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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 October 2023 (12.10.2023)

WIPO | PCT

(10) International Publication Number
WO 2023/193162 A1

(51) International Patent Classification:
G06F 9/451 (2018.01)

(21) International Application Number:
PCT/CN2022/085449

(22) International Filing Date:
07 April 2022 (07.04.2022)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **CITRIX SYSTEMS, INC.** [US/US]; 851 West Cypress Creek Road, Fort Lauderdale, Florida 33309 (US).

(72) Inventor; and

(71) Applicant *(for SC only)*: **JIANG, Tianze** [CN/CN]; Building C3, No. 19 Suyuan Avenue, Nanjing, Jiangsu 211100 (CN).

(72) Inventor: **XIN, Yu**; Building C3, No. 19 Suyuan Avenue, Nanjing, Jiangsu 211100 (CN).

(74) Agent: **CHINA PATENT AGENT (H.K.) LTD.**; 22/F., Great Eagle Center, 23 Harbour Road, Wanchai, Hong Kong (CN).

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

WO 2023/193162 A1

(54) Title: COMPUTING DEVICE AND METHODS PROVIDING ENHANCED LANGUAGE DETECTION AND DISPLAY FEATURES FOR VIRTUAL COMPUTING SESSIONS

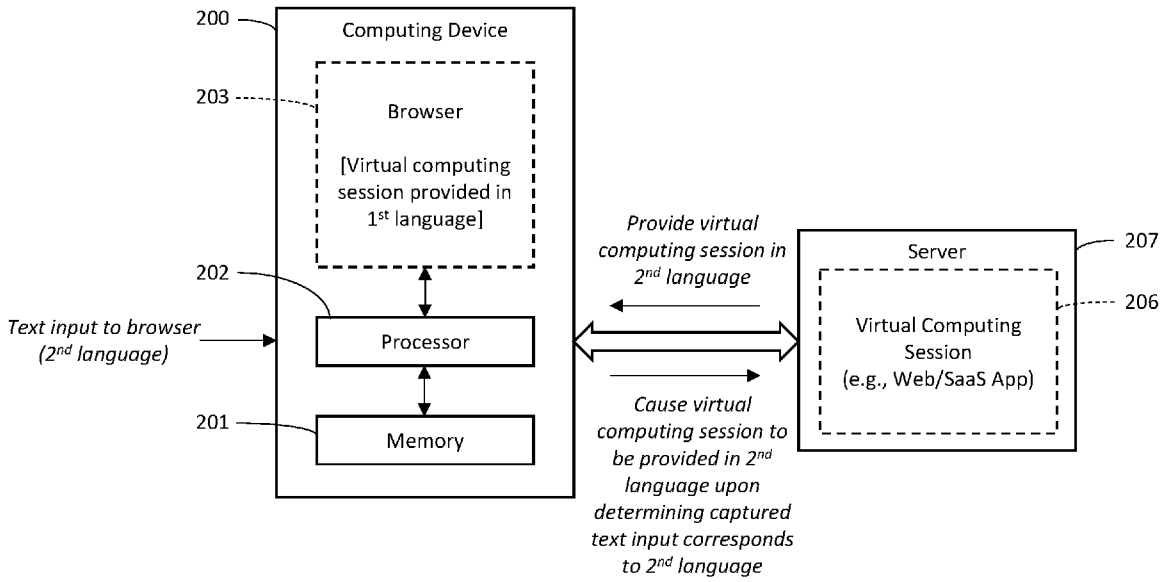

FIG. 6

(57) Abstract: A computing device may include a memory and a processor coupled to the memory and configured to run a browser to access a virtual computing session. The browser may have a first language associated therewith, and the virtual computing session may be provided in the first language responsive to the browser. The processor may be further configured to capture text input to the browser for the virtual computing session, determine whether the captured text corresponds to a second language different than the first language, and cause the virtual computing session to be provided in the second language responsive to determining the captured text corresponds to the second language.

[Continued on next page]

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

Published:

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

COMPUTING DEVICE AND METHODS PROVIDING ENHANCED LANGUAGE DETECTION AND DISPLAY FEATURES FOR VIRTUAL COMPUTING SESSIONS

Background

[0001] Web applications or apps are software programs that run on a server and are accessed remotely by client devices through a Web browser. That is, while Web applications have a similar functionality to native applications installed directly on the client device, Web applications are instead installed and run on the server, and only the browser application is installed on the client device. Although in some implementations, a hosted browser running on a virtualization server may be used to access Web applications as well.

[0002] One advantage of using Web applications is that this allows client devices to run numerous different applications without having to install all of these applications on the client device. This may be particularly beneficial for thin client devices, which typically have reduced memory and processing capabilities. Moreover, updating Web applications may be easier than native applications, as updating is done at the server level rather than having to push out updates to numerous different types of client devices.

[0003] Software as a Service (SaaS) is a Web application licensing and delivery model in which applications are delivered remotely as a web-based service, typically on a subscription basis. SaaS is used for delivering several different types of business (and other) applications, including office, database, accounting, customer relation management (CRM), etc.

Summary

[0004] A computing device may include a memory and a processor coupled to the memory and configured to run a browser to access a virtual computing session. The browser may have a first language associated therewith, and the virtual computing session may be provided in the first language responsive to the browser. The processor may be further configured to capture text input to the browser for the virtual computing session, determine whether the captured text corresponds to a second language different than the first language, and cause the virtual computing session to be provided in the second language responsive to determining the captured text corresponds to the second language.

[0005] In an example embodiment, the processor may be further configured to determine if the second language is on a supported language list for the virtual computing session, and to cause the virtual computing session to be provided in the second language further based upon determining the second language is on the supported language list. In accordance with another example, the processor may be further configured to cause the browser to display an input element responsive to determining the captured text corresponds to the second language, and to cause the virtual computing session to be provided in the second language further based upon input received via the input element.

[0006] In some implementations, the browser may have an Application Programming Interface (API) associated therewith, and the processor may be configured to capture the text input via the API, for example. In accordance with another example implementation, the processor may be further configured to run an Input Method Editor (IME), and to capture the text input to the browser from the IME.

[0007] By way of example, the virtual computing session may comprise at least one of a Web application and a Software as a Service (SaaS) application. In one example implementation, the processor may be further configured to communicate the captured text to a language detection server, and to determine whether the captured text corresponds to the second language responsive to the language detection server.

[0008] A related method may include running a browser at a computing device to access a virtual computing session, with the browser having a first language associated therewith, and the virtual computing session being provided in the first language responsive to the browser. The method may also include capturing text input to the browser at the computing device for the virtual computing session, determining, at the computing device, whether the captured text corresponds to a second language different than the first language, and causing the virtual computing session to be provided in the second language at the computing device responsive to determining the captured text corresponds to the second language.

[0009] A related non-transitory computer-readable medium may have computer-executable instructions for causing a computing device to perform steps including running a browser to access a virtual computing session, with the browser having a first language associated therewith, and the virtual computing session being provided in the first language responsive to the browser. The steps may further include capturing text input to the browser for the virtual computing session, determining whether the captured text corresponds to a second language different than the first language, and causing the virtual computing session to be provided in the second language responsive to determining the captured text corresponds to the second language.

Brief Description of the Drawings

[0010] FIG. 1 is a schematic block diagram of a network environment of computing devices in which various aspects of the disclosure may be implemented.

[0011] FIG. 2 is a schematic block diagram of a computing device useful for practicing an embodiment of the client machines or the remote machines illustrated in FIG. 1.

[0012] FIG. 3 is a schematic block diagram of a cloud computing environment in which various aspects of the disclosure may be implemented.

[0013] FIG. 4 is a schematic block diagram of desktop, mobile and web-based devices operating a workspace app in which various aspects of the disclosure may be implemented.

[0014] FIG. 5 is a schematic block diagram of a workspace network environment of computing devices in which various aspects of the disclosure may be implemented.

