Abstract
Systems and methods for providing a generic interface in a communications environment are provided. A communications services interface may receive a first and second application function call specifying a first and second communications channel, respectively. The first and second communications channels may utilize a first and second communications protocol, respectively, to communicate over a network. The communications services interface may send a first protocol command and a second protocol command to control communications over the first and second communications channels, respectively. The first and the second protocol commands may be responsive to the first and second application function calls, respectively. The second application function call may have a same syntax as the first application function call, and the second communications protocol may be different than the first communications protocol.
FIG. 1
Register Protocol Stack with Communications Services Interface 302

Notify Application of Available Protocol 304

Allocate a Communications Service 306

Create Communications Service via Protocol Stack 308

Report Creation Status of Communications Service 310

Report Communications Service Allocation Status 312

Set Communications Service Parameters 314

Configure Protocol Stack According to Communications Service Parameters 316

Activate Communications Service 318

Activate Protocol Stack 320

Report Activation Status of Protocol Stack 322

Report Communications Service Activation Status 324

Repeat Steps 302 – 324 for Another Communications Service 326

End

FIG. 3
Start

Signal to Initiate Call 402

Command Protocol Stack to Initiate Call 404

Perform Protocol Signaling to Called Party 406

Report Protocol Signaling Called Party 408

Report Calling Called Party 410

Report Receiving Acknowledgement from Called Party via Protocol Stack 412

Report Called Party Acknowledges Call 414

Report Receiving Acceptance of Call from Called Party via Protocol Stack 416

Report Called Party Accepting Call 418

End

FIG. 4
Start

Signal to Terminate Call 502

Command Protocol Stack to Terminate Call 504

Perform Protocol Signaling with Called Party 506

Report Call Termination with Called Party by Protocol Stack 508

Report Call Termination 510

End

FIG. 5
Start

Receive Protocol Signaling for Incoming Call 602

Generate Protocol Event for Incoming Call 604

Notify Application of Incoming Call 606

Acknowledge Incoming Call 608

Generate Protocol Command to Acknowledge Incoming Call 610

Accept Incoming Call 612

Generate Protocol Command to Accept Incoming Call 614

End

FIG. 6
Start

Include Common Function Call in Application Source Code 702

Target Compiler to A Communications Device 704

Compile Application Executable 706

Repeat Steps 704 and 706 for Another Communications Device 708

End

FIG. 7
Start

Utilize a First Voice Communications Protocol with a First Entity 802

Utilize a Second Voice Communications Protocol with a Second Entity 804

Communicatively Couple the First Entity and the Second Entity 806

End

FIG. 8
Start

Receive First Application Function Call 902

Send First Protocol Command 904

Receive Second Application Function Call 906

Send Second Protocol Command 908

End

FIG. 9
COMMUNICATIONS DEVICE 1000

COMMUNICATIONS INTERFACE 1002

PROCESSOR 1004

MEMORY 1006

STORAGE 1008

FIG. 10
SYSTEMS AND METHODS FOR PROVIDING A GENERIC INTERFACE IN A COMMUNICATIONS ENVIRONMENT

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit and priority of U.S. Provisional Application No. 60/982,684 entitled “Systems and Methods for Providing Unified Mobile Communications,” and filed on Oct. 25, 2007, which is incorporated herein by reference.

BACKGROUND

[0002] 1. Field of the Invention

[0003] The present invention relates generally to the field of digital communications interfaces and more specifically to systems and methods for providing a generic interface in a communications environment.

[0004] 2. Background Art

[0005] Digital communications may be performed over a variety of communications media, including voice, email, text messaging, multimedia messaging, etc. These digital communications may utilize any of a variety of protocols. The communications media may not be compatible with one another. Therefore, a communications device may not be able to seamlessly communicate over two different communications media, such as voice and text. In addition, the variety of protocols may also be incompatible with one another. These protocol incompatibilities may prevent a communications device that utilizes one protocol from communicating with another communications device that utilizes a different protocol.

[0006] Because of these incompatibilities, a software application that is intended to communicate using more than one communication media and/or protocol may need to include separate code with separate function calls for each communications media and/or protocol. This can become unwieldy and cumbersome, making multiple communications media and/or protocols interoperating seamlessly in the application difficult or impractical.

SUMMARY

[0007] A method for providing a communications services interface is disclosed. The method includes receiving a first application function call. The first application function call specifies a first communications channel utilizing a first communications protocol to communicate over a network. The method further includes sending a first protocol command to control communications over the first communications channel. The first protocol command is responsive to the first application function call.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. 1 illustrates an exemplary communications system.

[0009] FIG. 2 illustrates an exemplary communications system including a communications device configured to communicate over a network.

[0010] FIG. 3 illustrates an exemplary method of allocating and activating communications services.

[0011] FIG. 4 illustrates an exemplary method of initiating a call.

[0012] FIG. 5 illustrates an exemplary method of terminating a call.

[0013] FIG. 6 illustrates an exemplary method of accepting an incoming call.

[0014] FIG. 7 illustrates an exemplary method of providing an interface between a communications application and a plurality of communications protocols.

[0015] FIG. 8 illustrates an exemplary method of providing communications between multiple entities.

[0016] FIG. 9 illustrates an exemplary method of providing communications services interface.

[0017] FIG. 10 illustrates an exemplary communications device.

DETAILED DESCRIPTION

[0018] Systems and methods for providing a generic interface in a communications environment are described. These systems and methods enable applications to seamlessly communicate over multiple communications media and/or use multiple communications protocols by using a shared application programming interface. The application programming interface may provide a common command function call configured to interface with multiple protocol stacks. The shared application programming interface may abstract the uniqueness of and differences between various communications media and protocols into a common set of function calls and data structures.

[0019] Using the application programming interface, an application may interface to multiple incompatible communications media and communications protocols using a common set of software code. Because the details of the communications protocols are abstracted by the application programming interface, the application software code may not need to address the details of each individual communications protocol. Because the application software code may be independent of the details of the individual communications protocols, the application software code may easily be expanded to support new communications protocols also supported by the application programming interface.

[0020] FIG. 1 illustrates an exemplary communications system 100. The communications system 100 includes a communications device 102 and a network 104. The communications device 102 may be configured to communicate using a variety of communications protocols 106 over various communications media 108. One or more communications services providers 110 may provide various communications services over the network 104.

[0021] The communications device 102 may include a mobile communications device such as a cellular telephone,
cordless telephone handset, portable computer equipped with wireless internet access, personal digital assistant, etc. The communications device 102 may be configured to operate in multiple modes, such as both Wi-Fi and cellular, and thereby provide communications via multiple communications media 108 and/or networks 104 using one or more communications protocols 106. The communications device 102 may also include a fixed-location communications device such as a desktop personal computer with internet access, desktop telephone, conference speakerphone, etc. Software configured to implement the communications protocols 106 may be associated with the communications device 102. The communications device 102 may include a transceiver configured to communicate over the network 104.

[0022] The various communications media 108 may include media connections provided by various communications protocols including circuit-switched (CS) protocols, internet protocol (IP), and proprietary protocols. The communications media 108 may include voice communications such as plain old telephone service (POTS), cellular telephone service, voice over internet protocol (VoIP), voice mail, and/or text communications such as email, Short Message Service (SMS), instant messaging (IM), and text chat services. Multimedia communications including text, still images, video, and audio may also be provided as part of the communications media 108. The various communications media 108 may also include internet access, web browsing, file transfer protocol (FTP), and the like.

