 4 provide block diagrams illustrating examples of data formats of interactive proximity display tethering within embodiments of the IPDT operation;

[0011] FIGURE 5A-5D provide diagrams illustrating example implementations of Co-play IPDT within embodiments of the IPDT operation;

[0012] FIGURE 6A-6B provide logic flow diagrams illustrating aspects of realtime co-play within embodiments of the IPDT operation;

[0013] FIGURE 7A-7C provide diagrams of aspects of subscription process and other embodiments of the IPDT in one embodiment of the IPDT operation;
FIGURE 8 illustrate implementations of an IPDT housing configured as a golf club in one embodiment;

FIGURE 9 is of a block diagram illustrating embodiments of the IPDT controller;

The leading number of each reference number within the drawings indicates the figure in which that reference number is introduced and/or detailed. As such, a detailed discussion of reference number 101 would be found and/or introduced in Figure 1. Reference number 201 is introduced in Figure 2, etc.
DETAILED DESCRIPTION

This disclosure details the implementation of APPARATUSES, METHODS AND SYSTEMS FOR AN INTERACTIVE PROXIMITY DISPLAY TETHER WITH REALTIME CO-PLAY (hereinafter "IPDT"). IPDT implements an interactive application at a user handset to tether with a target device and/or another user handset whereby users may project a local screen running a gaming application on the handsets to a larger display device.

For example, in one embodiment, IPDT may be implemented on various mobile devices, such as on a smart phone platform, e.g. Apple's iPhone OS, Google's Android OS, Blackberry's OS, and/or the like. In one embodiment, a user operating such a mobile device may select to play with an IPDT gaming application from his/her mobile device, e.g. an iGolf game, etc. If the user launch the IPDT application, for example, by clicking an IPDT component icon from the mobile device screen menu, the mobile device may query for available target devices within a local area network. For instance, in one implementation, the mobile device may search for a laptop, a desktop, a projector, a television, and/or the like that are registered within a Bluetooth range. The user may then be provided a list of search results of available target device and may enter a selection. In one implementation, the user may choose a television, and the mobile device may establish a communication channel and tether with the television. In one implementation, the gaming screen on the mobile device may be projected to a larger display of the television, which allows the user to operate the mobile device as a
remote game controller while the video gaming screen is displayed in a realtime and interactive manner on the television.

In one embodiment, a method is disclosed, comprising: receiving a request to initialize a remote display tether component; querying for tether target devices; obtaining a selection of a tether target device; configuring a communications channel for a selected tether target device; determining a type of the selected tether target device; instantiating a remote display through the communications channel if the selected tether target device is processor-based; instantiating an interactive tether application on a source device if the selected tether target device is processor-based; generating data based on control inputs; and casting the generated data to drive the remote display through the communications channel.

It is to be understood that, depending on the particular needs and/or characteristics of an IPDT user, administrator, server, data, monetization structure, hardware configuration, network framework, and/or the like, various embodiments of the IPDT may be implemented that facilitate a great deal of flexibility and customization. The instant disclosure discusses embodiments of the IPDT primarily within the context of video gaming applications. However, it is to be understood that the system described herein may be readily configured/customized for a wide range of other applications or implementations. For example, aspects of the IPDT may be adapted for cryptographic communications, artificial intelligence simulations, remote access presentation and/or the like. It is to be understood that the IPDT may be further adapted to other implementations for general network management applications and network protocol designs.
FIGURE i is a block diagram illustrating an overview of a practical implementation of data flows between an IPDT system and affiliated entities in one embodiment of the IPDT operation. In Figure 1, a user (or users) operating a source device 110, a target device 120 with a remote display 125, an IPDT database 119 and a system administrator 130 are shown to interact via a communication network 113.

In one embodiment, the user 110 may operate with a source device 110 to connect to and share a screen of the source display 115 of the source device with a remote display 125 of a target device 120. The source device 110 may include a wide variety of different devices and technologies such as, but not limited to mobile devices, dedicated game handsets, general computing devices, and/or the like. The target devices 120 may include devices and technologies such as mobile handsets, dedicated game handsets, general computing devices, game consoles, set top boxes, cable boxes, video displays, and/or the like. For example, as illustrated in Figure 1, in one implementation, the source device 110 may be a portable handset, such as an Apple Inc. iPhone, while the target device 120 may be a computer with a display screen 125. For another example, the source device 110 may be a mobile device such as an Apple iPhone secured in a housing shaped as a gaming implement, such as a golf club for a golf game (as will be further illustrated in one implementation in Figure 8), tennis racket for tennis game, fishing pole for a fishing game, baseball bat for a baseball game, and/or the like.

The IPDT facilitates connections through the communication network 113 based on a broad range of protocols that include WiFi, Bluetooth, 3G cellular, Ethernet, physical tethers (e.g., iPhone Video AV to Dock Connector Cable, which allows for...
connection to a monitor or TV), and/or the like. In one embodiment, the communication network 113 may be the Internet, a Wide Area Network (WAN), a telephony network, a Local Area Network (LAN), a Peer-to-Peer (P2P) connection, and/or the like. In one embodiment, the source device 110 may detect, handshake and interact with the target device 120 to exchange control information and data payloads via the communication network 113, as will be further illustrate in one implementation in Figure 4. For example, in one implementation, the communication network 113 provides a communications path such that the source device 110 may project its source display 115 onto the target device remote display 125 one of which may also open the communication path to client device 112. In this manner, a relatively small source device may drive a larger display on a target display, as well as communication with one or more client devices, allowing for tethered and interactive control by the source device using the remote display 125. In another embodiment, the IPDT may implement an interactive control scheme that allows for the interactive control of games by one or more source devices 110 via the larger remote display 125 of the target device 120. For example, in one implementation, a user 105 may co-play a gaming application displayed at the target device with another user 105. In this manner, the source devices may be configured to communicate with each other and with the target device in different modes, as will be further illustrated in one implementation in Figures 5A-5D.

[0023] In one embodiment, the IPDT entities such as the source device 110, the target device 120 and/or the like, may also communicate with a IPDT database 119. In some embodiments, distributed IPDT databases may be integrated in-house with the target device 120, and/or the source device 110. In other embodiments, the IPDT entities may access a remote IPDT database 119 via the communication network 113. In
In one embodiment, the IPDT entities may send data to the database 119 for storage, such as, but not limited to, user account information, application data, protocol data, application history, and/or the like.

In one embodiment, the IPDT database 119 may be one or more online database connected to a variety of vendors, such as, but not limited to, hardware vendors (e.g., Apple Inc., Intel, Sony, etc.), gaming application vendors (e.g., Nintendo, Game Cube, Game Boy, etc.), service vendors (e.g., PlayStation Network, \VliC0nnect24, etc.) and/or the like, and obtain updated hardware driver information, new gaming application packages and services from such vendors. In one embodiment, the source device 110 and/or the target device 120 may constantly, intermittently, and/or periodically download updates, such as updated user profile, updated software programs, updated command instructions, and/or the like, from the IPDT database 119 via a variety of connection protocols, such as Telnet FTP, HTTP transfer, P2P transmission and/or the like.

In a further embodiment, the target device 120 and the source device 110 may connect to an online gaming server 130 via the communication network 113. For example, in one implementation, users 105 may employ source devices 110 to join an Internet game community (e.g., F.A.S.T., etc.) at an online gaming server 130, which is locally displayed at the target device 120.

In one embodiment, a system administrator 140 may communicate with the IPDT entities for regular maintenance, service failure, system updates, database renewal, security surveillance and/or the like via the communication network 113. For example, in one implementation, the system administrator may be a user, who may directly operate with the target device 120 to configure system settings, parental control,
and/or the like. In another implementation, the system administrator may be a service vendor for Internet gaming.

[0027] FIGURE 2 illustrates an implementation of IPDT system components in one embodiment of IPDT operation. An IPDT device 201 may contain a number of functional modules and/or data stores. An IPDT controller 205 may serve a central role in some embodiments of IPDT operation, serving to orchestrate the reception, generation, and distribution of data and/or instructions to, from and between target device(s) and/or client device(s) via IPDT modules and in some instances mediating communications with external entities and systems.

[0028] In one embodiment, the IPDT controller 205 may be housed separately from other modules and/or databases within the IPDT system, while in another embodiment, some or all of the other modules and/or databases may be housed within and/or configured as part of the IPDT controller. Further detail regarding implementations of IPDT controller operations, modules, and databases is provided below.

[0029] In one embodiment, the IPDT Controller 205 may be coupled to one or more interface components and/or modules. In one embodiment, the IPDT Controller may be coupled to a user interface (UI) 210, a maintenance interface 212, and a power interface 214. The user interface 210 may be configured to receive user inputs and display application states and/or other outputs. The UI may, for example, allow a user to adjust IPDT system settings, select communication methods and/or protocols, initiate a remote display mode, engage mobile device application features, identify possible target/client device(s) and/or the like. In one implementation, the user
interface 210 may include, but not limited to devices such as, keyboard(s), mouse, stylus(es), touch screen(s), digital display(s), and/or the like. In one embodiment, the maintenance interface 212 may, for example, configure regular inspection and repairs, receive system upgrade data, report system behaviors, and/or the like. In one embodiment, the power interface 214 may, for example, connect the IPDT controlled 205 to an embedded battery and/or an external power source.

[0030] In one embodiment, the IPDT Controller may further be coupled to an applications engine 260, configured to run device application software. In one implementation, the applications engine 260 may receive sensory input information originating from one or more integrated sensors and interpret the information to update the configuration of an application state. In an implementation the updated application state data may be transferred to a target, client and/or source device depending on the implementation for display. For example, an application run by the applications engine may comprise a video game, such as may be controlled via a motion-sensitive mobile device, which uses IPDT to establish a communications channel with a laptop, configured in turn, to display transferred video game data.

[0031] In one implementation, the IPDT Controller 205 may further be coupled to a sensor module 220, configured to interface with and/or process signals from sensor input/output (I/O) components 225. The sensor I/O components 225 may be stimulated by user manipulation, environmental conditions, and/or the like to generate electrical signals that may be received and/or processed by the sensor module 220 and/or other IPDT components, which in turn act to generate input controls which can be used by the application. A wide variety of different sensors may be compatible with
IPDT operation and may be integrated with sensor I/O components 225, such as but not limited to transducers, accelerometers, thermometers, anemometers, barometers, microphones, and/or the like, configured to measure states of motion, sound level, volume, pitch, pressure, wind speed, temperature, data transfer rate, light intensity level, position, elevation, weather, moisture level, humidity, and/or the like. In one implementation, the sensor module 220 may configure signals received from the sensor I/O components 225 in a form suitable for an application being run by the applications engine 260. In another implementation, the applications engine 260 may receive signals directly from sensor I/O components 225 for processing to update an application state for one or more running applications. For example, in one implementation, a user may engage a IPDT remote control device housing in a golf club (as will be further illustrated in one implementation in Figure 8). The user may swing the remote control device as if swinging a real golf club in field, and the sensor I/O 225 may detect signals of the motion of the club and transfer the signals (e.g. electrical pulses from accelerometers indicating a velocity and a direction of a swing, etc.) suitable to the sensor module 220. The sensor module 220 may process and analyze the received signals and generate data describing characteristics of the movement, e.g. direction of the movement, speed of the movement, motion level, etc., and transmit the data to the IPDT controller 205. For example, in one embodiment, the iPhone SDK toolkit and/or runtime libraries may be installed and/or used to access and interpret such actions.