[0015] FIG. 6 is a schematic block diagram of a computing device providing enhanced language detection and display features in accordance with an example embodiment.

[0016] FIG. 7 is a schematic block diagram illustrating an example implementation of the computing device of FIG. 6.

[0017] FIGS. 8 and 9A-9B are flow diagram illustrating example method aspects associated with FIGS. 6 and 7.

Detailed Description

[0018] Globalization is important for a software product to provide services for customers around the world. In this regard, it is advantageous to set the display language for a software User Interface (UI) according to the user's convention when an app is installed or launched (startup). However, for

Web/Software as a Service (SaaS) apps which run on remote servers and are accessed through browsers, the display language is sometimes not as expected for various reasons. For example, the browser language may follow the system locale during installation by default, which is then adopted by Web/SaaS apps even though this is not the language preferred by the user. While browsers may provide language settings for users to switch between different languages, this may not be easy or intuitive for all users. Furthermore, browsers typically cannot read current system locale/keyboard information due to limitations of internal Application Programming Interfaces (APIs), whereas the current system locale/keyboard layout is more likely to be set to a user's preference. Another reason is that, for users working on shared computers or kiosks, these types of computers may record the preferences of the last user.

[0019] The approach set forth herein overcomes these technical challenges by capturing text input to the browser running a Web/SaaS app in virtual computing session, and determining whether the captured text corresponds to a second language different than a first (e.g., default) language of the browser. As a result, the Web/SaaS app may instead automatically be provided in the second language preferred by the user. That is, this approach allows for dynamic changing of languages displayed by Web/SaaS apps based upon detecting user keyboard/IME input via the browser running the Web/SaaS app.

[0020] Referring initially to FIG. 1, a non-limiting network environment 10 in which various aspects of the disclosure may be implemented includes one or more client machines 12A-12N, one or more remote machines 16A-16N, one or more networks 14, 14', and one or more appliances 18 installed within the computing

environment 10. The client machines 12A-12N communicate with the remote machines 16A-16N via the networks 14, 14’.

[0021] In some embodiments, the client machines 12A-12N communicate with the remote machines 16A-16N via an intermediary appliance 18. The illustrated appliance 18 is positioned between the networks 14, 14’ and may also be referred to as a network interface or gateway. In some embodiments, the appliance 108 may operate as an application delivery controller (ADC) to provide clients with access to business applications and other data deployed in a data center, the cloud, or delivered as Software as a Service (SaaS) across a range of client devices, and/or provide other functionality such as load balancing, etc. In some embodiments, multiple appliances 18 may be used, and the appliance(s) 18 may be deployed as part of the network 14 and/or 14’.

[0022] The client machines 12A-12N may be generally referred to as client machines 12, local machines 12, clients 12, client nodes 12, client computers 12, client devices 12, computing devices 12, endpoints 12, or endpoint nodes 12. The remote machines 16A-16N may be generally referred to as servers 16 or a server farm 16. In some embodiments, a client device 12 may have the capacity to function as both a client node seeking access to resources provided by a server 16 and as a server 16 providing access to hosted resources for other client devices 12A-12N. The networks 14, 14’ may be generally referred to as a network 14. The networks 14 may be configured in any combination of wired and wireless networks.

[0023] A server 16 may be any server type such as, for example: a file server; an application server; a web server; a proxy server; an appliance; a network appliance; a gateway; an application gateway; a gateway server; a virtualization server;

a deployment server; a Secure Sockets Layer Virtual Private Network (SSL VPN) server; a firewall; a web server; a server executing an active directory; a cloud server; or a server executing an application acceleration program that provides firewall functionality, application functionality, or load balancing functionality.

[0024] A server 16 may execute, operate or otherwise provide an application that may be any one of the following: software; a program; executable instructions; a virtual machine; a hypervisor; a web browser; a web-based client; a client-server application; a thin-client computing client; an ActiveX control; a Java applet; software related to voice over internet protocol (VoIP) communications like a soft IP telephone; an application for streaming video and/or audio; an application for facilitating real-time-data communications; a HTTP client; a FTP client; an Oscar client; a Telnet client; or any other set of executable instructions.

[0025] In some embodiments, a server 16 may execute a remote presentation services program or other program that uses a thin-client or a remote-display protocol to capture display output generated by an application executing on a server 16 and transmit the application display output to a client device 12.

[0026] In yet other embodiments, a server 16 may execute a virtual machine providing, to a user of a client device 12, access to a computing environment. The client device 12 may be a virtual machine. The virtual machine may be managed by, for example, a hypervisor, a virtual machine manager (VMM), or any other hardware virtualization technique within the server 16.

[0027] In some embodiments, the network 14 may be: a local-area network (LAN); a metropolitan area network (MAN); a wide area network (WAN); a primary public network 14; and a primary

private network 14. Additional embodiments may include a network 14 of mobile telephone networks that use various protocols to communicate among mobile devices. For short range communications within a wireless local-area network (WLAN), the protocols may include 802.11, Bluetooth, and Near Field Communication (NFC).

[0028] FIG. 2 depicts a block diagram of a computing device 20 useful for practicing an embodiment of client devices 12, appliances 18 and/or servers 16. The computing device 20 includes one or more processors 22, volatile memory 24 (e.g., random access memory (RAM)), non-volatile memory 30, user interface (UI) 38, one or more communications interfaces 26, and a communications bus 48.

[0029] The non-volatile memory 30 may include: one or more hard disk drives (HDDs) or other magnetic or optical storage media; one or more solid state drives (SSDs), such as a flash drive or other solid-state storage media; one or more hybrid magnetic and solid-state drives; and/or one or more virtual storage volumes, such as a cloud storage, or a combination of such physical storage volumes and virtual storage volumes or arrays thereof.

[0030] The user interface 38 may include a graphical user interface (GUI) 40 (e.g., a touchscreen, a display, etc.) and one or more input/output (I/O) devices 42 (e.g., a mouse, a keyboard, a microphone, one or more speakers, one or more cameras, one or more biometric scanners, one or more environmental sensors, and one or more accelerometers, etc.).

[0031] The non-volatile memory 30 stores an operating system 32, one or more applications 34, and data 36 such that, for example, computer instructions of the operating system 32 and/or the applications 34 are executed by processor(s) 22 out of the

volatile memory 24. In some embodiments, the volatile memory 24 may include one or more types of RAM and/or a cache memory that may offer a faster response time than a main memory. Data may be entered using an input device of the GUI 40 or received from the I/O device(s) 42. Various elements of the computer 20 may communicate via the communications bus 48.

[0032] The illustrated computing device 20 is shown merely as an example client device or server, and may be implemented by any computing or processing environment with any type of machine or set of machines that may have suitable hardware and/or software capable of operating as described herein.

[0033] The processor(s) 22 may be implemented by one or more programmable processors to execute one or more executable instructions, such as a computer program, to perform the functions of the system. As used herein, the term "processor" describes circuitry that performs a function, an operation, or a sequence of operations. The function, operation, or sequence of operations may be hard coded into the circuitry or soft coded by way of instructions held in a memory device and executed by the circuitry. A processor may perform the function, operation, or sequence of operations using digital values and/or using analog signals.

[0034] In some embodiments, the processor can be embodied in one or more application specific integrated circuits (ASICs), microprocessors, digital signal processors (DSPs), graphics processing units (GPUs), microcontrollers, field programmable gate arrays (FPGAs), programmable logic arrays (PLAs), multi-core processors, or general-purpose computers with associated memory.

[0035] The processor 22 may be analog, digital or mixed-signal. In some embodiments, the processor 22 may be one or more

physical processors, or one or more virtual (e.g., remotely located or cloud) processors. A processor including multiple processor cores and/or multiple processors may provide functionality for parallel, simultaneous execution of instructions or for parallel, simultaneous execution of one instruction on more than one piece of data.