[0023] The media connections provided by the communications media 108 may include communications services associated with the one or more communications services providers 110. The one or more communications services providers 110 may associate a communications services provider account with the communications device 102. The communications services provider account may be associated with one or more communications protocols 106, usernames, passwords, internet addresses, and other communications attributes.

[0024] A communications channel may be provided by the communications device 102 utilizing the one or more communications services providers 110 over the network 104. The communications channel may be established by the communications device 102 connecting and authenticating with the one or more communications services providers 110 using the communications protocols 106. The communications device 102 may establish one or more communications channels simultaneously. Each of the communications channels may use a different protocol or be carried by a different communications services provider 110. In some embodiments, more than one communications channel may use a single protocol or be provided by a single communications services provider 110. Each communications channel may be associated with a unique service identification tag (ID), and be independent from other communications channels that may be established by the communications device 102.

[0025] The communications protocols 106 may adhere to standards defined for communicating over the various communications media 108. For example, communications protocols 106 adhering to internet standards may include HTTP, AJAX, File Transfer Protocol (FTP), User Datagram Protocol (UDP), and Transmission Control Protocol/Internet Protocol (TCP/IP). In addition, communications protocols adhering to voice communications standards may include Global System for Mobile communications (GSM), Extensible Messaging and Presence Protocol (XMPP), and Session Initiation Protocol (SIP). GSM may be used to support cellular telephone communications over cellular networks provided by one or more communications services providers 110. XMPP may be used to support instant messaging services provided by one or more communications services providers 110. SIP may be used to support voice over internet protocol (VoIP) communications provided by one or more communications services provider 110. Furthermore, communications protocols adhering to presence and instant messaging standards may include Common Profile for Presence (CPP), Common Profile for Instant Messaging (CPIM), Common Presence and Instant Messaging (CPIM) Message Format, and XMPP.

[0026] In one embodiment, the communications protocols 106 may be implemented on a computing device (not shown) disposed external to the communications device 102 and communicatively coupled with the communications device 102 via the network 104. In this embodiment, commands configured to exercise the communications protocols 106 may be issued from the communications device 102 to the computing device.

[0027] The network 104 may include a wired communications network such as an Ethernet local area network (LAN) or a landline POTS network. The network 104 may also include a wireless communications network such as a Wi-Fi network, a WiMax network, or a cellular telephone network. The network 104 may include an internet or World Wide Web (WWW) network. The network 104 may be configured to carry communications associated with the communications media 108 and the communications protocols 106.

[0028] FIG. 2 illustrates an exemplary communications system 200 including the communications device 102 configured to communicate over the network 104. The communications device 102 includes a communications system software 202 as well as an operating system 204. The communications system software 202 includes an application 206 configured to communicate with another entity over the network 104. The other entity may include another communications device (not shown) configured to communicate over the network 104. A communications services interface module 208 includes software for providing communications services to the application 206 to enable the application 206 to communicate over the network 104. The communications system software 202 also includes protocol stacks 210 and a DSP engine 212. The protocol stacks 210 may comprise modules including software and/or electronic circuitry configured to implement communications protocols for communicating over the network 104 via a network interface 214. The protocol stacks 210 include a XMPP protocol stack 210A, a SIP protocol stack 210B, a proprietary protocols protocol stack 210C, an IM services protocol stack 210D, and a cellular services protocol stack 210E, each of which may be hereinafter referred to as a member of the protocol stacks 210. The DSP engine 212 may comprise a module including software and/or electronic circuitry configured to perform digital signal processing functions in conjunction with the communications services interface module 208.

[0029] Each member of the protocol stacks 210 may be configured to establish one or more communications channels over the network 104 by interfacing with the network 104 via the network interface 214. The network interface 214 may also interface with, communicate with, and/or interoperate with the operating system 204 to implement the one or more
communications channels. The operating system 204 may include Palm OS, Symbian, Windows CE, Windows Mobile, Windows XP, Windows Vista, MacOS, UNIX, Linux, Solaris, Nucleus OS, Vx Works, etc.

[0030] The application 206 may comprise a software module including a user interface. The user interface may be configured to receive commands and instructions from a user associated with the communications device 102 and report a status relating to the one or more communications channels to the user. The application 206 may also be configured to receive communications input from the user and provide communications output to the user.

[0031] The communications services interface module 208 provides generalized control of the protocol stacks 210 for establishing and manipulating the communications media 108. Other protocol stacks (not shown) implementing other communications protocols, such as Simple Traversal of User Datagram Protocol Through Network Address Translators (STUN), may also be controlled by the communications services interface module 208. The communications services interface module 208 may include data and algorithms taking into account the communications services which each member of the protocol stacks 210 is capable of supporting. The communications services interface module 208 may also include data and algorithms configured to couple communications services and functionality between one member of the protocol stacks 210 or the DSP engine 212 with another member of the protocol stacks 210 or the DSP engine 212. The communications services interface module 208 may enable one or more communications services over one or more communications media 108, using one or more communications protocols 106, and carried by the one or more communications services providers 110 to seamlessly interface with one another.

[0032] The communications services interface module 208 may utilize the DSP engine 212 to provide digital processing of voice, audio, and/or video in conjunction with the protocol stacks 210. The communications services interface module 208 may create common audio tones used in communications services supported by the protocol stacks 210, such as dial tones, call progress tones, DTMF tones, and telephone ringing tones.

[0033] The communications services interface module 208 may control multimedia streaming over the network 104. For example, the communications services interface module 208 may control the streaming of voice over the network 104. The communications services interface module 208 may also control the streaming of audio and/or video over the network 104.

[0034] The communications services interface module 208 includes an application programming interface 216 and a communications services protocol interface 218. The application programming interface 216 is configured to provide a variety of function calls related to communications for use by the application 206. The function calls may be configured to abstract the details of underlying protocols implemented in the protocol stacks 210 and the DSP engine 212. By abstracting these details, the function calls enable a generalized common interface to the plurality of underlying protocols which may be independent of the details of each of the underlying protocols, according to exemplary embodiments. The function calls may abstract protocol-specific communications between the communications services interface module 208 and each member of the protocol stacks 210 into common function calls between the application 206 and the communications services interface module 208. The application 206 may use the function calls to send one or more API calls 220 to the communications services interface module 208. The application 206 may also use the function calls to configure the application 206 to process one or more API events 222 which may be received from the communications services interface module 208.

[0035] The application programming interface 216 may provide support for two types of the one or more API calls 220. A first type of the one or more API calls 220 may be configured to enable the application 206 to control an action performed by at least one member of the protocol stacks 210 or the DSP engine 212. A second type of the one or more API calls 220 may be configured to establish a feedback mechanism by which at least one member of the protocol stacks 210 or the DSP engine 212 may notify the application 206 of an event or status.

[0036] The feedback mechanism may include the one or more API events 222. Each API event of the one or more API events 222 may be associated with a particular function call, or instance of the one or more API calls 220, which may have established the feedback path associated with the API event. The one or more API events 222 may be asynchronous. Examples illustrating the feedback mechanism are described herein with reference to FIGS. 3-6.