In one embodiment, the IPDT Controller 205 may further be coupled to a communications module 230, configured to interface with and/or process signals from communications I/O components 235. The communications I/O components 235 may comprise components facilitating transmission of electronic communications via a
variety of different communication protocols and/or formats as coordinated with and/or by the communications module 230. Communication I/O components 240 may, for example, contain ports, slots, antennas, amplifiers, and/or the like to facilitate transmission of display instructions, such as may instruct a remote display what and/or how to display aspects of a mobile device application state, via any of the aforementioned methods. Communication protocols and/or formats for which the communications module 230 and/or communications I/O components 235 may be compatible may include, but are not limited to, GSM, GPRS, W-CDMA, CDMA, CDMA2000, HSDPA, Ethernet, WiFi, Bluetooth, USB, and/or the like. In various implementations, the communication I/O 235 may, for example, serve to configure data into application, transport, network, media access control, and/or physical layer formats in accordance with a network transmission protocol, such as, but not limited to FTP, TCP/IP, SMTP, Short Message Peer-to-Peer (SMPP) and/or the like. The communications module 230 and communications I/O 235 may further be configurable to implement and/or translate Wireless Application Protocol (WAP), VoIP and/or the like data formats and/or protocols. The communications I/O 235 may further house one or more ports, jacks, antennas, and/or the like to facilitate wired and/or wireless communications with and/or within the IPDT system. For instance, in the above example, the IPDT controller 205 may transmit the received sensor data characteristics of the movement of the controller device to the communication module 230, and the data may then be transmitted to external entities (e.g. the target device, etc.) through the communications I/O 235.

Numerous data transfer protocols may also be employed as IPDT connections, for example, TCP/IP and/or higher protocols such as HTTP post, FTP put
commands, and/or the like. In one implementation, the communications module 230 may comprise web server software equipped to configure application state data for publication on the World Wide Web. Published application state data may, in one implementation, be represented as an integrated video, animation, rich internet application, and/or the like configured in accordance with a multimedia plug-in such as Adobe Flash. In another implementation, the communications module 230 may comprise remote access software, such as Citrix, Virtual Network Computing (VNC), and/or the like equipped to configure application state data for viewing on a remote client (e.g., a remote display device).

In one implementation, the IPDT controller 205 may further be coupled to a plurality of databases configured to store and maintain IPDT data. An applications database 240 may contain application data, user IDs, settings, configurations, saved games, game states, application interface elements, and/or the like. A protocols database 245 may include data pertaining to communication protocols and/or data configurations suitable for publication on the World Wide Web, sharing between client and server devices in a remote-access software setup, and/or the like. A user database 250 may contain information pertaining to account information, contact information, profile information, identities of hardware devices, Customer Premise Equipments (CPEs), and/or the like associated with users, application history, system configurations, and/or the like. A hardware database 245 may contain information pertaining to hardware devices with which the IPDT system may communicate, such as but not limited to user devices, display devices, target devices, Email servers, user telephony devices, CPEs, gateways, routers, user terminals, and/or the like. The hardware database 228 may specify transmission protocols, data formats, and/or the
like suitable for communicating with hardware devices employed by any of a variety of IPDT affiliated entities.

[0035] In one embodiment, the IPDT databases may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. For example, in one implementation, the XML for the User Profile in the user database 250 may take a form similar to the following example:

```xml
<User>
  <Quasi-static info>
    <User_ID>i23-45-6789</User_ID>
    <Hardware ID>SDASFK45é32_iPhone 3.0</Hardware ID>
    <Census info>John Smith; 123 Maple Dr., Smalltown, CA 92676; (123)456-7890; jsmith@email.com; 55 years; male; white; married; 2 children; etc.</Census info>
  </Quasi-static info>
  ...
  </Quasi-static info>
  <Dynamic info>
  <Application history>
  <Last login>
    <Server ID>US-CA-ADD00089</Server ID>
    <Time>19:33:25 08-30-2009</Time>
    ...
  </Last login>
  ...
```
FIGURE 3A is of a logic flow diagram illustrating aspects of interactive proximity display tethering within embodiments of the IPDT operation. In one embodiment, a user wishing to tether a remote display may engage an application that is implemented with an IPDT component. In an implementation, the IPDT component may be implemented as a software development kit (SDK) and/or as an object library that may be included in any application where an IPDT may be useful. In one embodiment, any program that incorporates the IPDT component may become available and/or downloaded and installed on a source device, where a user will have the option to run the IPDT application 300, and a user may submit request to initialize the IPDT application 302. For example, in one implementation, a user may initialize a "remote game control" application from an IPDT mobile device by selecting an IPDT icon from the iPhone menu, and enter an IPDT gaming application, as illustrated in one implementation in screen 370 of Figure 3D. Once the application is selected and begins its instantiation, the IPDT component may query for tether targets where the source device may search for appropriate tether devices via all of its communication channels 310, as illustrated in one implementation in screen 372 of Figure 3D. In one embodiment, the IPDT component may search for available target devices within a local area network based on the registered devices and/or communications channels in the source devices communication stack in the hardware database, and/or based on zero...
configuration protocols, and/or based on user submitted target information, as will be further illustrated in Figure 3B.

In one embodiment, the source device may then provide the user with a list of potential tether targets from which the user may make a selection, as illustrated in one implementation in screen 373 of Figure 3D. Upon selecting a target device, the source device may then connect with the selected tether target based on the type of the tether target, as illustrated in one implementation in screens 375 and 378 of Figure 3D and will be further illustrated in Figure 3C. Once the IPDT is established, the IPDT may attempt to incorporate additional clients to facilitate IPDT, otherwise the source and target devices may proceed with the IPDT (as illustrated in one implementation in screen 380 of Figure 3D). This type of IPDT may be extended to facilitate Co-play IPDT as described below in the example implementations of Figures 5A-5D wherein target devices assume Server responsibilities for the illustrated Co-play IPDT implementations.

Figure 3B illustrates aspects of querying for tether targets within embodiments of the IPDT operation. In one embodiment, if a user submit an indication of target device 315, the IPDT may search for target device based on the user submitted information 316. For example, in one implementation, a user may enter the known IP address, MAC address, acronym, hardware label, digital signature, driver certificate, zero-configuration information, and/or the like of a target device via a mobile device, and the mobile device may search within an available range of a local area network for the corresponding indications. Otherwise, if there is no user submitted indications, the IPDT may determine whether to query the communication stack in the hardware.
database 318. For example, in one implementation, the IPDT may display a message to a user and receive an indication from the user to determine whether to query the database. If yes, the IPDT may form a query on the communication stack for registered devices/communication channels that are compatible with the source device 320. Otherwise, if not, the IPDT may start a search to locate suitable target devices within a range of local area network by zero configuration protocols 322, such as, but not limited to Service Location Protocol (SLP), Universal Plug and Play (UPnP), Jini, Bluetooth Service Discovery Protocol, WS-Discovery, Proprietary Discovery Protocol using UDP, Bonjour and/or the like.

[0 0 3 9 ] In one embodiment, the tether target device may broadcast its availability and publish a server via a zero-configuration network, e.g. an Apple SDK may create Bonjour service. For example, in one implementation, the C++ implementation based on Bonjour for creating a service and publishing the tether target may take a form similar to:

//Creating the Bonjour Service
CFNetService netService = CFNetServiceCreate(NULL, CFSTR(""), type, name, port);
...

// Publishing the Server
void init (CFNetServiceRef netService)
{
    CFStreamError error;
    CFNetServiceClientContext context = { o, NULL, NULL, NULL, NULL };
    CFNetServiceSetClient(netService, registerCallback, &context);
    CFNetServiceScheduleWithRunLoop(netService, CFRunLoopGetCurrent(),
    kCFRunLoopCommonModes);
    CFNetServiceRegister(netService, NULL);
    if (CFNetServiceRegister(netService, &error) == false) {

For another example, the C/C++ implementation based on Proprietary Discovery Protocol using UDP for creating a service and broadcasting service information as a tether target, may take a form similar to:

```c
//Broadcast Service Information
...
SOCKET sock;
sock = socket(AF_INET,SOCK_DGRAM,o);
char broadcast = '1';
int sres =
setsockopt(sock,SOL_SOCKET,SO_BROADCAST,&broadcast,sizeof(broadcast))
if (sres < 0)
{
    closesocket(sock);
    return 0;
}
struct sockaddr_in recv;
int len = sizeof(struct sockaddr_in);
char msg[]="IPDT PC V1.01";
recv.sin_family = AF_INET;
recv.sin_port = htons(MYPORT);
recv.sin_addr.s_addr = INADDR_BROADCAST;
sendto(sock,sendMSG,strlen(msg)+i,o,(sockaddr *)&recv,sizeof(recv));
```

...
In one embodiment, if such query returns a null result, the IPDT may display a message to the user indicating the search is unsuccessful, and may proceed with 310 upon user request. If the query returns at least one result, the IPDT may generate a list of available target devices, and proceed with 330.

Figure 3C illustrates aspects of connecting with tether target within embodiments of the IPDT operation. The IPDT may determine whether the target is a physical tether. If the target device is connected via a physical tether, for example, a direct audio/video connection such as an iPhone Video AV to Dock Connector Cable that is plugged directly into a large television, then the IPDT application may begin communicating data associated with the display of the source device to the larger remote display device via the physical tether, and the application may continue to execute taking advantage of a larger display. In an alternative embodiment, the physical tether may be a Casio XJ-S57 with a WiFi YW-2 adapter connected via WiFi. In one embodiment, the source device may transmit data to the target device through the physical tether in a variety of formats, such as, but not limited to Component Video, S-Video, HDMI, VGA, DVI, DisplayPort and/or the like.