[0036] The communications interfaces 26 may include one or more interfaces to enable the computing device 20 to access a computer network such as a Local Area Network (LAN), a Wide Area Network (WAN), a Personal Area Network (PAN), or the Internet through a variety of wired and/or wireless connections, including cellular connections.

[0037] In described embodiments, the computing device 20 may execute an application on behalf of a user of a client device. For example, the computing device 20 may execute one or more virtual machines managed by a hypervisor. Each virtual machine may provide an execution session within which applications execute on behalf of a user or a client device, such as a hosted desktop session. The computing device 20 may also execute a terminal services session to provide a hosted desktop environment. The computing device 20 may provide access to a remote computing environment including one or more applications, one or more desktop applications, and one or more desktop sessions in which one or more applications may execute.

[0038] An example virtualization server 16 may be implemented using Citrix Hypervisor provided by Citrix Systems, Inc., of Fort Lauderdale, Florida ("Citrix Systems"). Virtual app and desktop sessions may further be provided by Citrix Virtual Apps and Desktops (CVAD), also from Citrix Systems. Citrix Virtual Apps and Desktops is an application virtualization solution that enhances productivity with universal access to virtual sessions

including virtual app, desktop, and data sessions from any device, plus the option to implement a scalable VDI solution. Virtual sessions may further include Software as a Service (SaaS) and Desktop as a Service (DaaS) sessions, for example.

[0039] Referring to FIG. 3, a cloud computing environment 50 is depicted, which may also be referred to as a cloud environment, cloud computing or cloud network. The cloud computing environment 50 can provide the delivery of shared computing services and/or resources to multiple users or tenants. For example, the shared resources and services can include, but are not limited to, networks, network bandwidth, servers, processing, memory, storage, applications, virtual machines, databases, software, hardware, analytics, and intelligence.

[0040] In the cloud computing environment 50, one or more clients 52A-52C (such as those described above) are in communication with a cloud network 54. The cloud network 54 may include backend platforms, e.g., servers, storage, server farms or data centers. The users or clients 52A-52C can correspond to a single organization/tenant or multiple organizations/tenants. More particularly, in one example implementation the cloud computing environment 50 may provide a private cloud serving a single organization (e.g., enterprise cloud). In another example, the cloud computing environment 50 may provide a community or public cloud serving multiple organizations/ tenants. In still further embodiments, the cloud computing environment 50 may provide a hybrid cloud that is a combination of a public cloud and a private cloud. Public clouds may include public servers that are maintained by third parties to the clients 52A-52C or the enterprise/tenant. The servers may be located off-site in remote geographical locations or otherwise.

[0041] The cloud computing environment 50 can provide

resource pooling to serve multiple users via clients 52A-52C through a multi-tenant environment or multi-tenant model with different physical and virtual resources dynamically assigned and reassigned responsive to different demands within the respective environment. The multi-tenant environment can include a system or architecture that can provide a single instance of software, an application or a software application to serve multiple users. In some embodiments, the cloud computing environment 50 can provide on-demand self-service to unilaterally provision computing capabilities (e.g., server time, network storage) across a network for multiple clients 52A-52C. The cloud computing environment 50 can provide an elasticity to dynamically scale out or scale in responsive to different demands from one or more clients 52. In some embodiments, the computing environment 50 can include or provide monitoring services to monitor, control and/or generate reports corresponding to the provided shared services and resources.

[0042] In some embodiments, the cloud computing environment 50 may provide cloud-based delivery of different types of cloud computing services, such as Software as a service (SaaS) 56, Platform as a Service (PaaS) 58, Infrastructure as a Service (IaaS) 60, and Desktop as a Service (DaaS) 62, for example. IaaS may refer to a user renting the use of infrastructure resources that are needed during a specified time period. IaaS providers may offer storage, networking, servers or virtualization resources from large pools, allowing the users to quickly scale up by accessing more resources as needed. Examples of IaaS include AMAZON WEB SERVICES provided by Amazon.com, Inc., of Seattle, Washington, RACKSPACE CLOUD provided by Rackspace US, Inc., of San Antonio, Texas, Google Compute Engine provided by Google Inc. of Mountain View, California, or RIGHTSCALE provided

by RightScale, Inc., of Santa Barbara, California.

[0043] PaaS providers may offer functionality provided by IaaS, including, e.g., storage, networking, servers or virtualization, as well as additional resources such as, e.g., the operating system, middleware, or runtime resources. Examples of PaaS include WINDOWS AZURE provided by Microsoft Corporation of Redmond, Washington, Google App Engine provided by Google Inc., and HEROKU provided by Heroku, Inc. of San Francisco, California.

[0044] SaaS providers may offer the resources that PaaS provides, including storage, networking, servers, virtualization, operating system, middleware, or runtime resources. In some embodiments, SaaS providers may offer additional resources including, e.g., data and application resources. Examples of SaaS include GOOGLE APPS provided by Google Inc., SALESFORCE provided by Salesforce.com Inc. of San Francisco, California, or OFFICE 365 provided by Microsoft Corporation. Examples of SaaS may also include data storage providers, e.g. DROPBOX provided by Dropbox, Inc. of San Francisco, California, Microsoft SKYDRIVE provided by Microsoft Corporation, Google Drive provided by Google Inc., or Apple ICLOUD provided by Apple Inc. of Cupertino, California.

[0045] Similar to SaaS, DaaS (which is also known as hosted desktop services) is a form of virtual desktop infrastructure (VDI) in which virtual desktop sessions are typically delivered as a cloud service along with the apps used on the virtual desktop. Citrix Cloud is one example of a DaaS delivery platform. DaaS delivery platforms may be hosted on a public cloud computing infrastructure such as AZURE CLOUD from Microsoft Corporation of Redmond, Washington (herein "Azure"), or AMAZON WEB SERVICES provided by Amazon.com, Inc., of Seattle,

Washington (herein "AWS"), for example. In the case of Citrix Cloud, Citrix Workspace app may be used as a single-entry point for bringing apps, files and desktops together (whether on-premises or in the cloud) to deliver a unified experience.

[0046] The unified experience provided by the Citrix Workspace app will now be discussed in greater detail with reference to FIG. 4. The Citrix Workspace app will be generally referred to herein as the workspace app 70. The workspace app 70 is how a user gets access to their workspace resources, one category of which is applications. These applications can be SaaS apps, web apps or virtual apps. The workspace app 70 also gives users access to their desktops, which may be a local desktop or a virtual desktop. Further, the workspace app 70 gives users access to their files and data, which may be stored in numerous repositories. The files and data may be hosted on Citrix ShareFile, hosted on an on-premises network file server, or hosted in some other cloud storage provider, such as Microsoft OneDrive or Google Drive Box, for example.

[0047] To provide a unified experience, all of the resources a user requires may be located and accessible from the workspace app 70. The workspace app 70 is provided in different versions. One version of the workspace app 70 is an installed application for desktops 72, which may be based on Windows, Mac or Linux platforms. A second version of the workspace app 70 is an installed application for mobile devices 74, which may be based on iOS or Android platforms. A third version of the workspace app 70 uses a hypertext markup language (HTML) browser to provide a user access to their workspace environment. The web version of the workspace app 70 is used when a user does not want to install the workspace app or does not have the rights to install the workspace app, such as when operating a public kiosk

76.

[0048] Each of these different versions of the workspace app 70 may advantageously provide the same user experience. This advantageously allows a user to move from client device 72 to client device 74 to client device 76 in different platforms and still receive the same user experience for their workspace. The client devices 72, 74 and 76 are referred to as endpoints.

[0049] As noted above, the workspace app 70 supports Windows, Mac, Linux, iOS, and Android platforms as well as platforms with an HTML browser (HTML5). The workspace app 70 incorporates multiple engines 80-90 allowing users access to numerous types of app and data resources. Each engine 80-90 optimizes the user experience for a particular resource. Each engine 80-90 also provides an organization or enterprise with insights into user activities and potential security threats.