[0037] A single function call may be used by the application 206 to interface with a plurality of the members of the protocol stacks 210 by changing a parameter (or argument) of the single function call. For example, source code associated with the application 206 may include a single function call that includes a parameter specifying that the XMPP protocol stack 210A be utilized for performing a communications function. The single function call may then be used again by the application source code to perform the communications function using the SIP protocol stack 210B in addition, simply by replacing the parameter specifying that the XMPP protocol stack 210A be utilized with a parameter specifying that the SIP protocol stack 210B be utilized.

[0038] In response to the one or more API calls 220 received from the application 206, the communications services interface module 208 may control and communicate with the protocol stacks 210 as well as the DSP engine 212 to perform functions related to the one or more communications channels carried over the network 104. The communications services interface module 208 may issue protocol commands 222 to the protocol stacks 210. The communications services interface module 208 may also issue DSP commands 226 to the DSP engine 212. The protocol commands 224 issued to one member of the protocol stacks 210, such as the XMPP protocol stack 210A, may differ in form and/or syntax from the protocol commands 224 issued to another member of the protocol stacks 210, such as the SIP protocol stack 210B.

[0039] A first protocol command, such as one of the protocol commands 224, issued in response to a first API call, such as the one or more API calls 220, having a given protocol-dependent parameter may be syntactically different from a second protocol command issued in response to a second API call having a same function name as the first API call but a different protocol-dependent parameter than the first API call. The syntactical difference may be embodied by a different function name, a different set of parameters, or both. The syntactical difference may be dependent upon, or responsive to, a syntax required by each protocol implemented within the set of the protocol stacks 210 and the DSP engine 212. Each
protocol command of the protocol commands 224 may conform to a syntax standard corresponding to the member of the protocol stacks 210 to which the protocol command is directed. Each member of the protocol stacks 210 may have a different syntax standard for protocol commands 224 than other members of the protocol stacks 210.

[0040] The communications services interface module 208 may receive protocol events 228 from the protocol stacks 210. The communications services interface module 208 may also receive DSP events 230 from the DSP engine 212. An instance of the protocol events 228 received from one member of the protocol stacks 210 may differ in syntax and/or form from another instance of the protocol events 228 received from a different member of the protocol stacks 210. A specific protocol event among the instances of protocol events 228 may be in response to one or more communications 232 received through the network 104 by the communications device 102. For example, the specific protocol event may notify the communications services interface module 208 that a communications channel has been established in response to a network acknowledgement, which may include an instance of the one or more communications 232 received through the network 104.

[0041] The communications services interface module 208 may send the one or more API events 222 to the application 206 when triggered by the protocol events 228 to notify the application 206 of a particular occurrence or status. The one or more API events 222 may also be sent from the communications services interface module 208 when triggered by an instance of the DSP events 230 issued by the DSP engine 212 to notify the application 206 of a particular occurrence or status. The protocol events 228 and the DSP events 230 may be asynchronous.

[0042] Each instance of the protocol events 228 issued in response to a first communication, which may include an instance of the one or more communications 232, may be syntactically different from the other instances of the protocol events 228 issued in response to a second communication, which may include another instance of the one or more communications 232. The syntactical difference between each instance of the protocol events 228 may be dependent upon a syntax standard required by each protocol supported by members of the protocol stacks 210. The communications services interface module 208 may convert the syntactically different instances of the protocol events 228 and the DSP events 230 into the one or more API events 222 having a common syntax. For example, a first API event, such as an instance of the one or more API events 222, sent from the communications services interface module 208 to the application 206 in response to the first communication may have a same function name as a second API event sent from the communications services interface module 208 to the application 206 in response to the second communication, although a first protocol event, such as an instance of the protocol events 228, associated with the first communication and a second protocol event associated with the second communication may be syntactically different. The first API event may differ from the second API event in only a parameter value.

[0043] The communications services protocol interface 218 may provide a conversion between the common function calls provided by the application programming interface 216 and the protocol-specific communications between the communications services interface module 208 and the protocol stacks 210 and the DSP engine 212. The communications services protocol interface 218 may provide the conversion between a common API call, which may include an instance of the one or more API calls 220, and each of the protocol commands 224 and the DSP commands 226. The communications services protocol interface 218 may also provide the conversion between each instance of the protocol events 228 and the DSP events 230 and a common API event, which may include an instance of the one or more API events 222.

[0044] The protocol stacks 210 may communicate with the communications services interface module 208 via a command-event model. For example, the communications services interface module 208 may send a protocol command, such as an instance of the protocol commands 224, to a protocol stack which is a member of the protocol stacks 210. The protocol command may contain a block of digital data representative of the command. The communications services interface module 208 may then receive a protocol event, such as an instance of the protocol events 228, sent by the protocol stack. The protocol event may contain a block of digital data representative of the event. The event may include a status report in response to the protocol command, an IP event received from a communications channel over the network 104, etc. The communications services interface module 208 may use one or more Inter-Process Communication (IPC) techniques to send the protocol command and to receive the protocol event. A different IPC technique may be used to send the protocol command than to receive the protocol event. Examples of IPC techniques include pipes, sockets, remote procedure calls (RPC), shared memory, and message passing.

[0045] The communications services interface module 208 may support hot swapping or plug-and-play operation of the members of the protocol stacks 210. For example, a selected member of the protocol stacks 210 may be deactivated while the communications services interface module 208 continues operating and supporting other members of the protocol stacks 210. The selected member of the protocol stacks 210 may be deactivated without interrupting or ceasing operation of any communications services or communications channels supported by the other members of the protocol stacks 210. The selected member of the protocol stacks 210 may be deactivated because the communications services interface module 208 detects an error condition in the selected member of the protocol stacks 210, because hardware or system resources of the communications device 102 are not sufficient to support continued operation of the selected member of the protocol stacks 210, because the selected member of the protocol stacks 210 is being upgraded to a new version of the selected member of the protocol stacks 210, because the user associated with the communications device 102 wishes to deactivate communications services supported by the selected member of the protocol stacks 210, because resources associated with the network 104 and/or the communications services providers 110 are insufficient to support operation of communications services supported by the selected member of the protocol stacks 210, or other various reasons.

[0046] As another example, a new protocol stack may be added as a new member of the protocol stacks 210 and be registered with and supported by the communications services interface module 208 while the communications services interface module 208 continues to operate and support other members of the protocol stacks 210 as well as any one or more communications services or communications chan-
nels supported by the other members of the protocol stacks 210. The new protocol stack may be dynamically added when one or more resources become available to support operation of the new protocol stack. The one or more resources that become available may include hardware or system resources of the communications device 102 such as a network interface, a required quantity of free memory, or processing capacity. The one or more resources that become available may also include resources associated with the network 104 and/or the communications services providers 110 such as communications services, communications bandwidth, or network servers. The new protocol stack may also be dynamically added when one or more communications services and/or protocols supported by the new protocol stack are desired.

[0047] As a further example, the selected member of the protocol stacks 210 may be deactivated because of an error condition. Examples of error conditions include bugs, crashes, unresponsiveness, incorrect behavior, etc. Then, the selected member of the protocol stacks 210 may be restarted to restore the selected member of the protocol stacks 210 to full operation. While any communications services and/or communications channels supported by the selected member of the protocol stacks 210 may be interrupted when the selected member of the protocol stacks 210 is deactivated, the interrupted communications services and/or communications channels may be restored or reactivated after the selected member of the protocol stacks 210 is restarted.