In one embodiment, if the target is not a physical tether, the IPDT may determine whether the target is a processor-based device/entity. If not, the IPDT may display a message to the user indicating the tether is unsuccessful and proceed with 330 upon user request. In one implementation, if the remote device is a processor-based device, the source device may discern if the target device is capable of executing complementary application code as the remote device - for example: if the remote device is running a distributed object oriented application.
device is running a web server 352, if the remote device is running a remote display client (e.g., Citrix, VNC, etc.) 353, or a web browser 354. As already discussed, if the device is broadcasting based on a zero-configuration protocol (e.g., Bonjour, SLP, UPnP, Jini, Bluetooth Service Discovery Protocol, WS-Discovery, Proprietary Discovery Protocol using UDP, and/or the like) or responding to a proprietary communication request, then the tether target is processor-based,

In one embodiment, if the target computer is running a web browser 354, the source device may run a web server. For example, if the source device is an iPhone running UNIX, an Apache information server may be made part of the IPDT component, and that information server may be instantiated. At that point, the source device can provide an IP address and/or register itself via zero configure network protocols like Bonjour, and a user may use the web browser on the target machine to navigate to the web server on the source device. In one embodiment, an iPhone would be the source device and run an Apache web server, and a user at a target device, e.g., an laptop, would navigate to the IP address of the web server on the iPhone over a WiFi connection. At this point a custom communication channel has been established between the source and target devices 347, and the source device may provide instructions via its web server to instantiate the remote target, e.g., provide applications executable by the target web browser 348. At this point, the source application may engage the IPDT application 349 and use the custom application channel as a conduit to render its source display onto the remote display. For example, the source device web server may have a Java and/or Flash applet that it will provide to the target device which provides the target device with the capability of remote display viewing. As another example, a TightVNC Java Viewer applet may be downloaded by the target web browser
from the remote iPhone Apache web server and instantiated with configuration (e.g., with the IP address, set user/password keys, etc.) to connect to a VNC server that has been launched on the source device. In this manner, the target devices web browser becomes the remote display for the source device. In one embodiment, the source devices web server may instruct the target devices web browser to expand its window to the full size of the screen, thus providing an enlarged viewing area effectively mirroring the display on the source device.

In one embodiment, if the target device already has a remote desktop sharing component such as Visual Network Computing (VNC), Apple's remote display technology reachable via Bonjour, and/or the like, the user may be prompted with a screen instructing them on what to enter on the target device to complete the connection (e.g., IP address, user/password, help to access proper areas of the operating system, etc.) and configure a custom communications channel. In one embodiment, the source device will launch a VNC server as part of the IPDT component which will be viewable in the shared area of the Apple OS X Finder application, and may be selected for screen sharing, thereby instantiating the remote target to project the source device's display to the target. At this point, the IPDT application may continue to execute. For example, in one implementation, a C++ implementation for an iPhone to connect to a service based on Bonjour may take a form similar to:

```
// Connecting to a Service
NSNetService *service;
NSInputStream *istream = nil;
NSOutputStream *ostream = nil;
[service getInputStream:&istream outputStream:&ostream];
if (istream && ostream)
```
In one embodiment, if the target device is running either a custom application and/or web server configured to communicate with source devices, the source device has a number of channels over which it may communicate. In the case of distributed object oriented application, object oriented method calls may be sent to the target application to establish a connection. These custom applications may employ similar remote display technologies as already discussed. In another embodiment, where a source application uses a graphics rendering engine such as Open GL, Flash, and/or Apple's OS X development SDK, where such graphics libraries scale depending on a device's display abilities, the source device may be used as an input device, and the target device may run a more robust version of IPDT application stored on the source device.

The more robust version of the IPDT application may be provided to the target device in a number of ways. In one embodiment, the IPDT source application has a directory having multiple versions of the application, and may transfer more elaborate versions to the target device. In another embodiment, the IPDT source application will have a web link and have the target device automatically download the more elaborate applet and/or application and have that installed and instantiated. Upon the instantiation of the target version of the IPDT application, the source IPDT application will provide only user input signals which will be sent via the communication channel to the instantiated remote target where the remote application will interpret those instructions and execute the application in a more robust manner. For example, a
game using an iPhone as a controller, e.g., iGolf, may have a more robust version of iGolf loaded on a target laptop, and as such, take the accelerometer inputs of an iPhone source device to direct the execution of the game on the target device.

**[0051]** FIGURE 3D-3F provide examples of screenshots from an Apple iPhone illustrating aspects of an IPDT gaming application within an IPDT operation. In Figure 3D, a user may be provided options to launch an IPDT gaming application 370, and upon choosing the "competition", the user may be presented a list of gaming connection options 372. In one embodiment, if the user chooses "Bluetooth", the IPDT may search for another iPhone or other tether target devices via WiFi 373, and present a list of available tether target to the user 375. The user may then select a tether target 378 and connect with the tether target 380. Upon establishing connection with the tether target (in this case another iPhone), the IPDT application may be engaged on both the source device and the tether target 382 and 383. In one implementation, the IPDT application may facilitate two or more users to co-play a game, as illustrated in 383 and discussed in further detail in Figures 5A-5D and throughout the disclosure.

**[0052]** In one embodiment, as illustrated in Figure 3E, if the user chooses "lobby list" in the "competition" menu 372, the IPDT may provide a list of available "lobbies" on a gaming host server 374. Upon user's selection of a lobby, the IPDT may connect to a host server 376/378 and load the video game on the user device 382/383. In another embodiment, if the user selects the " In another embodiment, as illustrated in Figure 3F, if the user selects "market" at 370, the IPDT application may provide gaming features for sale 386. In another embodiment, if the user selects "social" at 370, the
IPDT application may provide an option to publish the gaming feeds of the user (e.g. through Facebook, etc.) 388.

In one embodiment, when IPDT application is engaged, the source device may transmit control information and data payloads to the target device in a variety of data formats. FIGURE 4 illustrates examples of data formats transmitted from the source device to the target device within embodiments of the IPDT operation. In one implementation, the source device may send data to a processor-based target device via a binary data packet 402, which includes fields such as message type 405, sequence number 406, acknowledgement number 407, data offset 408, data length 409, checksum 410, options 411 and data payload 412 and/or the like. The data 412 may include, but not limited to accelerometer information, pointer coordinates (pressing on the screen), images, GPS information, user information, and/or the like. For example, in one implementation, the data 412 may take a form similar to, a 64-byte user information field, including user ID, device ID etc., a 96-byte accelerometer information, including 3-dimentional coordinates (x,y,z), a 64-byte latitude/longitude value, a 64-byte pointer coordinate value (x,y), a 32-byte image length value, and data with variable lengths such as image raw data, video time information, and/or the like. In one embodiment, the image raw data may be generated and sent in compliance with the VNC video transport data format and/or a pointer to separate VNC video transport data stream. In one embodiment, the pointer may point to a .vnc file on the file system, or may incorporate the contents of such a file such that it specifies host, port, password, options (e.g., bit depth, mouse settings, scale, emulation, clipboard, etc.), and extensions such as sessions. In one embodiment, Remote Framebuffer Protocol (RFB) may be used as may be seen in: The Remote Framebuffer Protocol, Tristan Richardson (RealVNC Ltd),

In another implementation, the source device and the target device may exchange data via a Common Object Request Broker Architecture (CORBA) mechanism. For example, the source device and the target device may define a series of objects, such as, but not limited to accelerometer which may contain information regarding accelerometer status, GPS which may contain information about the device location, pointer which contains information about the user interaction on the screen, screen which contains the screen stream, and/or other data structure which may contain various data streams and constructs pertaining to a given application. In one implementation, the source/client device may interface with object reference generated stub code running on an object requested broker, and communicate with the target device via a network connection. The target device may include object implementation, generated skeleton code running on the target device side object requested broker. For example, in one implementation, the C++ implementation for defining data object under CORBA may take a form similar to the following:

```
//Define Accelerometer Data Object

class AccelerometerDataService : public PortableServer::RefCountServantBase
{
    public:
        AccelerometerDataServiceO;
    virtual ~AccelerometerDataServiceO;
    virtual CORBA::Boolean Update( CORBA::Long x,
```
CORBA::Boolean AccelerometerDataServiceUpdate (  
    CORBA::Long x,  
    CORBA::Long y,  
    CORBA::Long z)  
{
    return true;
}

[0055] In one example, the C++ implementation for connecting to the server under CORBA may take a form similar to the following:

//Connecting to the Server
...
CORBA_ORB_var orb=CORBA_ORB_init(argc, argv);  
const char* refFile="AccelerometerDataService.ref' ;  
ifstream in;  
in.open(refFile);  
CORBA_Object_var obj=orb->string_to_object(s);  
AccelerometerDataServiceads = AccelerometerDataService::_narrow(obj);  
ads->update(x,y,z);
...

[0056] FIGURE 5A illustrates an example implementation of an interactive proximity display tether with co-play (hereinafter "Co-play IPDT") wherein source device 550 establishes a communication path with target device 551 across communications network 555. In some implementations, target device 351 may be connected via cable 557 to display device 553 and configured to push the target display data to the display device. Otherwise, target device 551 may display Co-play IPDT data
on the target's display or in combination with the display device 553 as a dual monitor implementation.

[0057] In one embodiment, depending on the processing capabilities of target 551, source device 550 may drive communications/processing implementing server functionality, while pushing data for display to the target device 551 which would facilitate client functionality for Co-play IPDT. Alternately, source device 550 may be able to offload data processing functionality to target device 551, which could facilitate server functionality. In this event, source device 550 would facilitate client functionality, whereas target device 551 would facilitate Server functionality for Co-play IPDT.

[0058] In one embodiment, the second mobile device - client device 552 may be incorporated into the Co-play IPDT. In an embodiment, the communication path for client device 552 may be established across a communications network 556 with source device 550. Once the communications path is established, client device may be configured to sense user manipulation of the device and transmit the data across the IPDT to source device 550. If source device 550 is facilitating server functionality, source device 550 may process the data from client device 552 and forward to the target device 551 for display. On the other hand, if source device 550 is facilitating client functionality with target device 551 facilitating server functionality, source device 550 may act simply as a conduit to relay the user manipulation data from client device 552. In one embodiment, the Co-play IPDT may be configured to facilitate client device 552 and source device interacting in persistent platform gameplay that is displayed by target device 551 or display device 553. For example, in one implementation, the user
interactions with respective devices 550 and 552 may be displayed in realtime in an iGolf game, represented by first and second avatars 558 and 559, respectively.

[0059] Figure 5B illustrates another implementation of Co-play IPDT. In one embodiment, the source device 560 and target device 561 establish a communication path across communications network 565. Source device 560 and target device 561 also negotiate which device will facilitate server functionality and which device facilitate server functionality as described above. However, instead of client device 562 establishing a communication path and communicating with the Co-play IPDT through source device 560, client device 562 establishes a communication path and communicates with the Co-play IPDT through target device 561 across communications network 366 after source/target device client/server functionality has been established. Accordingly, the Co-play IPDT in Figure 5B will facilitate client device 562 and source device interacting in persistent platform gameplay that is displayed by target device 561 or display device 563. For example, the user interactions with respective devices 560 and 562 may be displayed in realtime in an iGolf game, represented by first and second avatars 568 and 569, respectively.