[0050] An embedded browser engine 80 keeps SaaS and web apps contained within the workspace app 70 instead of launching them on a locally installed and unmanaged browser. With the embedded browser, the workspace app 70 is able to intercept user-selected hyperlinks in SaaS and web apps and request a risk analysis before approving, denying, or isolating access.

[0051] A high definition experience (HDX) engine 82 establishes connections to virtual browsers, virtual apps and desktop sessions running on either Windows or Linux operating systems. With the HDX engine 82, Windows and Linux resources run remotely, while the display remains local, on the endpoint. To provide the best possible user experience, the HDX engine 82 utilizes different virtual channels to adapt to changing network conditions and application requirements. To overcome high-latency or high-packet loss networks, the HDX engine 82 automatically implements optimized transport protocols and

greater compression algorithms. Each algorithm is optimized for a certain type of display, such as video, images, or text. The HDX engine 82 identifies these types of resources in an application and applies the most appropriate algorithm to that section of the screen.

[0052] For many users, a workspace centers on data. A content collaboration engine 84 allows users to integrate all data into the workspace, whether that data lives on-premises or in the cloud. The content collaboration engine 84 allows administrators and users to create a set of connectors to corporate and user-specific data storage locations. This can include OneDrive, Dropbox, and on-premises network file shares, for example. Users can maintain files in multiple repositories and allow the workspace app 70 to consolidate them into a single, personalized library.

[0053] A networking engine 86 identifies whether or not an endpoint or an app on the endpoint requires network connectivity to a secured backend resource. The networking engine 86 can automatically establish a full VPN tunnel for the entire endpoint device, or it can create an app-specific µ-VPN connection. A µ-VPN defines what backend resources an application and an endpoint device can access, thus protecting the backend infrastructure. In many instances, certain user activities benefit from unique network-based optimizations. If the user requests a file copy, the workspace app 70 can automatically utilize multiple network connections simultaneously to complete the activity faster. If the user initiates a VoIP call, the workspace app 70 improves its quality by duplicating the call across multiple network connections. The networking engine 86 uses only the packets that arrive first.

[0054] An analytics engine 88 reports on the user’s device,

location and behavior, where cloud-based services identify any potential anomalies that might be the result of a stolen device, a hacked identity or a user who is preparing to leave the company. The information gathered by the analytics engine 88 protects company assets by automatically implementing counter-measures.

[0055] A management engine 90 keeps the workspace app 70 current. This not only provides users with the latest capabilities, but also includes extra security enhancements. The workspace app 70 includes an auto-update service that routinely checks and automatically deploys updates based on customizable policies.

[0056] Referring now to FIG. 5, a workspace network environment 100 providing a unified experience to a user based on the workspace app 70 will be discussed. The desktop, mobile and web versions of the workspace app 70 all communicate with the workspace experience service 102 running within the Cloud 104. The workspace experience service 102 then pulls in all the different resource feeds 16 via a resource feed micro-service 108. That is, all the different resources from other services running in the Cloud 104 are pulled in by the resource feed micro-service 108. The different services may include a virtual apps and desktop service 110, a secure browser service 112, an endpoint management service 114, a content collaboration service 116, and an access control service 118. Any service that an organization or enterprise subscribes to are automatically pulled into the workspace experience service 102 and delivered to the user's workspace app 70.

[0057] In addition to cloud feeds 120, the resource feed micro-service 108 can pull in on-premises feeds 122. A cloud connector 124 is used to provide virtual apps and desktop

deployments that are running in an on-premises data center. Desktop virtualization may be provided by Citrix virtual apps and desktops 126, Microsoft RDS 128 or VMware Horizon 130, for example. In addition to cloud feeds 120 and on-premises feeds 122, device feeds 132 from Internet of Thing (IoT) devices 134, for example, may be pulled in by the resource feed micro-service 108. Site aggregation is used to tie the different resources into the user's overall workspace experience.

[0058] The cloud feeds 120, on-premises feeds 122 and device feeds 132 each provides the user's workspace experience with a different and unique type of application. The workspace experience can support local apps, SaaS apps, virtual apps, and desktops browser apps, as well as storage apps. As the feeds continue to increase and expand, the workspace experience is able to include additional resources in the user's overall workspace. This means a user will be able to get to every single application that they need access to.

[0059] Still referring to the workspace network environment 20, a series of events will be described on how a unified experience is provided to a user. The unified experience starts with the user using the workspace app 70 to connect to the workspace experience service 102 running within the Cloud 104, and presenting their identity (event 1). The identity includes a username and password, for example.

[0060] The workspace experience service 102 forwards the user’s identity to an identity micro-service 140 within the Cloud 104 (event 2). The identity micro-service 140 authenticates the user to the correct identity provider 142 (event 3) based on the organization’s workspace configuration. Authentication may be based on an on-premises active directory 144 that requires the deployment of a cloud connector 146.

Authentication may also be based on Azure Active Directory 148 or even a third party identity provider 150, such as Citrix ADC or Okta, for example.

[0061] Once authorized, the workspace experience service 102 requests a list of authorized resources (event 4) from the resource feed micro-service 108. For each configured resource feed 106, the resource feed micro-service 108 requests an identity token (event 5) from the single-sign micro-service 152.

[0062] The resource feed specific identity token is passed to each resource's point of authentication (event 6). On-premises resources 122 are contacted through the Cloud Connector 124. Each resource feed 106 replies with a list of resources authorized for the respective identity (event 7).

[0063] The resource feed micro-service 108 aggregates all items from the different resource feeds 106 and forwards (event 8) to the workspace experience service 102. The user selects a resource from the workspace experience service 102 (event 9).

[0064] The workspace experience service 102 forwards the request to the resource feed micro-service 108 (event 10). The resource feed micro-service 108 requests an identity token from the single sign-on micro-service 152 (event 11). The user's identity token is sent to the workspace experience service 102 (event 12) where a launch ticket is generated and sent to the user.

[0065] The user initiates a secure session to a gateway service 160 and presents the launch ticket (event 13). The gateway service 160 initiates a secure session to the appropriate resource feed 106 and presents the identity token to seamlessly authenticate the user (event 14). Once the session initializes, the user is able to utilize the resource (event 15). Having an entire workspace delivered through a single access

point or application advantageously improves productivity and streamlines common workflows for the user.

[0066] Turning now to FIG. 6, a computing device 200 provides for dynamic language detection and change for Web/SaaS apps based upon text input to the browser that is providing access to the Web/SaaS app session. The computing device 200 illustratively includes a memory 201 and a processor 202 coupled to the memory and configured to run a browser to access a virtual computing session 206, e.g., a Web/SaaS app session from a server 207. By way of example, the computing device 200 may take the form of a client computing device, such as a desktop computer, laptop computer, tablet computer, smartphone, etc. The processor 202 may take the form of a microprocessor and associated hardware, and the memory 201 may include non-transitory computer-readable instructions for causing the processor to perform the various operations discussed below. In some embodiments, the browser 203 may be a managed browser that is embedded at the computing device 200, or a hosted browser that is run remotely on a server, such as provided by Citrix Workspace and described further above. Generally speaking, a managed browsers provides an IT or virtual computing service provider enhanced abilities to set permission levels associated with the browser and/or access data input to the browser or received by the browser. However, a managed browser need not be used in all embodiments, and other approaches may be used to capture text input to standard browsers (e.g., Google Chrome, Microsoft Edge, etc.), as will be discussed further below.

[0067] By way of example, consider a German user who is working on a Windows 10 Desktop with the system locale set to de-DE (German, Germany). Yet, the Chrome browser installed on the personal computer (PC) is displayed with English by default.

In this case, when the user launches the Web UI to access a Web/SaaS app (e.g., Citrix Workspace Web UI), the UI language will be set to English (en-US) consistent with the browser default setting. However, the user would otherwise expect a German UI, and it is typically not easy or intuitive for the user to manually change the preferred language through the browser, especially if the menu options to do so are not in the user's native language.