[0048] In an embodiment, the communications services interface module 208 may monitor usage of the members of the protocol stacks 210 as well as available hardware and system resources of the communications device 102. The communications services interface module 208 may dynamically activate and/or deactivate various members of the protocol stacks 210 as the various members of the protocol stacks 210 are needed and/or as available resources allow. In this way, a set of active members of the protocol stacks 210 may dynamically change depending on dynamically changing system resources of the communications device 102, resources associated with the network 104, and/or resources associated with the communications services providers 110. By dynamically changing the set of active members of the protocol stacks 210, memory and/or processing resources of the communications device 102 may be conserved.

[0049] The application 206 may be written as source code in a computer language such as C, C++, or Java. The application source code may include function calls provided by the application programming interface 216 to include communications functionality associated with each of the protocols in the protocol stacks 210 as well as the DSP engine 212. A software compiler executing on a computing processor (not shown) may compile the source code into object code. The object code may be targeted to a particular computing processor associated with the communications device 102. The application programming interface 216 may include one or more libraries configured to provide object code for various target computing processors. A compiler switch may be set to control the compiler to generate executable object code for the target computing processor associated with the compiler switch setting. Therefore, the application programming interface 216 may enable a single source code to be targeted to more than one target computing processor and/or communications device 102 based upon a compiler switch setting.

[0050] FIG. 3 illustrates an exemplary method of allocating and activating one or more communications services. The method illustrates how an application programming interface, such as the application programming interface 216 discussed herein, may be used by an application, such as the application 206, to allocate and activate communications channels using different protocol stacks, such as the members of the protocol stacks 210. The method also illustrates how a communications services interface module, such as the communications services interface module 208, may support plug-and-play protocol stacks whereby new members of the protocol stacks 210 may be dynamically added and supported by the communications services interface module 208 without re-coding, recompiling, or interrupting operation of the communications services interface module 208 and/or application 206. The new members of the protocol stacks 210 may be dynamically added and supported without interrupting operation of the one or more communications services already activated. After the one or more communications services are allocated and activated, the communications channels may be used by the application 206. The application 206 may include a state machine configured to implement the method.

[0051] In step 302, a protocol stack, such as a member of the protocol stacks 210, sends a registration event, such as an instance of the protocol events 220, to the communications services interface module 208. By sending the registration event, the protocol stack may register with the communications services interface module 208. The registration event may include data regarding the protocol stack such that the communications services interface module 208 may use the protocol stack to allocate and activate one or more communications services associated with the protocol stack. The communications services interface module 208 may be configured to utilize the protocol stack after receiving the registration event even though the communications services interface module 208 may have no data regarding the protocol stack prior to receiving the registration event. In this way, the communications services interface module 208 may support plug-and-play operation of a dynamically added protocol stack. The communications services interface module 208 may therefore support communications interface protocols via the dynamically added protocol stack which the communications services interface module 208 did not previously support.

[0052] In step 304, the application 206 is notified of the availability of one or more protocols supported by the protocol stack registered in step 302. The communications services interface module 208 may send a protocol availability event, which may include an instance of the one or more API events 222, to the application 206. The protocol availability event may include data regarding the one or more protocols supported by the protocol stack registered in step 302. In conjunction with the data regarding the one or more protocols, the application 206 may use the one or more protocols after receiving the protocol availability event even though the application 206 may have had no data regarding using the one or more protocols via the communications services interface module 208 prior to receiving the protocol availability event. In this way, the application 206 may utilize the one or more protocols supported by the protocol stack which is dynamically added and registered in step 302.

[0053] In step 306, a new communications service is allocated. The application 206 may use an API function call to send an allocation command, which may include an instance of the one or more API calls 220, to the communications
services interface module 208. The allocation command may specify which protocol, and consequently which member of the protocol stacks 210, is to be associated with the new communications service.

In step 308, the new communications service is created via the protocol stack associated with the new communications service, which may include a member of the protocol stacks 210. The communications services interface module 208 may send a communications service creation command, which may include an instance of the protocol commands 224, to the protocol stack associated with the new communications service. The communications service creation command may be sent in response to receiving the allocation command sent by the application 206 in step 306. The communications service creation command sent to the protocol stack associated with the new communications service may be syntactically different from another communications service creation command that may be sent to a different member of the protocol stacks 210. The communications service creation command sent, and the protocol stack associated with the new communications service, may be responsive to a parameter included in the allocation command sent by the application 206 to the communications services interface module 208 in step 306.

The protocol stack associated with the new communications service may perform functions related to creating the new communications service in response to receiving the communications service creation command. The functions performed by the protocol stack associated with the new communications service to create the new communications service may be specific to the protocol associated with the new communications service. For example, the SIP protocol stack 2103 may initialize a user agent, while the XMPP protocol stack 210A may initialize an endpoint.

In step 310, a creation status of the new communications service is reported to the communications services interface module 208. After the protocol stack associated with the new communications service creates the new communications service, the protocol stack associated with the new communications service may send a protocol service creation event, which may include an instance of the protocol events 228, to the communications services interface module 208 to notify the communications services interface module 208 that the new communications service has successfully created the new communications service. In some embodiments, the protocol service creation event may be sent to the communications services interface module 208 to report a communications service creation error.

In step 312, a communications service allocation status is reported to the application 206. After the communications services interface module 208 receives the protocol service creation event in step 310, the communications services interface module 208 may send a corresponding interface service creation event, which may include an instance of the one or more API events 222, to the application 206. The corresponding interface service creation event may report the communications service allocation success or failure status in response to the creation status reported by the protocol stack associated with the new communications service in step 310.

In step 314, communications service parameters are set. The application 206 may send one or more communications service parameter setting commands, each of which may correspond to an instance of the protocol commands 224, to the communications services interface module 208 using an API function call which may include an instance of the one or more API calls 220. The communications service parameters may include a uniform resource identifier (URI), server name and/or internet address, login userid and/or password, and so forth.

In step 316, the protocol stack associated with the new communications service is configured according to the communications service parameters set in step 314. The communications services interface module 208 may send one or more protocol parameter setting commands, each of which may include an instance of the protocol commands 224, to the protocol stack associated with the new communications service. The one or more protocol parameter setting commands may be sent in response to receiving the one or more communications service parameter setting commands sent by the application 206 in step 314. The one or more protocol parameter setting commands sent to the protocol stack associated with the new communications service may be syntactically different from one or more protocol parameter setting commands that may be sent to a different member of the protocol stacks 210.

In step 318, the new communications service is activated. The application 206 may send an activation command to the communications services interface module 208 using an API function call which may include an instance of the one or more API calls 220. The activation command may specify the new communications service to be activated by identifying the previously allocated new communications service.

In step 320, the protocol stack associated with the new communications service is activated. The communications services interface module 208 may send a communications service activation command, which may include an instance of the protocol commands 224, to the protocol stack associated with the new communications service. The communications service activation command may be sent in response to receiving the activation command sent by the application 206 in step 318. The communications service activation command sent to the protocol stack associated with the new communications service may be syntactically different from a communications service activation command that may be sent to a different member of the protocol stacks 210.

The protocol stack associated with the new communications service may perform functions related to activating the new communications service in response to receiving the communications service activation command. The functions performed by the protocol stack associated with the new communications service to activate the new communications service may be specific to the protocol associated with the new communications service. For example, the SIP protocol stack 2103 may register a user agent, while the XMPP protocol stack 210A may connect and authenticate against an XMPP server.