[0060] FIGURE 5C illustrates an alternate embodiment of Co-play IPDT. Source device 570 and target device 571 establish the server/client functionality across communications network 574 as discussed above. However, client device 572-A may join the Co-play IPDT by first connecting with client device 572-B across communications network 576 which in turn may join with target device 371 to achieve Co-play IPDT. In one embodiment, target device 571 facilitates server functionality and as such, drives the processing of user inputs from source device 570, and client devices
Once the client device data is processed by target device 571, target device 571 may display persistent platform gameplay on its display and/or via a tethered display 574A. In some instances, target device 571 may also transmit display data back to client device 572B for display on the client device 572B or tethered display 573B. For example, the user interactions with respective devices 570 and 572A may be displayed as persistent platform gameplay on display 573-A/573B in realtime in an iGolf game, represented by first and second avatars 578 and 579, respectively.

FIGURE 5D illustrates another embodiment of Co-play IPDT similar to the implementation illustrated in FIG 5C. In one embodiment, target device 581 facilitates client functionality instead of server functionality, which is facilitated by source device 580 in this implementation. As such, source device 580 receives and processes user interaction data from source device 580, as well as client device 582A via communication networks 584, 585, and 586 communicated through client devices 582B and 581. Once received, source device 580 processes and serves the processed user interaction data back to target 581 and client 582B for display. For example, the user interactions with respective devices 580 and 582A may be displayed as persistent platform gameplay on displays 583-A and/or 583B in realtime in an iGolf game, represented by first and second avatars 588 and 589, respectively.

FIGURE 6A shows an implementation of overall logic flow in one embodiment of Co-play IPDT operation. The logic flow shown is directed to an embodiment of the Co-play IPDT employing remote access software, such as Citrix or VNC, to couple a mobile source device with a target device as a remote display, as well as available client devices. A user may engage a mobile source device application 601, such
1. As by turning on the mobile device, selecting an application icon, and/or the like. In one
2. implementation, both an application and remote access software may be engaged at the
3. mobile source device. A remote display target device application may also be engaged
4. on the mobile device, selecting an application (e.g., remote
5. access software), and/or the like. The mobile source device and/or the remote display
6. target device may check for a data link to the counterpart device, such as by pinging the
7. counterpart device and a determination may be made as to whether a data link has been
8. established. If not, then the mobile source device and/or the remote display target
9. device may retry establishing a data link, such as by repairing and/or refreshing a
10. network connection, presenting an error message via a user interface and requesting
11. that the user check communication components, and/or the like.

[0063] Once the source device and target establish a viable communications path,
the Co-play 1PDT may check whether additional client devices are available to be
incorporated into the Co-play IPDT. If so, the communication paths are established
607-610. Otherwise, sensor components may receive sensor inputs that may cause
user interaction updates to a mobile source device application configuration 622. For
example, in one implementation, a sensor input may comprise a state of motion, such as
may be detected by an accelerometer. This state of motion may be translated into a
virtual state of motion for an avatar or other virtual entity in the context of a mobile
device application, such as a video game. A determination may be made at 625 as to
whether or not a send period has concluded. If not, then the user's source/client device
may wait for a period of time 628 and check whether any new inputs have been received
from the sensors 631. If so, then the user device may add the most recent set of sensor
inputs and/or corresponding application configurations to a display signal cache 634
and proceed to receive the new sensor inputs 619. Otherwise, the user's source/client device may return to 625 to check whether the send period has concluded. Once a send period has concluded, the user's source/client device may configure a display signal message corresponding to a current and/or set of recent application states 638. For example, in one implementation, the display signal may represent a current and/or set of recent positions and/or states of motions of an avatar in a video game.

[0064] In implementation's where the source device facilitates server functionality, the user's source/client device may update remote access software at the source device, such as Citrix, VNC, and/or the like 641, and send the corresponding display signal to the remote display target device 644. The remote display on target device 644, in turn, may provide a visualization of the display signal, such as on a display screen 647. A determination may be made as to whether to continue with Co-play IPDT operation 650. If so, then the user's source/client device transition to receive further sensor inputs 619. Otherwise, the Co-play IPDT is completed 653.

[0065] FIGURE 6B shows an implementation of overall logic flow in another embodiment of Co-play IPDT operation. The logic flow shown in Fig. 6B is directed to an embodiment of the user's source/client device employing web server software to couple a mobile source device to a remote display target device, such as by publishing mobile device application data to a web site for retrieval by an internet-connected target device. A user may engage a user's source/client device application 656, such as by turning on the mobile source/client device, selecting an application icon, and/or the like. In one implementation, both an application and web server software may be engaged at the mobile device. The user's source/client device's sensor components may
receive sensor inputs 659 that may cause updates to a mobile device application configuration 662. A determination may be made at 665 as to whether or not a web client data request, such as an HTTP request, has been received. If not, then the user's source/client device may wait for a period of time 668 and check whether any new inputs have been received from the sensors 672. If so, then the user's source/client device may add the most recent set of sensor inputs and/or corresponding application configurations to a display signal cache 675 and proceed to receive the new sensor inputs 659. Otherwise, the user's source/client device may return to 665 to check whether a web client data request has been received. In an alternative implementation, the user's source/client device may send display signals to a target device without regard to whether a web client data request has been received or not. In this implementation, the user's source/client device may send display signals to an intermediary repository. The repository, then, may monitor the receipt of web client data requests and provide one or more current or historical display signals to the web client supplying the data request.

[0066] Once a web client data request has been received, the user's source/client device may configure a display signal message corresponding to a current and/or set of recent application states 678. The configured display signal may be published on the World Wide Web via web server software 681. The published application state content may then be accessed by one or more remote display target devices, such as by accessing a web site on which the application state data is hosted 684. In one implementation, application state data may be represented on a website as an integrated video, animation, rich Internet application, and/or the like configured in accordance with a multimedia plug-in such as Adobe Flash. A determination may be made at 687 as to
whether to update the displayed visualization of the application state at the remote
display. If so, then additional sensor inputs are received at 659. Otherwise, the Co-play
IPDT session concludes 690.

[0067] FIGURE 7A is of a block diagram illustrating an Co-play IPDT payment
model. In one embodiment, a central service 705 may provide offering applications 715
via a subscription 710 or on any individual purchase basis. For example, offerings that
otherwise may need to be purchased for a set price (e.g., $1.99 for each application) via a
service such as the Apple App Store or may otherwise be accessed via a subscription
service. In one embodiment, a service application may be downloaded onto the source
device. Service applications once purchased may act to download games/offerings 720
free of charged for specified periods of time based on key sets. For example, the
purchase date of the service application 705 may be used as a basis of providing
offerings 715. For example, if the service application (e.g., iPlay) is purchased via the
App Store on January 1, 2009, where the version of iPlay purchased is a 1 year
subscription (e.g., for $19.99), then any offerings obtained up until January 1, 2010 will
work accessing an authorization key within the iPlay application, that key expiring at the
end of the year.

[0068] FIGURE 7B is of a block diagram illustrating an Co-play IPDT screens. In
one embodiment, the subscription model may be wrapped in its own application 725
and execute as a full application on the source device 727 including icons and splash
screens. Offerings, e.g., iTennis, 730 in a subscription model will work as full
applications with instructions 740 and game screens. Locked versions of the offerings
may have a zippered appearance when out of subscription, which may be unlocked under a renewed subscription.

Interactive Extensions

Another advantage to the Co-play IPDT involves features that may extend interaction with applications. Such extensions are particularly useful in the areas of electronic gaming. For example, when a source device, e.g., an iPhone, is tethered to a target device, e.g., a large screen computer, the source device may then become a game controller input device relaying game data/instructions to the target device as it casts its display information across the communications tether. For example, a user may use an iPhone as an analogue to a tennis racquet, playing a virtual game of tennis via an iTennis application offering that is facilitated with an IPDT component. Beyond using the iPhone as a tennis racquet handle for swinging the racquet, where the iPhones 3D accelerometers may be measured to obtain the dynamics of the string, the IPDT can further extend features by employing the touch screen. For example, in one embodiment, a user can pinch the virtual tennis strings on the racquet together as they swing the iPhone to get an extra power boost in the game. Further, a user may use multitouch gestures to tighten or loosen the strings on the racquet, which will be used as additional input for the IPDT application and will affect the dynamics of the game (e.g., loosened strings increasing power but decreasing accuracy).

Additionally, the Co-play IPDT may be used to house user (e.g., game) profiles and progress. In this manner, user avatars and accounts such as (e.g., iMee) may be accessed on the source device even if the target device is unable and/or has no capacity to connect to the internet or otherwise gain access to the users profiles. In one
embodiment, the profiles are mirrored and/or cached onto the source device, e.g., the iPhone, from a service on the internet. These settings may be transferred to the target device and/or accessed across the tether. Furthermore, such avatars and settings may be used to interact with other IPDT application offerings. For example, two or more source devices, e.g., iPhones, from two different users may be simultaneously tethered onto a single target device and allowed to share a singular application space on the target application. For example, one of the two source devices can act as a host application and accept communication with the other source device. Thus, in one embodiment, where each user has a virtual avatar, the two avatars may be controlled in a common space on the target display simultaneously. Further, such an impromptu target virtual space may be the source for transactions between the parties: e.g., the avatars may trade valuable digital assets such as digital cash for digital objects (e.g., gold coins for enhanced game weapons and/or devices). In one embodiment, a common multiplayer display housing all the source, e.g., iPhone, players may be seen on the single target device, but the displays of the source devices may have another view that is private to each users. In such an embodiment, users may interact in a common area on the target display and engage in secondary and/or private/secret strategic activities on their own personal displays.

[0071]  In another embodiment, a single target display, like a large computer display or television would allow two or more players play high action games like ping-pong or tennis. In another embodiment, the IPDT allows the source device to turn most any target device into an impromptu presentation display device.
FIGURE 7C shows aspects of different applications of a Co-play IPDT in one embodiment. These may include, but are not limited to, games such as golf, bowling, billiards, baseball, shuffleboard, fishing, and/or the like.

FIGURES 8 show implementations of a Co-play IPDT housing configured as a golf club in one embodiment. A mobile device such as an Apple iPhone may be secured in a housing shaped as a gaming implement, such as a golf club for a golf game, tennis racket for tennis game, fishing pole for a fishing game, baseball bat for a baseball game, and/or the like.

IPDT Controller

FIGURE 9 illustrates inventive aspects of an IPDT controller 901 in a block diagram. In this embodiment, the IPDT controller 901 may serve to aggregate, process, store, search, serve, identify, instruct, generate, match, and/or facilitate interactions with a computer through network technologies, and/or other related data.

Typically, users, which may be people and/or other systems, may engage information technology systems (e.g., computers) to facilitate information processing. In turn, computers employ processors to process information; such processors 903 may be referred to as central processing units (CPU). One form of processor is referred to as a microprocessor. CPUs use communicative circuits to pass binary encoded signals acting as instructions to enable various operations. These instructions may be operational and/or data instructions containing and/or referencing other instructions and data in various processor accessible and operable areas of memory 929 (e.g.,
registers, cache memory, random access memory, etc.). Such communicative
instructions may be stored and/or transmitted in batches (e.g., batches of instructions)
as programs and/or data components to facilitate desired operations. These stored
instruction codes, e.g., programs, may engage the CPU circuit components and other
motherboard and/or system components to perform desired operations. One type of
program is a computer operating system, which, may be executed by CPU on a
computer; the operating system enables and facilitates users to access and operate
computer information technology and resources. Some resources that may employed in
information technology systems include: input and output mechanisms through which
data may pass into and out of a computer; memory storage into which data may be
saved; and processors by which information may be processed. These information
technology systems may be used to collect data for later retrieval, analysis, and
manipulation, which may be facilitated through a database program. These information
technology systems provide interfaces that allow users to access and operate various
system components.