[0068] Tuning now to an example use case described further with reference to FIG. 7, a language detection module 204 may be integrated into a Web/SaaS platform (e.g., Workspace UI in this example). When a user starts to work on the Web/SaaS app and inputs text with a keyboard, such as in a form field 205 generated by the browser 203, the language module 204 will be activated. The language detection module 204 will first capture the text input to the form field 205 with APIs provided by the browser. If the input is in a letter-based language like English, German, Spanish, etc., key events will be handled to extract the letters or words. If the input is done with client Input Method Editor (IME) software, such as for Chinese/Japanese/Korean users, composition events can be used to capture the characters being input.

[0069] After receiving the input text, it is sent to a language detection server 208 for language detection. Various language detection services or tools may be used for the language detection, such as Lingua, for example. In some embodiments, the language detection may be performed locally, instead of by a remote service, as well. The detection results will be returned from the language detection server 208 with a target language code or locale (Language=zh-CN in the present example). Then, the language detection module 204 will query the

target language in a supported language list of current Web apps, which may be stored in the memory 201 or by the server 207.

[0070] If the target language is supported and is not the same as the current (e.g., default) UI language, a pop-up window 209 is displayed in the present example which provides a message asking the user whether to switch to the target language (here, Chinese) for the current app. The user may accept or reject the switching operation as desired using the buttons provided in the pop-up window 209. However, in some embodiments, a pop-up window or prompt need not be used, and the language detection module may automatically trigger a change to the target language, if desired.

[0071] When the user confirms the switch to the target language, an i18n module 210 of the Web/SaaS app will update the current valid language. Moreover, the new language may be applied to the Web/SaaS app (e.g., SPA app) with the help of a frontend framework (e.g., React, Vue, etc.). In other instances, the Web/SaaS app may need to reload the page to fetch the new translated strings from the server 207, with the target language as a parameter. Once the new language has been chosen (or rejected) by the user, the language detection module 204 may suspend capturing text until needed again (e.g., upon the next start-up, etc.).

[0072] Turning now to the flow diagrams 280 and 290 of FIGS. 8, 9A and 9B, associated method aspects are now described. Beginning at Block 281, the computing device 200 runs a browser 203 to access the virtual computing session 206 (e.g., a Web or SaaS app), at Block 282. As noted above, the browser 203 will typically have a first (e.g., default) language associated therewith, and the virtual computing session 206 will be provided in the first language responsive to the browser. That

is, the virtual computing session 206 may detect the default language of the browser 203 upon start-up, and provide the virtual computing session automatically in the detected language.

[0073] The method further illustratively includes capturing text input to the browser 203 at the computing device 200 for the virtual computing session 206, at Block 283, and determining whether the captured text corresponds to a second language different than the first language, at Block 284. As noted above, this may be done through an API of the browser, whether from key events or composition events (e.g., via an IME). The computing device 200 may accordingly cause the virtual computing session 206 to be provided in the second language at the computing device responsive to determining the captured text corresponds to the second language, at Block 285. The method of FIG. 8 illustratively concludes at Block 286.

[0074] As also noted above, in some embodiments the computing device 200 may communicate the captured text to a language detection server 208, and determine whether the captured text corresponds to the second language responsive to the language detection server (e.g., from a language code provided by the language detection server for the detected language). Moreover, in some implementations the computing device 200 may be further configured to determine if the second language is on a supported language list for the virtual computing session 206, at Block 288, and if so to cause the virtual computing session to be provided in the second language responsive thereto. That is, the computing device 200 may first check to see if the language the user is attempting to use is in fact one that is supported by the virtual computing session 206 before requesting that the server 207 change the virtual computing session over to the second language.

[0075] As also discussed above, in some embodiments the computing device 200 may display an input element (e.g., the pop-up window 209) responsive to determining the captured text corresponds to the second language, at Blocks 289-290. As a result, the computing device 200 may cause the virtual computing session 206 to be provided in the second language further based upon input received via the input element, such as through UI buttons that allow the user to change to the detected language, or keep the current (e.g., default) language.

[0076] The above-described approach advantageously allows for dynamic changing of the display language for virtual computing sessions (e.g., Web/SaaS apps) accessed through a browser automatically according to the input characters provided to the browser. This provides for relatively easy adaptation of Web/SaaS application to the preferred display language of the user, which usually will correspond to the language the user is attempting to use to provide input to the browser.

[0077] As will be appreciated by one of skill in the art upon reading the foregoing disclosure, various aspects described herein may be embodied as a device, a method or a computer program product (e.g., a non-transitory computer-readable medium having computer executable instruction for performing the noted operations or steps). Accordingly, those aspects may take the form of an entirely hardware embodiment, an entirely software embodiment, or an embodiment combining software and hardware aspects.

[0078] Furthermore, such aspects may take the form of a computer program product stored by one or more computer-readable storage media having computer-readable program code, or instructions, embodied in or on the storage media. Any suitable computer readable storage media may be utilized, including hard

disks, CD-ROMs, optical storage devices, magnetic storage devices, and/or any combination thereof.

[0079] Many modifications and other embodiments will come to the mind of one skilled in the art having the benefit of the teachings presented in the foregoing descriptions and the associated drawings. Therefore, it is understood that the foregoing is not to be limited to the example embodiments, and that modifications and other embodiments are intended to be included within the scope of the appended claims.

CLAIMS:

1. A computing device comprising:

a memory and a processor coupled to the memory and configured to

run a browser to access a virtual computing session, the browser having a first language associated therewith, and the virtual computing session being provided in the first language responsive to the browser,

capture text input to the browser for the virtual computing session,

determine whether the captured text corresponds to a second language different than the first language, and

cause the virtual computing session to be provided in the second language responsive to determining the captured text corresponds to the second language.

2. The computing device of claim 1 wherein the processor is further configured to determine if the second language is on a supported language list for the virtual computing session, and to cause the virtual computing session to be provided in the second language further based upon determining the second language is on the supported language list.

3. The computing device of claim 1 wherein the processor is further configured to cause the browser to display an input element responsive to determining the captured text corresponds to the second language, and to cause the virtual

computing session to be provided in the second language further based upon input received via the input element.

4. The computing device of claim 1 wherein the browser has an Application Programming Interface (API) associated therewith, and wherein the processor is configured to capture the text input via the API.

5. The computing device of claim 1 wherein the processor is further configured to run an Input Method Editor (IME), and to capture the text input to the browser from the IME.

6. The computing device of claim 1 wherein the virtual computing session comprises at least one of a Web application and a Software as a Service (SaaS) application.

7. The computing device of claim 1 wherein the processor is further configured to communicate the captured text to a language detection server, and to determine whether the captured text corresponds to the second language responsive to the language detection server.

8. A method comprising:

running a browser at a computing device to access a virtual computing session, the browser having a first language associated therewith, and the virtual computing session being provided in the first language responsive to the browser;

capturing text input to the browser at the computing device for the virtual computing session;

determining, at the computing device, whether the captured text corresponds to a second language different than the first language; and

causing the virtual computing session to be provided in the second language at the computing device responsive to determining the captured text corresponds to the second language.

9. The method of claim 8 further comprising, at the computing device, determining if the second language is on a supported language list for the virtual computing session, and causing the virtual computing session to be provided in the second language further based upon determining the second language is on the supported language list.

10. The method of claim 8 further comprising, at the computing device, causing the browser to display an input element responsive to determining the captured text corresponds to the second language, and causing the virtual computing session to be provided in the second language further based upon input received via the input element.

11. The method of claim 8 wherein the browser has an Application Programming Interface (API) associated therewith, and wherein capturing the text input to the browser comprises capturing the text input to the browser via the API.