In step 322, an activation status of the protocol stack associated with the new communications service is reported to the communications services interface module 208. After the protocol stack associated with the new communications service activates the new communications service, the protocol stack associated with the new communications service may send a protocol activation event, which may include an instance of the protocol events 228, to the communications services interface module 208 to notify the communications services interface module 208 that the protocol stack associated with the new communications service has success-
fully activated the new communications service. In some embodiments, the protocol activation event may be sent to the communications services interface module 208 to report a communications service activation error.

[0064] In step 324, a communications service activation status is reported to the application 206. After the communications services interface module 208 receives the protocol activation event in step 322, the communications services interface module 208 may send a corresponding interface service activation event, such as an instance of the one or more API events 222, to the application 206. The corresponding interface service activation event may report the communications services activation success or failure status in response to the activation status reported by the protocol stack associated with the new communications service in step 322.

[0065] In optional step 326, another communications service may be allocated and activated by repeating steps 302-324. The other communications service may utilize a different protocol or member of the protocol stacks 210 than the prior new communications service. In other embodiments, the other communications service may utilize the same protocol or member of the protocol stacks 210 as the prior new communications service, but may allocate and activate a separate communications channel than the prior new communications service.

[0066] FIG. 4 illustrates an exemplary method of initiating a call. The call may be initiated over a communications service previously allocated and activated as described with reference to FIG. 3. The call may be initiated by a calling party and directed to a called party. The calling party may include a first communications service endpoint at a first communications device, which may include an example of the communications device 102. The first communications service endpoint may include a first application, which may include an example of the application 206. The called party may include a second communications service endpoint at a second communications device, which may include another example of the communications device 102. The second communications service endpoint may include a second application, which may include another example of the application 206. Protocol signaling and data traffic may be communicated over a network, such as the network 104, between the first communications service endpoint and the second communications service endpoint serving to the communications protocol associated with the communications service.

[0067] Protocol signaling between the calling party and the called party may include data packets sent and received over the network 104 in accordance with the protocol to perform communications functions such as initiating a call, acknowledging a call, accepting a call, rejecting a call, putting a call on hold, resuming a call previously put on hold, initiating a conference call, terminating a call, requesting contact information, sending contact information, requesting presence information or status, sending presence information or status, sending text messages, sending multimedia messages, and so forth.

[0068] In step 402, a call initiation is signaled. The first application may send a call initiation command to the communications services interface module 208 specifying a called party to call and a communications service to use to call the called party. The call initiation command may be sent using an API function call, which may include an instance of the one or more API calls 220. The API function call may have a consistent syntax regardless of which communications service, communications channel, or communications protocol may be associated with the call. API function call parameters may be specified to correspond with the communications service, communications channel, or communications protocol associated with the call.

[0069] In step 404, a protocol stack associated with the call is commanded to initiate the call. The protocol stack associated with the call may be chosen among the members of the protocol stacks 210 based on the communications service and/or the communications channel specified by the first application or associated with the called party. The communications services interface module 208 may then send a protocol call initiation command to the protocol stack associated with the call. The protocol call initiation command may include an instance of the protocol commands 224. Syntax of the protocol call initiation command may depend upon, or be responsive to, the protocol stack associated with the call. For example, the protocol call initiation command may differ in syntax than if a different member of the protocol stacks 210 was associated with the call, the communications service, and/or the communications channel.

[0070] In step 406, the protocol stack associated with the call performs protocol signaling to call the called party. The protocol signaling may include sending a call initiation data packet corresponding to the protocol associated with the call over the network 104 to the called party. The protocol signaling may also include monitoring the network 104 for an acknowledgement data packet that may be sent from the called party to the calling party to acknowledge receipt of the call initiation data packet.

[0071] In step 408, the protocol signaling of the called party is reported. After the protocol stack associated with the call has sent the data packet to the called party in step 406, the protocol stack associated with the call may send a protocol signaling event, which may include an instance of the protocol events 228, to the communications services interface module 208. The protocol signaling event may report that the protocol stack associated with the call has begun signaling the called party to initiate the call.

[0072] In step 410, calling the called party is reported. The communications services interface module 208 may send a calling event, which may include an instance of the one or more API events 222, to the first application to report that the protocol stack associated with the call has signaled the called party to initiate the call. The calling event may be sent to the first application in response to receiving the protocol signaling event from the protocol stack associated with the call indicating that the protocol stack associated with the call has begun protocol signaling to initiate the call.

[0073] In step 412, the protocol stack associated with the call’s receipt of an acknowledgement from the called party is reported. The received acknowledgement may indicate that the called party is in receipt of the request to initiate the call and is performing protocol signaling in response to the request. The protocol signaling being performed by the called party may include sending an acknowledgement data packet to the calling party. The calling party’s protocol stack associated with the call may process the acknowledgement data packet received from the called party and send a protocol acknowledgement event, which may include an instance of the protocol events 228, to the communications services interface module 208. The protocol acknowledgement event may report that the called party has acknowledged the initiation of the call.
In step 414, the called party's acknowledgement of the call is reported. The communications services interface module 208 may send an interface acknowledgement event, which may include an instance of the one or more API events 222, to the first application to report that the called party has acknowledged receipt of the call initiation. The interface acknowledgement event may be sent to the first application in response to receiving the protocol acknowledgement event sent from the protocol stack associated with the call in step 412. As an illustration, the reported acknowledgement may correspond to a called telephone ringing in the earpiece of a calling telephone after dialing a telephone number.

In step 416, the protocol stack associated with the call's receipt of an acceptance of the call from the called party is reported. The called party may send a call acceptance data packet to the calling party over the network 104 to tell the calling party that the call is being accepted. The calling party's protocol stack associated with the call may process the call acceptance data packet received from the called party and send a protocol call acceptance event, which may include an instance of the protocol events 228, to the communications services interface module 208. The protocol call acceptance event may report that the called party accepts the call from the calling party.

In step 418, the communications services interface module 208 reports that the called party accepts the call. The communications services interface module 208 may send an interface call acceptance event, which may include an instance of the one or more API events 222, to the first application to report that the called party accepts the call from the calling party. The event may be sent to the first application in response to receiving the protocol call acceptance event sent from the protocol stack associated with the call in step 416.

FIG. 5 illustrates an exemplary method of terminating the call. The method illustrated in FIG. 5 assumes that the call between the calling party and the called party has already been initiated as described with reference to FIG. 4. A method similar to that illustrated in FIG. 5 may also be used for another desired communications function, such as placing the call on hold and resuming the call previously placed on hold, by customizing the commands, events, and data-packets to correspond to the other desired communications function. In this exemplary method, the calling party terminates the call in progress. The method may also be applicable to the called party terminating the call in progress.

In step 502, termination of the call in progress is signaled. The first application may send a call termination command to the communications services interface module 208 specifying that the call with the called party be terminated. The call termination command may be sent using an API function call, such as an instance of the one or more API calls 220. The API function call may have a consistent syntax regardless of which member of the protocol stacks 210 is associated with the call in progress.

In step 504, the protocol stack associated with the call in progress is commanded to terminate the call in progress. The communications services interface module 208 may determine which member of the protocol stacks 210 is associated with the call in progress. The communications services interface module 208 may then send a call termination command to the protocol stack associated with the call in progress. The call termination command may include an instance of the protocol commands 224. Syntax of the call termination command may depend upon, or be responsive to, the protocol stack associated with the call in progress. For example, the termination command to be sent to the protocol stack associated with the call in progress may differ in syntax than if a different member of the protocol stacks 210 was associated with the call in progress.