[0077] In one embodiment, the IPDT controller 901 may be connected to and/or
communicate with entities such as, but not limited to: one or more users from user
input devices 911; peripheral devices 912; an optional cryptographic processor device
928; and/or a communications network 913.

[0078] Networks are commonly thought to comprise the interconnection and
interoperation of clients, servers, and intermediary nodes in a graph topology. It should
be noted that the term "server" as used throughout this application refers generally to a
computer, other device, program, or combination thereof that processes and responds to
the requests of remote users across a communications network. Servers serve their
information to requesting "clients." The term "client" as used herein refers generally to a
computer, program, other device, user and/or combination thereof that is capable of
processing and making requests and obtaining and processing any responses from
servers across a communications network. A computer, other device, program, or
combination thereof that facilitates, processes information and requests, and/or
furthers the passage of information from a source user to a destination user is
commonly referred to as a "node." Networks are generally thought to facilitate the
transfer of information from source points to destinations. A node specifically tasked
with furthering the passage of information from a source to a destination is commonly
called a "router." There are many forms of networks such as Local Area Networks
(LANs), Pico networks, Wide Area Networks (WANs), Wireless Networks (WLANs), etc.
For example, the Internet is generally accepted as being an interconnection of a
multitude of networks whereby remote clients and servers may access and interoperate
with one another.

The IPDT controller 901 may be based on computer systems that may
comprise, but are not limited to, components such as: a computer systemization 902
connected to memory 929.

Computer Systemization

A computer systemization 902 may comprise a clock 930, central
processing unit ("CPU(s)" and/or "processor(s)" (these terms are used interchangeable
throughout the disclosure unless noted to the contrary)) 903, a memory 929 (e.g., a read
only memory (ROM) 906, a random access memory (RAM) 905, etc.), and/or an
interface bus 907, and most frequently, although not necessarily, are all interconnected and/or communicating through a system bus 904 on one or more (mother)board(s) 902 having conductive and/or otherwise transportive circuit pathways through which instructions (e.g., binary encoded signals) may travel to effect communications, operations, storage, etc. Optionally, the computer systemization may be connected to an internal power source 986. Optionally, a cryptographic processor 926 may be connected to the system bus. The system clock typically has a crystal oscillator and generates a base signal through the computer systemization's circuit pathways. The clock is typically coupled to the system bus and various clock multipliers that will increase or decrease the base operating frequency for other components interconnected in the computer systemization. The clock and various components in a computer systemization drive signals embodying information throughout the system. Such transmission and reception of instructions embodying information throughout a computer systemization may be commonly referred to as communications. These communicative instructions may further be transmitted, received, and the cause of return and/or reply communications beyond the instant computer systemization to: communications networks, input devices, other computer systemizations, peripheral devices, and/or the like. Of course, any of the above components may be connected directly to one another, connected to the CPU, and/or organized in numerous variations employed as exemplified by various computer systems.

[0081] The CPU comprises at least one high-speed data processor adequate to execute program components for executing user and/or system-generated requests. Often, the processors themselves will incorporate various specialized processing units, such as, but not limited to: integrated system (bus) controllers, memory management
control units, floating point units, and even specialized processing sub-units like graphics processing units, digital signal processing units, and/or the like. Additionally, processors may include internal fast access addressable memory, and be capable of mapping and addressing memory beyond the processor itself; internal memory may include, but is not limited to: fast registers, various levels of cache memory (e.g., level 1, 2, 3, etc.), RAM, etc. The processor may access this memory through the use of a memory address space that is accessible via instruction address, which the processor can construct and decode allowing it to access a circuit path to a specific memory address space having a memory state. The CPU may be a microprocessor such as: AMD's Athlon, Duron and/or Opteron; ARM's application, embedded and secure processors; IBM and/or Motorola's DragonBall and PowerPC; IBM's and Sony's Cell processor; Intel's Celeron, Core (2) Duo, Itanium, Pentium, Xeon, and/or XScale; and/or the like processor(s). The CPU interacts with memory through instruction passing through conductive and/or transportive conduits (e.g., (printed) electronic and/or optic circuits) to execute stored instructions (i.e., program code) according to conventional data processing techniques. Such instruction passing facilitates communication within the IPDT controller and beyond through various interfaces. Should processing requirements dictate a greater amount speed and/or capacity, distributed processors (e.g., Distributed IPDT), mainframe, multi-core, parallel, and/or super-computer architectures may similarly be employed. Alternatively, should deployment requirements dictate greater portability, smaller Personal Digital Assistants (PDAs) may be employed.

Depending on the particular implementation, features of the IPDT may be achieved by implementing a microcontroller such as CAST'S R8051XC2 microcontroller;
Intel’s MCS 51 (i.e., 8051 microcontroller); and/or the like. Also, to implement certain features of the IPDT, some feature implementations may rely on embedded components, such as: Application-Specific Integrated Circuit ("ASIC"), Digital Signal Processing ("DSP"), Field Programmable Gate Array ("FPGA"), and/or the like embedded technology. For example, any of the IPDT component collection (distributed or otherwise) and/or features may be implemented via the microprocessor and/or via embedded components; e.g., via ASIC, coprocessor, DSP, FPGA, and/or the like.

Alternately, some implementations of the IPDT may be implemented with embedded components that are configured and used to achieve a variety of features or signal processing.

[0083] Depending on the particular implementation, the embedded components may include software solutions, hardware solutions, and/or some combination of both hardware/software solutions. For example, IPDT features discussed herein may be achieved through implementing FPGAs, which are a semiconductor devices containing programmable logic components called "logic blocks", and programmable interconnects, such as the high performance FPGA Virtex series and/or the low cost Spartan series manufactured by Xilinx. Logic blocks and interconnects can be programmed by the customer or designer, after the FPGA is manufactured, to implement any of the IPDT features. A hierarchy of programmable interconnects allow logic blocks to be interconnected as needed by the IPDT system designer/administrator, somewhat like a one-chip programmable breadboard. An FPGA’s logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or simple mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be
simple flip-flops or more complete blocks of memory. In some circumstances, the IPDT may be developed on regular FPGAs and then migrated into a fixed version that more resembles ASIC implementations. Alternate or coordinating implementations may migrate IPDT controller features to a final ASIC instead of or in addition to FPGAs. Depending on the implementation all of the aforementioned embedded components and microprocessors may be considered the "CPU" and/or "processor" for the IPDT.

Power Source

[0084] The power source 986 may be of any standard form for powering small electronic circuit board devices such as the following power cells: alkaline, lithium hydride, lithium ion, lithium polymer, nickel cadmium, solar cells, and/or the like. Other types of AC or DC power sources may be used as well. In the case of solar cells, in one embodiment, the case provides an aperture through which the solar cell may capture photonic energy. The power cell 986 is connected to at least one of the interconnected subsequent components of the IPDT thereby providing an electric current to all subsequent components. In one example, the power source 986 is connected to the system bus component 904. In an alternative embodiment, an outside power source 986 is provided through a connection across the I/O 908 interface. For example, a USB and/or IEEE 1394 connection carries both data and power across the connection and is therefore a suitable source of power.

Interface Adapters

[0085] Interface bus(ses) 907 may accept, connect, and/or communicate to a number of interface adapters, conventionally although not necessarily in the form of adapter cards, such as but not limited to: input output interfaces (I/O) 908, storage
interfaces 909, network interfaces 910, and/or the like. Optionally, cryptographic processor interfaces 927 similarly may be connected to the interface bus. The interface bus provides for the communications of interface adapters with one another as well as with other components of the computer systemization. Interface adapters are adapted for a compatible interface bus. Interface adapters conventionally connect to the interface bus via a slot architecture. Conventional slot architectures may be employed, such as, but not limited to: Accelerated Graphics Port (AGP), Card Bus, (Extended) Industry Standard Architecture ((E)ISA), Micro Channel Architecture (MCA), NuBus, Peripheral Component Interconnect (Extended) (PCI(X)), PCI Express, Personal Computer Memory Card International Association (PCMCIA), and/or the like.

Storage interfaces 909 may accept, communicate, and/or connect to a number of storage devices such as, but not limited to: storage devices 914, removable disc devices, and/or the like. Storage interfaces may employ connection protocols such as, but not limited to: (Ultra) (Serial) Advanced Technology Attachment (Packet Interface) ((Ultra) (Serial) ATA(PI)), (Enhanced) Integrated Drive Electronics ((E)IDE), Institute of Electrical and Electronics Engineers (IEEE) 1394, fiber channel, Small Computer Systems Interface (SCSI), Universal Serial Bus (USB), and/or the like.

Network interfaces 910 may accept, communicate, and/or connect to a communications network 913. Through a communications network 913, the IPDT controller is accessible through remote clients 933b (e.g., computers with web browsers) by users 933a. Network interfaces may employ connection protocols such as, but not limited to: direct connect, Ethernet (thick, thin, twisted pair 10/100/1000 Base T, and/or the like), Token Ring, wireless connection such as IEEE 802.na-x, and/or the like.
1 like. Should processing requirements dictate a greater amount speed and/or capacity, 2 distributed network controllers (e.g., Distributed IPDT), architectures may similarly be 3 employed to pool, load balance, and/or otherwise increase the communicative 4 bandwidth required by the IPDT controller. A communications network may be any one 5 and/or the combination of the following: a direct interconnection; the Internet; a Local 6 Area Network (LAN); a Metropolitan Area Network (MAN); an Operating Missions as 7 Nodes on the Internet (OMNI); a secured custom connection; a Wide Area Network 8 (WAN); a wireless network (e.g., employing protocols such as, but not limited to a 9 Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. A 10 network interface may be regarded as a specialized form of an input output interface. 11 Further, multiple network interfaces may be used to engage with various 12 communications network types. For example, multiple network interfaces may be 13 employed to allow for the communication over broadcast, multicast, and/or unicast 14 networks.