12. The method of claim 8 further comprising, at the computing device, running an Input Method Editor (IME), and wherein capturing the text input to the browser comprises capturing the text input to the browser from the IME.

13. The method of claim 8 wherein the virtual computing session comprises at least one of a Web application and a Software as a Service (SaaS) application.

14. The method of claim 8 further comprising, at the

computing device, communicating the captured text to a language detection server; and wherein determining whether the captured text corresponds to the second language comprises determining whether the captured text corresponds to the second language responsive to the language detection server.

15. A non-transitory computer-readable medium having computer-executable instructions for causing a computing device to perform steps comprising:

running a browser to access a virtual computing session, the browser having a first language associated therewith, and the virtual computing session being provided in the first language responsive to the browser;

capturing text input to the browser for the virtual computing session;

determining whether the captured text corresponds to a second language different than the first language; and

causing the virtual computing session to be provided in the second language responsive to determining the captured text corresponds to the second language.

16. The non-transitory computer-readable medium of claim 15 further having computer-executable instructions for causing the computing device to perform steps comprising:

determining if the second language is on a supported language list for the virtual computing session; and

causing the virtual computing session to be provided in the second language further based upon determining the second language is on the supported language list.

17. The non-transitory computer-readable medium of claim 15 further having computer-executable instructions for

causing the computing device to perform a step comprising causing the browser to display an input element responsive to determining the captured text corresponds to the second language; and wherein causing the virtual computing session to be provided in the second language is further based upon input received via the input element.

18. The non-transitory computer-readable medium of claim 15 wherein the browser has an Application Programming Interface (API) associated therewith, and wherein capturing the text input to the browser comprises capturing the text input to the browser via the API.

19. The non-transitory computer-readable medium of claim 15 further having computer-executable instructions for causing the computing device to perform a step comprising running an Input Method Editor (IME); and wherein capturing the text input to the browser comprises capturing the text input to the browser from the IME.

20. The non-transitory computer-readable medium of claim 15 further having computer-executable instructions for causing the computing device to perform a step comprising communicating the captured text to a language detection server; and wherein determining whether the captured text corresponds to the second language comprises determining whether the captured text corresponds to the second language responsive to the language detection server.

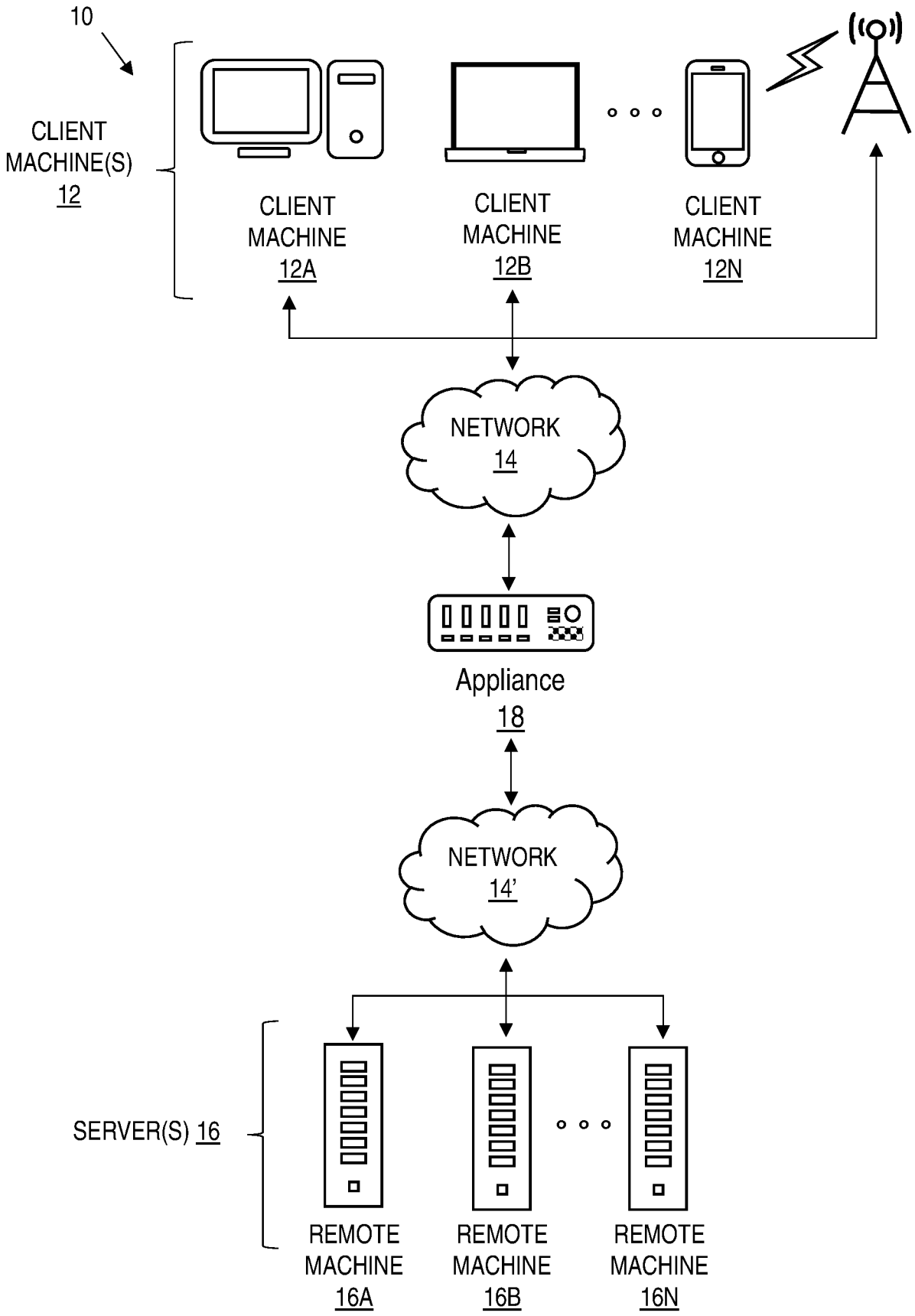
10
CLIENT MACHINE(S) 12
CLIENT MACHINE 12A
CLIENT MACHINE 12B
CLIENT MACHINE 12N
NETWORK 14
Appliance 18
NETWORK 14'
SERVER(S) 16
REMOTE MACHINE 16A
REMOTE MACHINE 16B
REMOTE MACHINE 16N