In step 506, protocol signaling with the called party is performed. The protocol stack associated with the call in progress may perform protocol signaling including sending a call termination data packet corresponding to the protocol associated with the call in progress over the network 104 to the called party. The call termination data packet may notify the called party that the call in progress is being terminated. The protocol signaling may also include monitoring the network 104 for an acknowledgement data packet that may be sent from the called party to the calling party to acknowledge receipt of the call termination data packet terminating the call.

In step 508, termination of the call in progress by the protocol stack associated with the call in progress is reported. After the protocol stack associated with the call in progress has sent the call termination data packet to the called party in step 506, the protocol stack associated with the call in progress may send a protocol call termination event, which may include an instance of the protocol events 228, to the communications services interface module 208. The protocol call termination event may report that the protocol stack associated with the call in progress has begun or completed signaling the called party to terminate the call. In some embodiments, the protocol call termination event may be sent to the communications services interface module 208 to report a termination error.

In step 510, termination of the call in progress is reported. The communications services interface module 208 may send an interface call termination event, which may include an instance of the one or more API events 222, to the first application to report that the protocol stack associated with the call in progress has terminated the call with the called party. The interface call termination event may be sent to the first application in response to receiving the protocol call termination event from the protocol stack associated with the call in progress in step 508.

FIG. 6 illustrates an exemplary method of accepting an incoming call. The incoming call may be accepted by the called party in response to the call initiation from the calling party, as described with reference to FIG. 4. In this exemplary method, the called party may include the second communications service endpoint and the second application at the second communications device.

In step 602, protocol signaling for the incoming call is received. A protocol stack associated with the incoming call, which may include a member of the protocol stacks 210, may monitor the network 104 for the call initiation data packet to arrive. The call initiation data packet may correspond to a call initiation protocol signal. When the protocol stack associated with the incoming call recognizes the call initiation protocol signal received over the network 104 as the incoming call initiation data packet, the protocol stack associated with the incoming call may receive the call initiation data packet for processing.

In step 604, a protocol incoming call event, which may include an instance of the protocol events 228, corresponding to the incoming call is generated. The protocol stack associated with the incoming call that receives the incoming call initiation data packet in step 602 may process the incoming call initiation data packet. If the protocol stack associated
with the incoming call determines that the incoming call initiation data packet represents the incoming call corresponding to the communications service or communications channel supported by the protocol stack associated with the incoming call, the protocol stack associated with the incoming call may send the protocol incoming call event to the communications services interface module 208. Syntax of the protocol incoming call event may depend upon, or be responsive to, the protocol stack associated with the incoming call. For example, the protocol incoming call event may differ in syntax than if a different member of the protocol stacks 210 is associated with the incoming call, the communications service, and/or the communications channel.

In step 606, the second application is notified of the incoming call. The communications services interface module 208 may receive the protocol incoming call event sent in step 604. The communications services interface module 208 may process the protocol incoming call event, effectively converting the protocol incoming call event whose syntax may be unique to the originating protocol stack associated with the incoming call into a protocol-independent incoming call event to be sent to the second application. The communications services interface module 208 may then send the protocol-independent incoming call event, which may include an instance of the one or more API events 222, to the second application to notify the second application of the incoming call. The protocol-independent incoming call event sent to the second application may be in a consistent syntax regardless of which member of the protocol stacks 210 is associated with the incoming call.

In step 608, the incoming call is acknowledged. The second application may send an acknowledgement command to the communications services interface module 208 specifying that an acknowledgement be sent to the calling party. The acknowledgement command may be sent using an API function call, such as an instance of the one or more API calls 220. The acknowledgement command may have a consistent syntax regardless of which member of the protocol stacks 210 is associated with the incoming call.

In step 610, a protocol acknowledgement command to acknowledge the incoming call is generated. The communications services interface module 208 may send the protocol acknowledgement command to the protocol stack associated with the incoming call. The protocol acknowledgement command may include an instance of the protocol commands 224. Syntax of the protocol acknowledgement command may depend upon, or be responsive to, the protocol stack associated with the incoming call. For example, the protocol acknowledgement command may differ in syntax than if a different member of the protocol stacks 210 was associated with the incoming call.

In step 612, the incoming call is accepted. The second application may send an accept call command to the communications services interface module 208 specifying that the incoming call be accepted. The accept call command may be sent using an API function call, which may include an instance of the one or more API calls 220. The API function call may have a consistent syntax regardless of which member of the protocol stacks 210 is associated with the incoming call.

In step 614, a protocol accept call command to accept the incoming call is generated. The communications services interface module 208 may send the protocol accept call command to the protocol stack associated with the incoming call. The protocol accept call command may include an instance of the protocol commands 224. Syntax of the protocol accept call command may depend upon, or be responsive to, the protocol stack associated with the incoming call. For example, the protocol accept call command may differ in syntax than if a different member of the protocol stacks 210 was associated with the incoming call. The protocol stack associated with the incoming call may then signal the calling party to accept the incoming call over the network 104.

FIG. 7 illustrates an exemplary method of providing an interface between a communications application and a plurality of communications protocols. The method includes providing a common function call as part of an API in source code for the communications application, which may include an example of the application 206, configuring a compiler for a target communications device or processor, and compiling the communications application source code into communications application executable code including the interface.

In step 702, the common function call is included in the communications application source code. The common function call may be part of the application programming interface 216, described with reference to FIG. 2. The communications application source code may use a common function call, which may correspond to an instance of the one or more API calls 220, for a first protocol stack as a second protocol stack. The first protocol stack and the second protocol stack may each include a member of the protocol stacks 210, and may each be different from one another. The first protocol stack or the second protocol stack may require that a protocol command, which may correspond to one of the protocol commands 224, be in syntax specific to the first protocol stack or the second protocol stack, respectively. The protocol command syntax for the first protocol stack may be different and incompatible with the protocol command syntax for the second protocol stack.

In step 704, a first target communications device, which may include an instance of the communications device 102, is targeted by a software compiler. The software compiler and one or more associated function libraries may be configured to compile the communications application source code into executable object code targeted to the first target communications device. The associated function libraries may include libraries associated with the application programming interface 216. The software compiler and/or the associated function libraries may include an option to target the executable object code to one or more different communications devices 102. The option may include a specification of a computing processor and/or an operating system associated with the one or more different communications devices 102.

In step 706, the communications application executable code is compiled. The software compiler may compile the communications application source code into the communications application executable code using the associated function libraries on a computing processor. The communications application executable code may include executable API calls, which may include instances of the one or more API calls 220, to interface to the communications services interface module 208. The communications services interface module 208 may be configured to issue an instance of the protocol commands 224 to a member of the protocol stacks 210.
210, in response to the executable API call to be received from the communications application executable code.

[0095] A syntax of the instance of the protocol commands 224 may depend upon, or be responsive to, the member of the protocol stacks 210 to which the instance of the protocol commands 224 is issued. For example, the instance of the protocol commands 224 to be issued to one member of the protocol stacks 210 may differ in syntax from another instance of the protocol commands 224 to be issued to a different member of the protocol stacks 210. Therefore, the communications services interface module 208 converts a single common function call syntax as received from the application 206 into a plurality of protocol command syntaxes responsive to the member of the protocol stacks 210 to which the instance of the protocol commands 224 is to be issued. This conversion enables the communications application source code to be independent of the details of each individual protocol supported by the communications services interface module 208.