[0088] Input Output interfaces (I/O) may accept, communicate, and/or 16 connect to user input devices, peripheral devices, cryptographic processor 17 devices, and/or the like. I/O may employ connection protocols such as, but not 18 limited to: audio: analog, digital, monaural, RCA, stereo, and/or the like; data: Apple 19 Desktop Bus (ADB), IEEE 1394a-b, serial, universal serial bus (USB); infrared; joystick; 20 keyboard; midi; optical; PC AT; PS/2; parallel; radio; video interface: Apple Desktop 21 Connector (ADC), BNC, coaxial, component, composite, digital, Digital Visual Interface 22 (DVI), high-definition multimedia interface (HDMI), RCA, RF antennae, S-Video, VGA, 23 and/or the like; wireless: 802.na/b/g/n/x, Bluetooth, code division multiple access 24 (CDMA), global system for mobile communications (GSM), WiMax, etc.; and/or the
like. One typical output device may include a video display, which typically comprises a Cathode Ray Tube (CRT) or Liquid Crystal Display (LCD) based monitor with an interface (e.g., DVI circuitry and cable) that accepts signals from a video interface, may be used. The video interface composites information generated by a computer systemization and generates video signals based on the composites information in a video memory frame. Another output device is a television set, which accepts signals from a video interface. Typically, the video interface provides the composites video information through a video connection interface that accepts a video display interface (e.g., an RCA composite video connector accepting an RCA composite video cable; a DVI connector accepting a DVI display cable, etc.).

[0089] User input devices 911 may be card readers, dongles, fingerprint readers, gloves, graphics tablets, joysticks, keyboards, mouse (mice), remote controls, retina readers, trackballs, trackpads, and/or the like.

[0090] Peripheral devices 912 may be connected and/or communicate to I/O and/or other facilities of the like such as network interfaces, storage interfaces, and/or the like. Peripheral devices may be audio devices, cameras, dongles (e.g., for copy protection, ensuring secure transactions with a digital signature, and/or the like), external processors (for added functionality), goggles, microphones, monitors, network interfaces, printers, scanners, storage devices, video devices, video sources, visors, and/or the like.

[0091] It should be noted that although user input devices and peripheral devices may be employed, the IPDT controller may be embodied as an embedded, dedicated,
and/or monitor-less (i.e., headless) device, wherein access would be provided over a network interface connection.

[0092] Cryptographic units such as, but not limited to, microcontrollers, processors 926, interfaces 927, and/or devices 928 may be attached, and/or communicate with the IPDT controller. A MC68HC16 microcontroller, manufactured by Motorola Inc., may be used for and/or within cryptographic units. The MC68HC16 microcontroller utilizes a 16-bit multiply-and-accumulate instruction in the 16 MHz configuration and requires less than one second to perform a 512-bit RSA private key operation. Cryptographic units support the authentication of communications from interacting agents, as well as allowing for anonymous transactions. Cryptographic units may also be configured as part of CPU. Equivalent microcontrollers and/or processors may also be used. Other commercially available specialized cryptographic processors include: the Broadcom's CryptoNetX and other Security Processors; nCipher's nShield, SafeNet's Luna PCI (e.g., 7100) series; Semaphore Communications' 40 MHz Roadrunner 184; Sun's Cryptographic Accelerators (e.g., Accelerator 6000 PCIe Board, Accelerator 500 Daughtercard); Via Nano Processor (e.g., L2100, L2200, U2400) line, which is capable of performing 500+ MB/s of cryptographic instructions; VLSI Technology's 33 MHz 6868; and/or the like.

Memory

[0093] Generally, any mechanization and/or embodiment allowing a processor to affect the storage and/or retrieval of information is regarded as memory 929. However, memory is a fungible technology and resource, thus, any number of memory embodiments may be employed in lieu of or in concert with one another. It is to be
I understood that the IPDT controller and/or a computer systemization may employ various forms of memory 929. For example, a computer systemization maybe configured wherein the functionality of on-chip CPU memory (e.g., registers), RAM, ROM, and any other storage devices are provided by a paper punch tape or paper punch card mechanism; of course such an embodiment would result in an extremely slow rate of operation. In a typical configuration, memory 929 will include ROM 906, RAM 905, and a storage device 914. A storage device 914 may be any conventional computer system storage. Storage devices may include a drum; a (fixed and/or removable) magnetic disk drive; a magneto-optical drive; an optical drive (i.e., Blueray, CD ROM/RAM/Recordable (R)/ReWritable (RW), DVD R/RW, HD DVD R/RW etc.); an array of devices (e.g., Redundant Array of Independent Disks (RAID)); solid state memory devices (USB memory, solid state drives (SSD), etc.); other processor-readable storage mediums; and/or other devices of the like. Thus, a computer systemization generally requires and makes use of memory.

Component Collection

[oo94] The memory 929 may contain a collection of program and/or database components and/or data such as, but not limited to: operating system component(s) 915 (operating system); information server component(s) 916 (information server); user interface component(s) 917 (user interface); Web browser component(s) 918 (Web browser); database(s) 919; mail server component(s) 921; mail client component(s) 922; cryptographic server component(s) 920 (cryptographic server); the IPDT component(s) 935; and/or the like (i.e., collectively a component collection). These components may be stored and accessed from the storage devices and/or from storage devices accessible
through an interface bus. Although non-conventional program components such as those in the component collection, typically, are stored in a local storage device 914, they may also be loaded and/or stored in memory such as: peripheral devices, RAM, remote storage facilities through a communications network, ROM, various forms of memory, and/or the like.

Operating System

[0095] The operating system component 915 is an executable program component facilitating the operation of the IPDT controller. Typically, the operating system facilitates access of I/O, network interfaces, peripheral devices, storage devices, and/or the like. The operating system may be a highly fault tolerant, scalable, and secure system such as: Apple Macintosh OS X (Server); AT&T Plan 9; Be OS; Unix and Unix-like system distributions (such as AT&T's UNIX; Berkley Software Distribution (BSD) variations such as FreeBSD, NetBSD, OpenBSD, and/or the like; Linux distributions such as Red Hat, Ubuntu, and/or the like); and/or the like operating systems. However, more limited and/or less secure operating systems also may be employed such as Apple Macintosh OS, IBM OS/2, Microsoft DOS, Microsoft Windows 2000/2003/3.1/95/98/CE/Millenium/NT/Vista/XP (Server), Palm OS, and/or the like.

An operating system may communicate to and/or with other components in a component collection, including itself, and/or the like. Most frequently, the operating system communicates with other program components, user interfaces, and/or the like. For example, the operating system may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses. The operating system, once executed by the CPU, may enable the
interaction with communications networks, data, I/O, peripheral devices, program components, memory, user input devices, and/or the like. The operating system may provide communications protocols that allow the IPDT controller to communicate with other entities through a communications network 913. Various communication protocols may be used by the IPDT controller as a subcarrier transport mechanism for interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast, and/or the like.

Information Server

[0096] An information server component 916 is a stored program component that is executed by a CPU. The information server may be a conventional Internet information server such as, but not limited to Apache Software Foundation's Apache, Microsoft's Internet Information Server, and/or the like. The information server may allow for the execution of program components through facilities such as Active Server Page (ASP), ActiveX, (ANSI) (Objective-) C (++) , C# and/or .NET, Common Gateway Interface (CGI) scripts, dynamic (D) hypertext markup language (HTML), FLASH, Java, JavaScript, Practical Extraction Report Language (PERL), Hypertext Pre-Processor (PHP), pipes, Python, wireless application protocol (WAP), WebObjects, and/or the like. The information server may support secure communications protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), messaging protocols (e.g., America Online (AOL) Instant Messenger (AIM), Application Exchange (APEX), ICQ, Internet Relay Chat (IRC), Microsoft Network (MSN) Messenger Service, Presence and Instant Messaging Protocol (PRIM), Internet Engineering Task Force's (IETF's)
Session Initiation Protocol (SIP), SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE), open XML-based Extensible Messaging and Presence Protocol (XMPP) (i.e., Jabber or Open Mobile Alliance's (OMA's) Instant Messaging and Presence Service (IMPS)), Yahoo! Instant Messenger Service, and/or the like. The information server provides results in the form of Web pages to Web browsers, and allows for the manipulated generation of the Web pages through interaction with other program components. After a Domain Name System (DNS) resolution portion of an HTTP request is resolved to a particular information server, the information server resolves requests for information at specified locations on the IPDT controller based on the remainder of the HTTP request. For example, a request such as http://123.124.125.126/mylnformation.html might have the IP portion of the request "123.124.125.126" resolved by a DNS server to an information server at that IP address; that information server might in turn further parse the http request for the "/myInformation.html" portion of the request and resolve it to a location in memory containing the information "myInformation.html." Additionally, other information serving protocols maybe employed across various ports, e.g., FTP communications across port 21, and/or the like. An information server may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the information server communicates with the IPDT database 919, operating systems, other program components, user interfaces, Web browsers, and/or the like.

[0097] Access to the IPDT database may be achieved through a number of database bridge mechanisms such as through scripting languages as enumerated below (e.g., CGI) and through inter-application communication channels as enumerated below.
(e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed through the bridge mechanism into appropriate grammars as required by the IPDT. In one embodiment, the information server would provide a Web form accessible by a Web browser. Entries made into supplied fields in the Web form are tagged as having been entered into the particular fields, and parsed as such. The entered terms are then passed along with the field tags, which act to instruct the parser to generate queries directed to appropriate tables and/or fields. In one embodiment, the parser may generate queries in standard SQL by instantiating a search string with the proper join/select commands based on the tagged text entries, wherein the resulting command is provided over the bridge mechanism to the IPDT as a query. Upon generating query results from the query, the results are passed over the bridge mechanism, and may be parsed for formatting and generation of a new results Web page by the bridge mechanism. Such a new results Web page is then provided to the information server, which may supply it to the requesting Web browser.

[0098] Also, an information server may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses.

User Interface

[0099] The function of computer interfaces in some respects is similar to automobile operation interfaces. Automobile operation interface elements such as steering wheels, gearshifts, and speedometers facilitate the access, operation, and display of automobile resources, functionality, and status. Computer interaction interface elements such as check boxes, cursors, menus, scrollers, and windows
1 (collectively and commonly referred to as widgets) similarly facilitate the access, operation, and display of data and computer hardware and operating system resources, functionality, and status. Operation interfaces are commonly called user interfaces.

4 Graphical user interfaces (GUIs) such as the Apple Macintosh Operating System's Aqua, IBM's OS/2, Microsoft's Windows

6 2000/2003/3. i/95/98/CE/Millenium/NT/XP/Vista/7 (i.e., Aero), Unix's X-Windows

7 (e.g., which may include additional Unix graphic interface libraries and layers such as K Desktop Environment (KDE), mythTV and GNU Network Object Model Environment

9 (GNOME)), web interface libraries (e.g., ActiveX, AJAX, (D)HTML, FLASH, Java,

0 JavaScript, etc. interface libraries such as, but not limited to, Dojo, jQuery(UI),

1 MooTools, Prototype, script.aculo.us, SWFObject, Yahoo! User Interface, any of which

2 may be used and) provide a baseline and means of accessing and displaying information

3 graphically to users.