FIG. 1

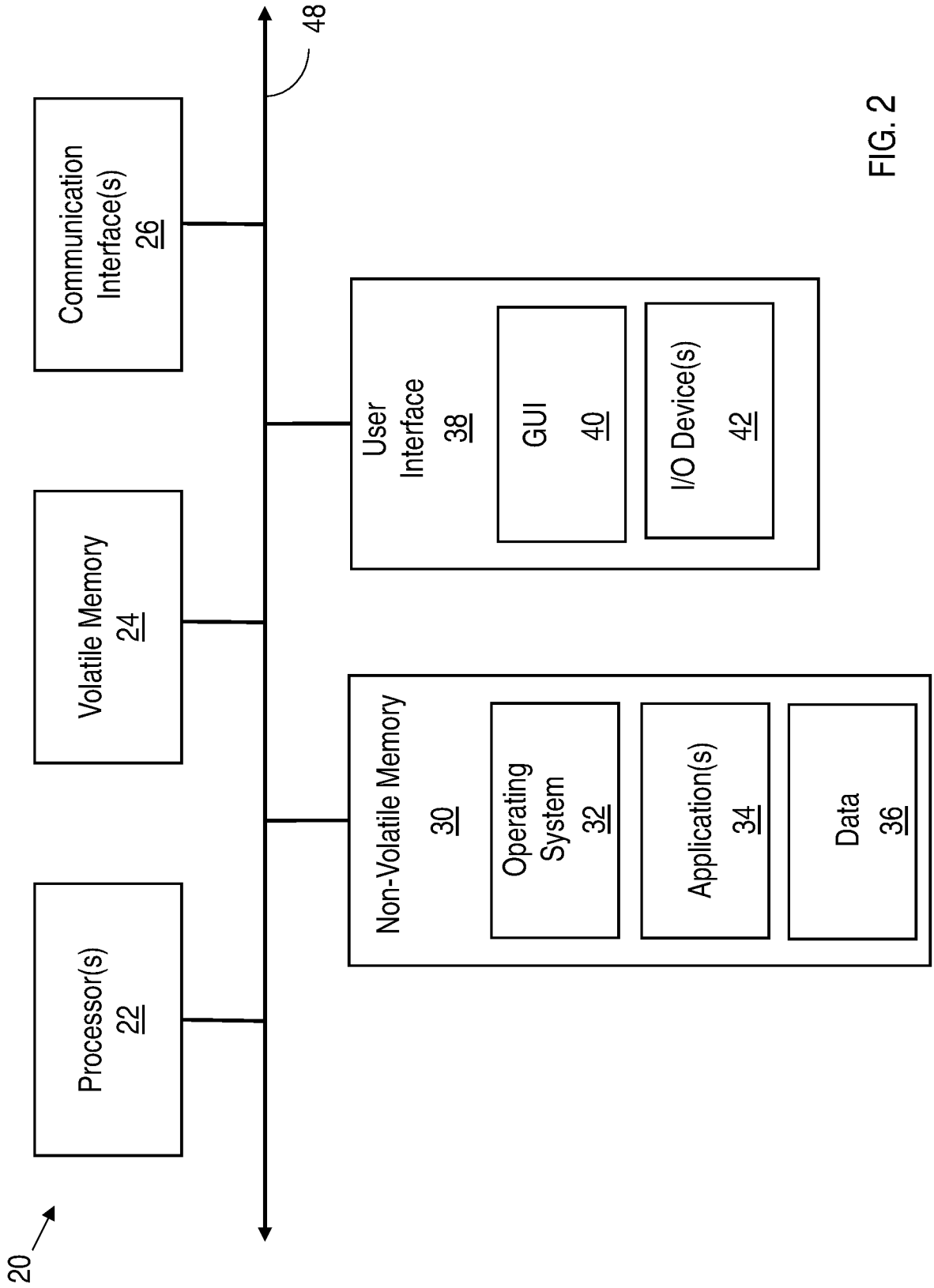
20
Processor(s)
22
Volatile Memory
24
Communication Interface(s)
26
48
Non-Volatile Memory
30
Operating System
32
Application(s)
34
Data
36
User Interface
38
GUI
40
I/O Device(s)
42

FIG. 2

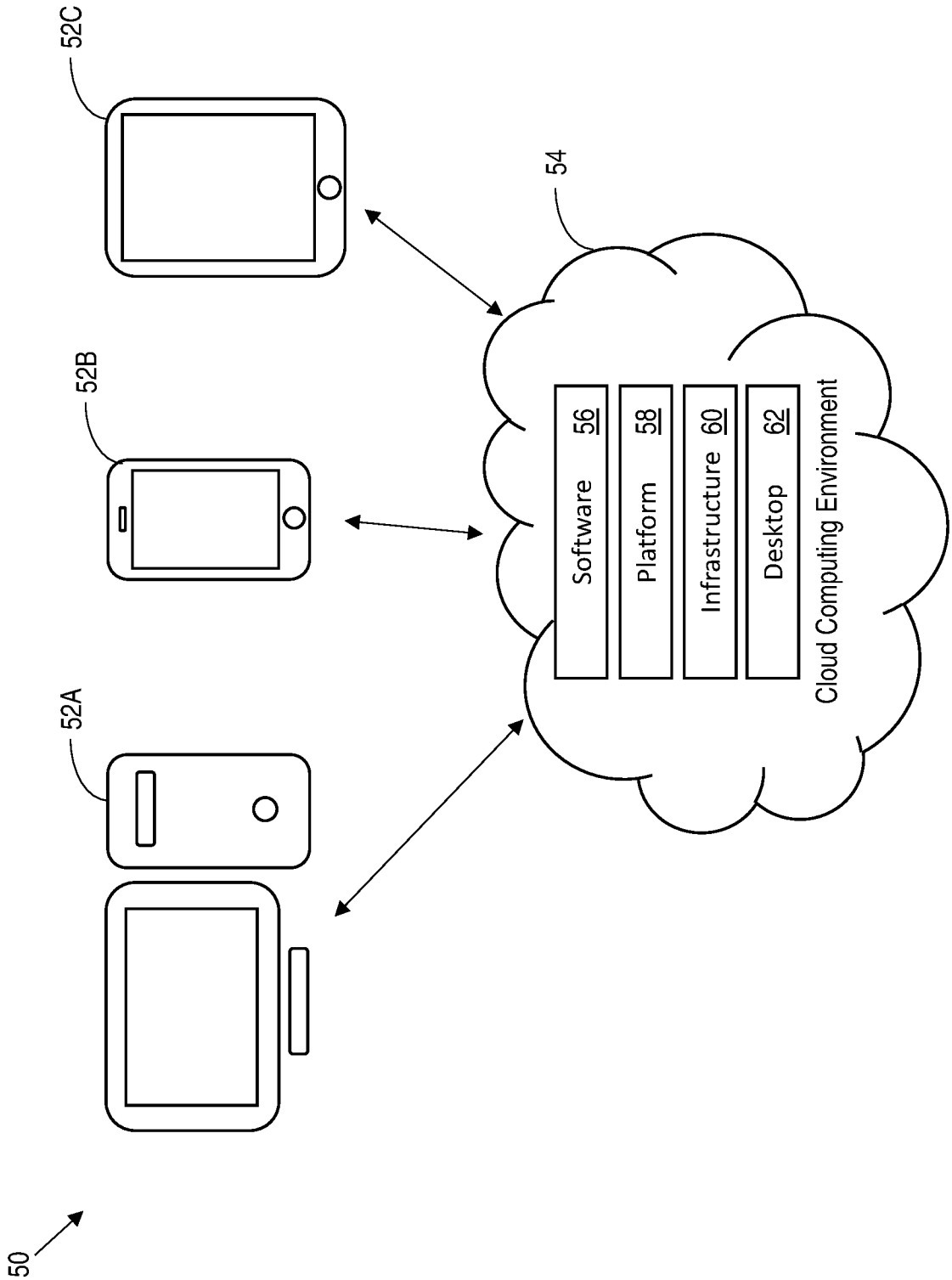
50
52A
52B
52C
54
Cloud Computing Environment
Software 56
Platform 58
Infrastructure 60
Desktop 62