[0096] In optional step 708, steps 704 and 706 may be repeated for one or more additional target communications devices. Step 708 facilitates the creation of a plurality of different communications application executable codes from a single communications application source code. A first of the plurality of communications application executable codes may execute on the first target communications device while a second and/or subsequent communications application executable code created for a different target communications device than the first communications application executable code may execute on a different target communications device.

[0097] FIG. 8 illustrates an exemplary method of providing communications between multiple entities. The multiple entities may include a central entity, a first entity, and a second entity, where the central entity may be communicatively coupled with both the first entity and the second entity. The communications may include voice communications protocols such as XMPP, SIP, GSM, etc. The communications may also include other protocols such as text communications protocols and multimedia communications protocols. The entities may include communications devices such as the communications device 102. The communications between the multiple entities may be carried over a network such as the network 104.

[0098] In step 802, a first voice communications protocol is utilized with the first entity. The central entity and the first entity may establish a first communications channel therebetween utilizing the first voice communications protocol. The first voice communications protocol may be implemented using a protocol stack, such as a member of the protocol stacks 210, on an example of the communications device 102.

[0099] In step 804, a second voice communications protocol is utilized with the second entity. The central entity and the second entity may establish a second communications channel therebetween utilizing the second voice communications protocol. The second voice communications protocol may be different from the first communications protocol. In other embodiments, the second voice communications protocol may be the same as the first communications protocol. The first voice communications channel and the second voice communications channel may be carried over different communications services providers, such as communications services providers 110. For example, the first voice communications channel may be carried over a cellular telephone service provider, while the second voice communications channel may be carried over a voice over internet protocol service provider or an internet service provider.

[0100] In step 806, the first entity and the second entity are communicatively coupled. In one embodiment, a voice conference service may be provided between the central entity, the first entity, and the second entity, such that all three entities may communicate with each other via the voice conference service by communicatively coupling the first communications channel and the second communications channel. In some embodiments, one or more additional entities may join the voice conference. A new communications channel between the central entity and the one or more additional entities may use a different communications protocol or communications services provider 110 than either the first communications channel or the second communications channel.

[0101] In another embodiment, a voice call may be transferred. The central entity may receive and accept a call from the first entity, and transfer the call to the second entity. The call from the first entity may be received over the first communications channel using the first communications protocol, while the call may be transferred to the second entity over the second communications channel using the second communications protocol. In some embodiments, the first communications protocol and the second communications protocol are different. In other embodiments, the first communications protocol and the second communications protocol may be the same. The first voice communications channel and the second voice communications channel may be carried over different communications services providers 110.

[0102] In a further embodiment, a voice call may be forwarded. The central entity may receive the call initiation protocol signal from the first entity, and forward the call to the second entity without answering the call. The call initiation protocol signal from the first entity may be received over the first communications channel using the first communications protocol, while the call may be forwarded to the second entity over the second communications channel using the second communications protocol. In some embodiments, the first communications protocol and the second communications protocol may be different. In other embodiments, the first communications protocol and the second communications protocol may be the same. The first voice communications channel and the second voice communications channel may be carried over different communications services providers 110.

[0103] FIG. 9 illustrates an exemplary method of providing a communications services interface. The method illustrates how the communications services interface, which may include an example of the communications services interface module 208 discussed herein, may be used by a communications application, which may include an example of the application 206, to communicate over multiple communications channels. The multiple communications channels may communicate over a network, such as the network 104.

[0104] In step 902, the communications services interface module 208 receives a first application function call, which may include an instance of the one or more API calls 220, from the application 206. The first application function call may specify a first communications channel to which the first application function call pertains. The first communications channel may be controlled by a first module, which may include a member of the protocol stacks 210. The first module
may utilize a first module communications protocol to communicate. The first module communications protocol may include any communications protocol supported by the first module.

[0105] In step 904, the communications services interface module 208 sends a first protocol command, which may include an instance of the protocol commands 224, to the first module. The first protocol command may be configured to control communications over the first communications channel. The first protocol command may be responsive to, or configured according to, the first application function call received in step 902. The communications services interface module 208 may monitor a status of the first communications channel. The communications services interface module 208 may send a first status event, which may include an instance of the one or more API events 222, to the application 206 reporting the status of the first communications channel.

[0106] In step 906, the communications services interface module 208 receives a second application function call, which may include an instance of the one or more API calls 220, from the application 206. The second application function call may specify a second communications channel to which the second application function call pertains. The second communications channel may be controlled by a second module, which may include a member of the protocol stacks 210. The second module may utilize a second module communications protocol to communicate. The second module communications protocol may be any communications protocol supported by the second module.

[0107] In some embodiments, the first application function call may have a different syntax from the second application function call. In other embodiments, the first application function call and the second application function call may have a same syntax. In some embodiments, the first module may be a different module from the second module. In other embodiments, the first module and the second module may be a same module. In some embodiments, the first module communications protocol may include a different communications protocol from the second module communications protocol. In other embodiments, the first module communications protocol and the second module communications protocol may include a same communications protocol.

[0108] In step 908, the communications services interface module 208 sends a second protocol command, which may include an instance of the protocol commands 224, to the second module. The second protocol command may be configured to control communications over the second communications channel. The second protocol command may be responsive to, or configured according to, the second application function call received in step 906. The communications services interface module 208 may monitor a status of the second communications channel. The communications services interface module 208 may send a second status event, which may include an instance of the one or more API events 222, to the application 206 reporting the status of the second communications channel.

[0109] In some embodiments, the second protocol command may have a different syntax from the first protocol command. In other embodiments, the second protocol command and the first protocol command may have a same syntax. In some embodiments, the second communications channel may be carried by a different communications services provider, such as an instance of the one or more communications services providers 110, than the first communications channel. In other embodiments, the second communications channel may be carried by a same communications services provider 110 as the first communications channel.

[0110] FIG. 10 illustrates an exemplary communications device 1000. The communications device 1000 may comprise the communications device 102 according to some embodiments. The communications device 1000 comprises at least a communications interface 1002, a processor 1004, a memory 1006, and storage 1008, which are all coupled to a bus 1010. The bus 1010 provides communications between the communications interface 1002, the processor 1004, the memory 1006, and the storage 1008.

[0111] The communications interface 1002 may communicate with other communications devices (not shown) via a communications medium 1012. The processor 1004 executes instructions. The memory 1006 permanently or temporarily stores data. Some examples of the memory 1006 are RAM and ROM. The storage 1008 also permanently or temporarily stores data. Some examples of the storage 1008 are hard disks and disk drives.

[0112] The embodiments discussed herein are illustrative. As these embodiments are described with reference to illustrations, various modifications or adaptations of the methods and/or specific structures described may become apparent to those skilled in the art.

[0113] The above-described components and functions can be comprised of instructions that are stored on a computer-readable storage medium. The instructions can be retrieved and executed by a processor (e.g., processor 1004). Some examples of instructions are software, program code, and firmware. Some examples of storage medium are memory devices, tape, disks, integrated circuits, and servers. The instructions are operational when executed by the processor to direct the processor to operate in accord with the invention. Those skilled in the art are familiar with instructions, processors(s), and storage medium.

[0114] While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. For example, any of the elements associated with the application programming interface may employ any of the desired functionality set forth hereinabove. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above-described exemplary embodiments.