A user interface component is a stored program component that is

executed by a CPU. The user interface may be a conventional graphic user interface as

provided by, with, and/or atop operating systems and/or operating environments such

as already discussed. The user interface may allow for the display, execution,

interaction, manipulation, and/or operation of program components and/or system

facilities through textual and/or graphical facilities. The user interface provides a facility

through which users may affect, interact, and/or operate a computer system. A user

interface may communicate to and/or with other components in a component

collection, including itself, and/or facilities of the like. Most frequently, the user

interface communicates with operating systems, other program components, and/or the

like. The user interface may contain, communicate, generate, obtain, and/or provide
program component, system, user, and/or data communications, requests, and/or responses.

A Web browser component is a stored program component that is executed by a CPU. The Web browser may be a conventional hypertext viewing application such as Microsoft Internet Explorer or Netscape Navigator. Secure Web browsing may be supplied with 128bit (or greater) encryption by way of HTTPS, SSL, and/or the like. Web browsers allowing for the execution of program components through facilities such as ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, web browser plug-in APIs (e.g., FireFox, Safari Plug-in, and/or the like APIs), and/or the like. Web browsers and like information access tools may be integrated into PDAs, cellular telephones, and/or other mobile devices. A Web browser may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the Web browser communicates with information servers, operating systems, integrated program components (e.g., plug-ins), and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, and/or responses. Of course, in place of a Web browser and information server, a combined application may be developed to perform similar functions of both. The combined application would similarly affect the obtaining and the provision of information to users, user agents, and/or the like from the IPDT enabled nodes. The combined application may be nugatory on systems employing standard Web browsers.
Mail Server

A mail server component 921 is a stored program component that is executed by a CPU 903. The mail server may be a conventional Internet mail server such as, but not limited to sendmail, Microsoft Exchange, and/or the like. The mail server may allow for the execution of program components through facilities such as ASP, ActiveX, (ANSI) (Objective-) C (++) , C# and/or .NET, CGI scripts, Java, JavaScript, PERL, PHP, pipes, Python, WebObjects, and/or the like. The mail server may support communications protocols such as, but not limited to: Internet message access protocol (IMAP), Messaging Application Programming Interface (MAPI)/Microsoft Exchange, post office protocol (POP3), simple mail transfer protocol (SMTP), and/or the like. The mail server can route, forward, and process incoming and outgoing mail messages that have been sent, relayed and/or otherwise traversing through and/or to the IPDT.

Access to the IPDT mail may be achieved through a number of APIs offered by the individual Web server components and/or the operating system.

Also, a mail server may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, information, and/or responses.

Mail Client

A mail client component 922 is a stored program component that is executed by a CPU 903. The mail client may be a conventional mail viewing application such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Microsoft Outlook Express, Mozilla, Thunderbird, and/or the like. Mail clients may support a number of transfer protocols, such as: IMAP, Microsoft Exchange, POP3, SMTP, and/or the like. A
mail client may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the mail client communicates with mail servers, operating systems, other mail clients, and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, information, and/or responses. Generally, the mail client provides a facility to compose and transmit electronic mail messages.

Cryptographic Server

A cryptographic server component 920 is a stored program component that is executed by a CPU 903, cryptographic processor 926, cryptographic processor interface 927, cryptographic processor device 928, and/or the like. Cryptographic processor interfaces will allow for expedition of encryption and/or decryption requests by the cryptographic component; however, the cryptographic component, alternatively, may run on a conventional CPU. The cryptographic component allows for the encryption and/or decryption of provided data. The cryptographic component allows for both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or decryption. The cryptographic component may employ cryptographic techniques such as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital signatures, dual signatures, enveloping, password access protection, public key management, and/or the like. The cryptographic component will facilitate numerous (encryption and/or decryption) security protocols such as, but not limited to: checksum, Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash.
function), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet
cryptography and authentication system that uses an algorithm developed in 1977 by Ron
Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure
Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like.
Employing such encryption security protocols, the IPDT may encrypt all incoming
and/or outgoing communications and may serve as node within a virtual private
network (VPN) with a wider communications network. The cryptographic component
facilitates the process of "security authorization" whereby access to a resource is
inhibited by a security protocol wherein the cryptographic component effects authorized
access to the secured resource. In addition, the cryptographic component may provide
unique identifiers of content, e.g., employing and MD5 hash to obtain a unique
signature for an digital audio file. A cryptographic component may communicate to
and/or with other components in a component collection, including itself, and/or
facilities of the like. The cryptographic component supports encryption schemes
allowing for the secure transmission of information across a communications network
to enable the IPDT component to engage in secure transactions if so desired. The
cryptographic component facilitates the secure accessing of resources on the IPDT and
facilitates the access of secured resources on remote systems; i.e., it may act as a client
and/or server of secured resources. Most frequently, the cryptographic component
communicates with information servers, operating systems, other program components,
and/or the like. The cryptographic component may contain, communicate, generate,
obtain, and/or provide program component, system, user, and/or data communications,
requests, and/or responses.
The IPDT Database

The IPDT database component 919 may be embodied in a database and its stored data. The database is a stored program component, which is executed by the CPU; the stored program component portion configuring the CPU to process the stored data. The database may be a conventional, fault tolerant, relational, scalable, secure database such as Oracle or Sybase. Relational databases are an extension of a flat file. Relational databases consist of a series of related tables. The tables are interconnected via a key field. Use of the key field allows the combination of the tables by indexing against the key field; i.e., the key fields act as dimensional pivot points for combining information from various tables. Relationships generally identify links maintained between tables by matching primary keys. Primary keys represent fields that uniquely identify the rows of a table in a relational database. More precisely, they uniquely identify rows of a table on the "one" side of a one-to-many relationship.

Alternatively, the IPDT database may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. Such data-structures may be stored in memory and/or in (structured) files. In another alternative, an object-oriented database maybe used, such as Frontier, ObjectStore, Poet, Zope, and/or the like. Object databases can include a number of object collections that are grouped and/or linked together by common attributes; they may be related to other object collections by some common attributes. Object-oriented databases perform similarly to relational databases with the exception that objects are not just pieces of data but may have other types of functionality encapsulated within a given object. If the IPDT database is implemented as a data-structure, the use of the IPDT database 919 may be integrated into another
component such as the IPDT component 935. Also, the database may be implemented as a mix of data structures, objects, and relational structures. Databases may be consolidated and/or distributed in countless variations through standard data processing techniques. Portions of databases, e.g., tables, may be exported and/or imported and thus decentralized and/or integrated.

[00109] In one embodiment, the database component 919 includes several tables 9i9a-d. A Users table 919a may include fields such as, but not limited to: user_ID, user_name, user_password, contact_info, hardware_ID, payload_history, user_evaluation and/or the like. A Hardware table 919b may include fields such as, but not limited to: hardware_ID, hardware_type, hardware_name, data_formatting_requirements, protocols, addressing_info, usage_history, hardware_requirements, user_ID, and/or the like. An Application table 919c may include fields such as, but not limited to: app_ID, protocol_ID, user_type, app_type, app_version, policy_ID, app_setting, app_interface, app_authentication, and/or the like. A protocol table 9i9d may include fields such as, but not limited to: protocol_ID, user_ID, protocol_version, protocol_request, protocol_compatability, and/or the like. A Source table 9i9e may include fields such as, but not limited to source_name, source_ID, hardware_ID, hardware_type, related_app, and/or the like. A Target table 9i9f may include fields such as, but not limited to target_name, target_ID, hardware_ID, hardware_type, related_app, and/or the like.

[00110] In one embodiment, the IPDT database may interact with other database systems. For example, employing a distributed database system, queries and data access
by search IPDT component may treat the combination of the IPDT database, an integrated data security layer database as a single database entity.

[00113] The IPDT database may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the IPDT database communicates with the IPDT component, other program components, and/or the like. The database may contain, retain, and provide information regarding other nodes and data.
The IPDTs

[00114] The IPDT component 935 is a stored program component that is executed by a CPU. In one embodiment, the IPDT component incorporates any and/or all combinations of the aspects of the IPDT that was discussed in the previous figures. As such, the IPDT affects accessing, obtaining and the provision of information, services, transactions, and/or the like across various communications networks.

[00115] The IPDT component enables the network configuration, application implementation, and/or the like and use of the IPDT.

[00116] The IPDT component enabling access of information between nodes may be developed by employing standard development tools and languages such as, but not limited to: Apache components, Assembly, ActiveX, binary executables, (ANSI) (Objective-) C(++) , C# and/or .NET, database adapters, CGI scripts, Java, JavaScript, mapping tools, procedural and object oriented development tools, PERL, PHP, Python, shell scripts, SQL commands, web application server extensions, web development environments and libraries (e.g., Microsoft’s ActiveX; Adobe AIR, FLEX & FLASH; AJAX; (D)HTML; Dojo, Java; JavaScript; jQuery(UI); MooTools; Prototype; script.aculo.us; Simple Object Access Protocol (SOAP); SWFObject; Yahoo! User Interface; and/or the like), WebObjects, and/or the like. In one embodiment, the IPDT server employs a cryptographic server to encrypt and decrypt communications. The IPDT component may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the IPDT component communicates with the IPDT database, operating systems, other program components, and/or the like. The IPDT may contain, communicate, generate, obtain,
and/or provide program component, system, user, and/or data communications, requests, and/or responses.

Distributed IPDTs

[00117] The structure and/or operation of any of the IPDT node controller components may be combined, consolidated, and/or distributed in any number of ways to facilitate development and/or deployment. Similarly, the component collection may be combined in any number of ways to facilitate deployment and/or development. To accomplish this, one may integrate the components into a common code base or in a facility that can dynamically load the components on demand in an integrated fashion.

[00118] The component collection may be consolidated and/or distributed in countless variations through standard data processing and/or development techniques. Multiple instances of any one of the program components in the program component collection may be instantiated on a single node, and/or across numerous nodes to improve performance through load-balancing and/or data-processing techniques. Furthermore, single instances may also be distributed across multiple controllers and/or storage devices; e.g., databases. All program component instances and controllers working in concert may do so through standard data processing communication techniques.

[00119] The configuration of the IPDT controller will depend on the context of system deployment. Factors such as, but not limited to, the budget, capacity, location, and/or use of the underlying hardware resources may affect deployment requirements and configuration. Regardless of if the configuration results in more consolidated and/or integrated program components, results in a more distributed series of program
components, and/or results in some combination between a consolidated and
distributed configuration, data may be communicated, obtained, and/or provided.
Instances of components consolidated into a common code base from the program
collection may communicate, obtain, and/or provide data. This may be
accomplished through intra-application data processing communication techniques
such as, but not limited to: data referencing (e.g., pointers), internal messaging, object
instance variable communication, shared memory space, variable passing, and/or the
like.

[ooi2 θ] If component collection components are discrete, separate, and/or
external to one another, then communicating, obtaining, and/or providing data with
and/or to other component components may be accomplished through inter-application
data processing communication techniques such as, but not limited to: Application
Program Interfaces (API) information passage; (distributed) Component Object Model
((D)COM), (Distributed) Object Linking and Embedding ((D)OLE), and/or the like,
Common Object Request Broker Architecture (CORBA), local and remote application
program interfaces Jini, Remote Method Invocation (RMI), SOAP, process pipes, shared
files, and/or the like. Messages sent between discrete component components for inter-
application communication or within memory spaces of a singular component for intra-
application communication may be facilitated through the creation and parsing of a
grammar. A grammar may be developed by using standard development tools such as
lex, yacc, XML, and/or the like, which allow for grammar generation and parsing
functionality, which in turn may form the basis of communication messages within and
between components. For example, a grammar may be arranged to recognize the tokens
of an HTTP post command, e.g.:
where Valuei is discerned as being a parameter because "http:/' is part of the grammar syntax, and what follows is considered part of the post value. Similarly, with such a grammar, a variable "Valuei" may be inserted into an "http:/' post command and then sent. The grammar syntax itself may be presented as structured data that is interpreted and/or otherwise used to generate the parsing mechanism (e.g., a syntax description text file as processed by lex, yacc, etc.). Also, once the parsing mechanism is generated and/or instantiated, it itself may process and/or parse structured data such as, but not limited to: character (e.g., tab) delineated text, HTML, structured text streams, XML, and/or the like structured data. In another embodiment, inter-application data processing protocols themselves may have integrated and/or readily available parsers (e.g., the SOAP parser) that may be employed to parse communications data. Further, the parsing grammar may be used beyond message parsing, but may also be used to parse: databases, data collections, data stores, structured data, and/or the like. Again, the desired configuration will depend upon the context, environment, and requirements of system deployment.

The entirety of this application (including the Cover Page, Title, Headings, Field, Background, Summary, Brief Description of the Drawings, Detailed Description, Claims, Abstract, Figures, and otherwise) shows by way of illustration various embodiments in which the claimed inventions may be practiced. The advantages and features of the application are of a representative sample of embodiments only, and are not exhaustive and/or exclusive. They are presented only to assist in understanding and
teach the claimed principles. It should be understood that they are not representative of all claimed inventions. As such, certain aspects of the disclosure have not been discussed herein. That alternate embodiments may not have been presented for a specific portion of the invention or that further undescribed alternate embodiments may be available for a portion is not to be considered a disclaimer of those alternate embodiments. It will be appreciated that many of those undescribed embodiments incorporate the same principles of the invention and others are equivalent. Thus, it is to be understood that other embodiments may be utilized and functional, logical, organizational, structural and/or topological modifications may be made without departing from the scope and/or spirit of the disclosure. As such, all examples and/or embodiments are deemed to be non-limiting throughout this disclosure. Also, no inference should be drawn regarding those embodiments discussed herein relative to those not discussed herein other than it is as such for purposes of reducing space and repetition. For instance, it is to be understood that the logical and/or topological structure of any combination of any program components (a component collection), other components and/or any present feature sets as described in the figures and/or throughout are not limited to a fixed operating order and/or arrangement, but rather, any disclosed order is exemplary and all equivalents, regardless of order, are contemplated by the disclosure. Furthermore, it is to be understood that such features are not limited to serial execution, but rather, any number of threads, processes, services, servers, and/or the like that may execute asynchronously, concurrently, in parallel, simultaneously, synchronously, and/or the like are contemplated by the disclosure. As such, some of these features may be mutually contradictory, in that they cannot be simultaneously present in a single embodiment. Similarly, some features are applicable to one aspect of the invention, and
inapplicable to others. In addition, the disclosure includes other inventions not presently claimed. Applicant reserves all rights in those presently unclaimed inventions including the right to claim such inventions, file additional applications, continuations, continuations in part, divisions, and/or the like thereof. As such, it should be understood that advantages, embodiments, examples, functional, features, logical, organizational, structural, topological, and/or other aspects of the disclosure are not to be considered limitations on the disclosure as defined by the claims or limitations on equivalents to the claims.
What is claimed is:

1. A processor-implemented interactive proximity display tether method, comprising:
   receiving a request to initialize a remote display tether component;
   querying for tether target devices;
   obtaining a selection of a tether target device;
   configuring a communications channel for a selected tether target device;
   instantiating a remote display through the communications channel;
   determining a type of the selected tether target device;
   instantiating an interactive tether application on a source device if the selected tether target device is processor-based;
   generating data based on control inputs; and
   casting the generated data to drive the remote display through the communications channel.

2. The method of claim 1, wherein query for tether target devices comprises:
   receiving an indication of tether target devices; and
   search a range of local area network for tether target devices based on the received indication.

3. The method of claim 2, wherein the indication of tether target devices may be any of an IP address, MAC address, acronym, hardware label, digital signature and driver certificate.

4. The method of claim 1, wherein query for tether target devices comprises:
form a query based on registered tether target devices on a communication stack in the database.

5. The method of claim 1, wherein query for tether target devices comprises:
locating tether target devices based on zero configuration protocols,
wherein the zero configuration protocols may be any of Service Location Protocol (SLP), Universal Plug and Play (UPnP), Jini, Bluetooth Service Discovery Protocol, WS-Discovery, Proprietary Discovery Protocol using UDP and Bonjour.

6. The method of claim 1, further comprising:

generating a list of available tether target device based on the query; and
presenting the list of available tether target device via a user interface.

7. The method of claim 1, wherein determining a type of the tether target device comprises determining whether the tether target device is a physical tether or a processor-enabled device.

8. The method of claim 7, wherein the physical tether may be a direct audio/video connection.

9. The method of claim 7, further comprising instantiating a display mirror mode on a remote display via the physical tether if the tether target device is a physical tether.

10. The method of claim 9, wherein data transmitted to the target device through the physical tether under the mirror mote may be in compliance with any of Component VideoS-Video, HDMI, VGA, DVI, DisplayPort.

11. The method of claim 7, further comprising determining whether the tether target device is running any of a distributed object oriented application, a web server, a remote display client and a web browser if the tether target device is processor-enabled.

12. The method of claim 7, further comprising:
if the tether target device is processor-enabled,

2 enabling custom communications channels connecting the tether target device,

3 providing instructions to instantiate the tether target device,

4 and engaging an interactive tether enabled application on the tether target device.

13. The method of claim 11, further comprising:

if the tether target device is running the web browser,

instantiating a web server,

connecting to the tether target device,

instructing the target devices web browser to expand its window to the full size of the screen.

14. The method of claim 11, further comprising: if the tether target device is running the remote desktop client, prompting with a screen instructing a user on what to enter on the target device to complete the connection

15. The method of claim 11, further comprising: running a graphics rendering engine if the tether target device is running a distributed object oriented application.

16. The method of claim 12, wherein engaging the interactive tether enabled application comprises one of:

transferring more elaborate versions of interactive applications to the tether target device; and

facilitating the tether target device automatically download and install more elaborate versions of interactive applications.

17. The method of claim 1, wherein the generated data is in the format of a binary data packet, wherein the binary data packet includes fields: message type, sequence
1 number, acknowledgement number, data offset, data length, checksum, options and 2 data payload.

3 18. The method of claim Yj, wherein the data payload includes at least one of 4 accelerometer information, pointer coordinates (pressing on the screen), images, GPS 5 information, user information, and/or the like.

6 19. The method of claim 1, wherein the generated data is transmitted to the tether 7 target device based on a Common Object Request Broker Architecture (CORBA) 8 mechanism.

9 20. The method of claim 19, wherein the CORBA mechanism encapsulates the generated data in a plurality of objects.

21. The method of claim 1, further comprising:

   querying for a client device;

   obtaining a selection of one or more available client devices; and

   establishing communications on the communications channel for the 15 selected client devices.

22. The method of claim 21, further comprising:

   exchanging data with the selected client devices via the communication 18 channel in a real-time manner.

23. An interactive proximity display tether system, comprising:

   means for receiving a request to initialize a remote display tether enabled 20 component;

   means for querying for tether target devices;

   means for obtaining a selection of a tether target device;

   means for configuring a communications channel for a selected tether target
device;

means for instantiating a remote display through the communications channel;

means for determining a type of the selected tether target device;

means for instantiating an interactive tether application on a source device if the
selected tether target device is processor-based;

means for generating data based on control inputs; and

means for casting the generated data to drive the remote display through the
communications channel.

24. A processor-readable medium storing a plurality of processing instructions,
comprising issuable instructions by a processor to:

receive a request to initialize a remote display tether enabled component;

query for tether target devices;

obtain a selection of a tether target device;

configure a communications channel for a selected tether target device;

instantiate a remote display through the communications channel;

determine a type of the selected tether target device;

instantiate an interactive tether application on a source device if the selected

tether target device is processor-based;

generate data based on control inputs; and

cast the generated data to drive the remote display through the communications

channel.
Provide option to run IPDT enabled application 300

Submit request to initialize IPDT application 302

Query for tether targets 310

Show tether targets 330

Submit selection of tether target 335

Connect with tether target 340

Extend to additional clients - Co-play? 360

Proceed with IPDT 367

Proceed with Figs. 5A-D 365
Figure 3B

User submit target? 315

Query communication stack? 318

Search result non-null? 315

Generate a list of available target devices 325

Display unsuccessful query and proceed with 310

Locate target devices by O-configuration protocols 322

Search for target device based on user submitted info 316

Form a query based on registered devices/channels 320

Query for tether targets 310

SLP
UHiP
Jini
Bluetooth
PDP by UDP
WS-Discovery
Bonjour
FIGURE 6A

Engage mobile device source app 601

Engage target device app 604

Check data link (e.g., ping remote display) 607

Include Client Devices for Co-Play IPDT616

Established? 610

Y

Receive sensor input 619

Update mobile device app configuration 622

Send period? 625

Y

Configure display signal message 638

Update remote access software 641

Wait 628

N

Add prev. input to display signal cache 634

New input? 631

Y

N

Continue? 650

Done 653

N

Provide remote display visualization of display signal 647

Send display signal to remote display 644
Engage source/client device app 656

Receive sensor input 659

Update mobile device app configuration 662

Add prev. input to display signal cache 675

New input? 672

Y

Received web client data request? 665

Y

Configure display signal 678

Publish display signal via web server software 681

Access web-published content at remote display 684

Update display? 687

N

Done 690

N

Wait 668

N

Wait 668

N

Wait 668
INTERNATIONAL SEARCH REPORT

PCT/US2009/062253

A CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04N 7/1 67 (2009.01)
USPC - 725/81

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(8) - H04N 7/167 (2009.01)
USPC - 725/81

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 practicable, search terms used)
PatBase

C DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No</th>
</tr>
</thead>
</table>

D Further documents are listed in the continuation of Box C

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

Date of the actual completion of the international search
14 December 2009

Date of mailing of the international search report
29 DEC 2009

Name and mailing address of the ISA/US
Mail Stop PCT, Attn ISA/US, Commissioner for Patents
P.O. Box 1450, Alexandria, Virginia 22313-1450
Facsimile No 571-273-3201

Authorized officer
Blaine R Copenheaver
PCT Helpdesk 571-272-4300
PCT OSP 571-272-7774

Form PCT/ISA/2 10 (second sheet) (July 2009)