What is claimed is:

1. A method for providing a communications services interface, the method comprising:
   receiving a first application function call specifying a first communications channel utilizing a first communications protocol to communicate over a network;
   sending a first protocol command to control communications over the first communications channel, the first protocol command being responsive to the first application function call;
   receiving a second application function call specifying a second communications channel utilizing a second communications protocol to communicate over the network, the second application function call having a same syntax as the first application function call, the second communications protocol being different than the first communications protocol; and
   sending a second protocol command to control communications over the second communications channel, the
second protocol command being responsive to the second application function call.

2. The method of claim 1, further comprising monitoring a status of the first and second communications channels and reporting an event corresponding to the first or second communications channel.

3. The method of claim 2, wherein the status of the first and the second communications channels includes a presence status.

4. The method of claim 1, wherein the first communications protocol includes a cellular telephone voice communications protocol and the second communications protocol includes a voice over internet communications protocol.

5. The method of claim 1, wherein the first communications channel utilizes a different communications service provider than the second communications channel.

6. The method of claim 1, wherein the first communications protocol includes a text communications protocol.

7. The method of claim 6, wherein the second communications protocol includes a voice communications protocol.

8. The method of claim 1, further comprising controlling streaming video over the network.

9. The method of claim 1, further comprising controlling streaming audio over the network.

10. The method of claim 1, further comprising providing a text chat room service providing text communications over the first communications channel and over the second communications channel, wherein the text communications sent over the first communications channel are also sent over the second communications channel.

11. The method of claim 10, wherein the first communications protocol is different than the second communications protocol.

12. The method of claim 1, wherein the first communications protocol includes an SIP protocol and the second communications protocol includes an XMPP protocol.

13. A method comprising:
utilizing a first voice communications protocol to provide voice communications between a central entity and a first entity;
utilizing a second voice communications protocol to provide voice communications between the central entity and a second entity, the second voice communications protocol being different from the first voice communications protocol; and
communicatively coupling the first entity and the second entity.

14. The method of claim 13, wherein the first communications channel and the second communications channel are carried over different communications service providers.

15. The method of claim 13, wherein the first voice communications protocol includes a cellular service protocol and the second voice communications protocol includes a voice over internet protocol.

16. The method of claim 13, wherein the first voice communications protocol includes an SIP protocol and the second voice communications protocol includes an XMPP protocol.

17. The method of claim 13, wherein communicatively coupling the first entity and the second entity includes providing a voice conference service between the central entity, the first entity, and the second entity.

18. The method of claim 13, wherein communicatively coupling the first entity and the second entity includes transferring a voice call from the first entity to the second entity.

19. The method of claim 13, wherein communicatively coupling the first entity and the second entity includes forwarding a voice call from the first entity to the second entity.

20. A method comprising:
including an application programming interface function call in an application source code, the application programming interface function call configured to selectively utilize either a first communications protocol or a second communications protocol, the selection of communications protocol to utilize being responsive to a parameter setting of the application programming interface function call, the first protocol and the second protocol being different;
targeting a compiler to compile the application source code into an executable application configured to execute on a first processor;
compiling the application source code into an executable application targeted to execute on the first processor, the compilation being performed using the compiler;
targeting the compiler to compile the application source code into an executable application configured to execute on a second processor; and
compiling the application source code into an executable application targeted to execute on the second processor, the compilation being performed using the compiler.

21. A system for providing communications services comprising:
a communications device including
a transceiver configured to transmit and receive communication signals,
a memory configured to store computer-readable instructions,
a processor configured to read the computer-readable instructions from the memory,
execute the computer-readable instructions, and
communicate with other communications devices using the transceiver; and
a program including computer-readable instructions stored on the memory, the program being executable by the processor for performing a method for providing communications services interface, the method comprising:
receiving a first application function call specifying a first communications channel utilizing a first communications protocol to communicate over a network;
sending a first protocol command to control communications over the first communications channel, the first protocol command being responsive to the first application function call;
receiving a second application function call specifying a second communications channel utilizing a second communications protocol to communicate over the network, the second application function call having a same syntax as the first application function call, the second communications protocol being different than the first communications protocol; and
sending a second protocol command to control communications over the second communications channel, the second protocol command being responsive to the second application function call.

22. A computer readable storage medium having stored thereon a program, the program being executable by a processor for performing a method for providing communications services interface, the method comprising:
receiving a first application function call specifying a first communications channel utilizing a first communications protocol to communicate over a network;
sending a first protocol command to control communications over the first communications channel, the first protocol command being responsive to the first application function call;
receiving a second application function call specifying a second communications channel utilizing a second communications protocol to communicate over the network, the second application function call having a same syntax as the first application function call, the second communications protocol being different than the first communications protocol; and
sending a second protocol command to control communications over the second communications channel, the second protocol command being responsive to the second application function call.
23. The computer readable storage medium of claim 22, the method further comprising monitoring a status of the first and second communications channels and reporting an event corresponding to the first or second communications channel.
24. The computer readable storage medium of claim 23, wherein the status of the first and second communications channels includes a presence status.
25. The computer readable storage medium of claim 22, wherein the first communications protocol includes a cellular telephone voice communications protocol and the second communications protocol includes a voice over internet communications protocol.
26. The computer readable storage medium of claim 22, wherein the first communications channel utilizes a different communications service provider than the second communications channel.
27. The computer readable storage medium of claim 22, wherein the first communications protocol includes a text communications protocol.
28. The computer readable storage medium of claim 27, wherein the second communications protocol includes a voice communications protocol.
29. The computer readable storage medium of claim 22, the method further comprising controlling streaming video over the network.
30. The computer readable storage medium of claim 22, the method further comprising controlling streaming audio over the network.
31. The computer readable storage medium of claim 22, the method further comprising providing a text chat room service providing text communications over the first communications channel and over the second communications channel, wherein the text communications sent over the first communications channel are also sent over the second communications channel.
32. The computer readable storage medium of claim 31, wherein the first communications protocol is different than the second communications protocol.
33. The computer readable storage medium of claim 22, wherein the first communications protocol includes an SIP protocol and the second communications protocol includes an XMPP protocol.
34. A method for providing a communications services interface, the method comprising:
receiving a first application function call specifying a first communications channel to communicate over a network;
sending a first protocol command to control communications over the first communications channel, the first protocol command being responsive to the first application function call;
receiving a second application function call specifying a second communications channel to communicate over the network, the second application function call having a same syntax as the first application function call, the second communications channel being carried by a different communications services provider than the first communications channel; and
sending a second protocol command to control communications over the second communications channel, the second protocol command being responsive to the second application function call.
35. A method for providing a communications services interface, the method comprising:
registering a first protocol stack with a communications services interface module;
activating a first communications service using the first protocol stack; and
registering a second protocol stack with the communications services interface module without interrupting operation of the first communications service.
36. The method of claim 35, wherein the second protocol stack is configured to support a second communications protocol which the communications services interface module did not previously support.
37. The method of claim 35, further comprising activating a second communications service using the second protocol stack without interrupting operation of the first communications service.
38. The method of claim 35, further comprising deactivating the second protocol stack without interrupting operation of the first communications service.
39. The method of claim 38, wherein deactivating the second protocol stack is responsive to availability of communications resources.
40. The method of claim 35, wherein registering the second protocol stack is responsive to availability of communications resources.

* * * * *