FIG. 3

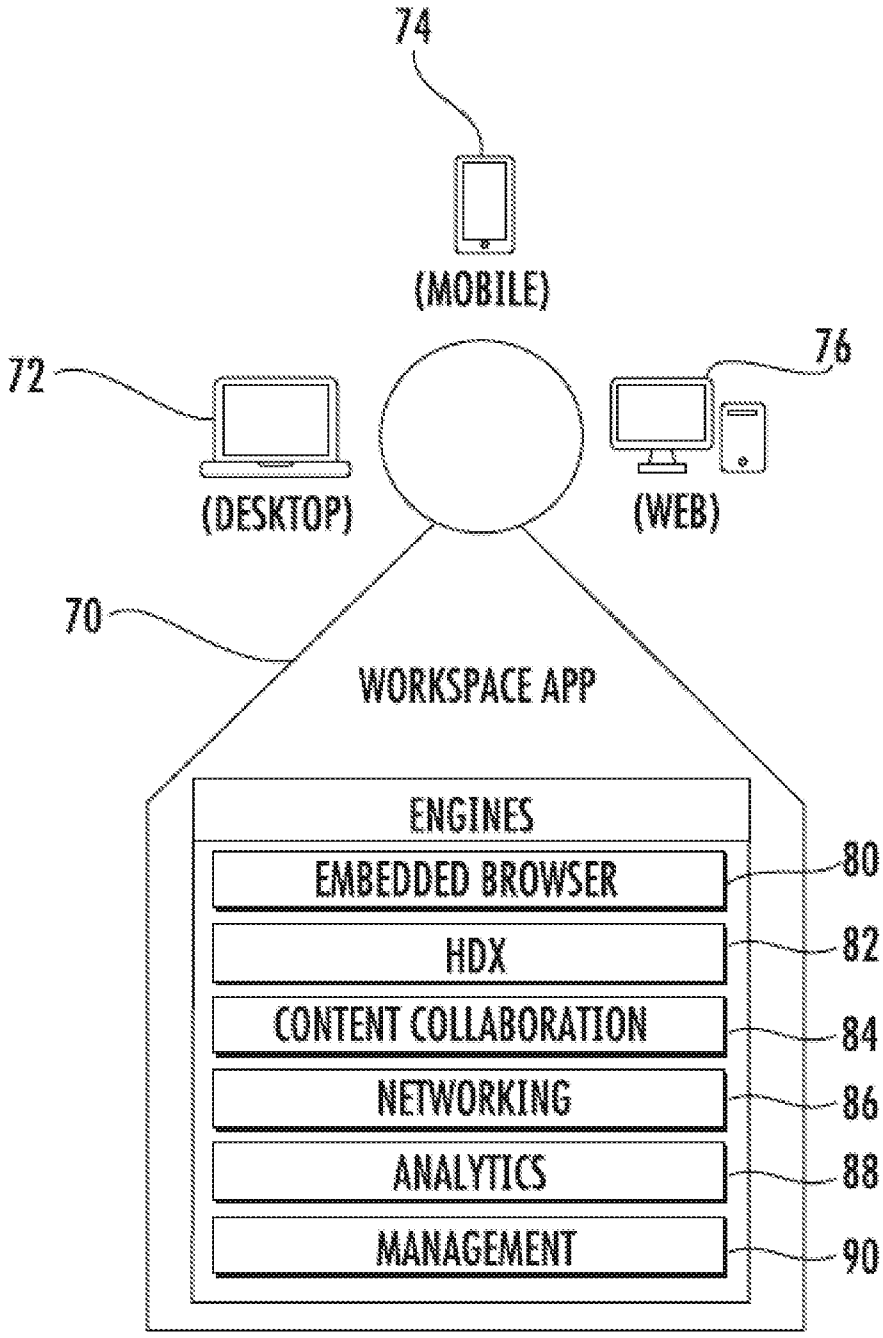
74
(MOBILE)
72
(DESKTOP)
76
(WEB)
70
WORKSPACE APP
ENGINES
EMBEDDED BROWSER
80
HDX
82
CONTENT COLLABORATION
84
NETWORKING
86
ANALYTICS
88
MANAGEMENT
90

FIG. 4

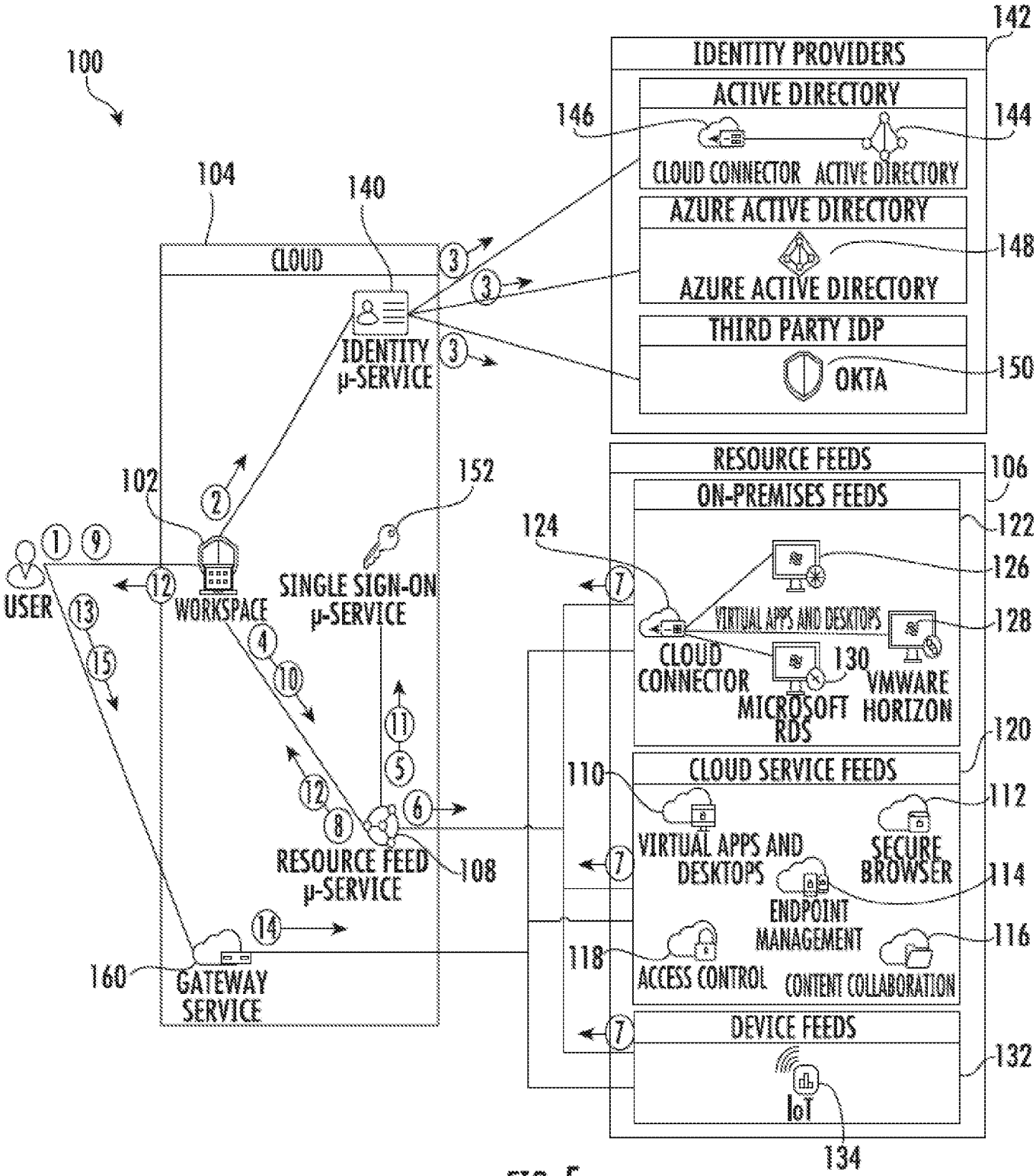
100
142
IDENTITY PROVIDERS
ACTIVE DIRECTORY
146
144
CLOUD CONNECTOR
ACTIVE DIRECTORY
AZURE ACTIVE DIRECTORY
148
AZURE ACTIVE DIRECTORY
THIRD PARTY IDP
150
OKTA
104
140
CLOUD
IDENTITY
μ-SERVICE
RESOURCE FEEDS
106
ON-PREMISES FEEDS
122
124
126
VIRTUAL APPS AND DESKTOPS
128
CLOUD CONNECTOR
130
MICROSOFT RDS
VMWARE HORIZON
120
CLOUD SERVICE FEEDS
110
112
VIRTUAL APPS AND DESKTOPS
SECURE BROWSER
114
ENDPOINT MANAGEMENT
116
118
ACCESS CONTROL
CONTENT COLLABORATION
DEVICE FEEDS
132
IoT
134
102
152
USER
WORKSPACE
SINGLE SIGN-ON
μ-SERVICE
RESOURCE FEED
μ-SERVICE
108
160
GATEWAY SERVICE
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

FIG. 5

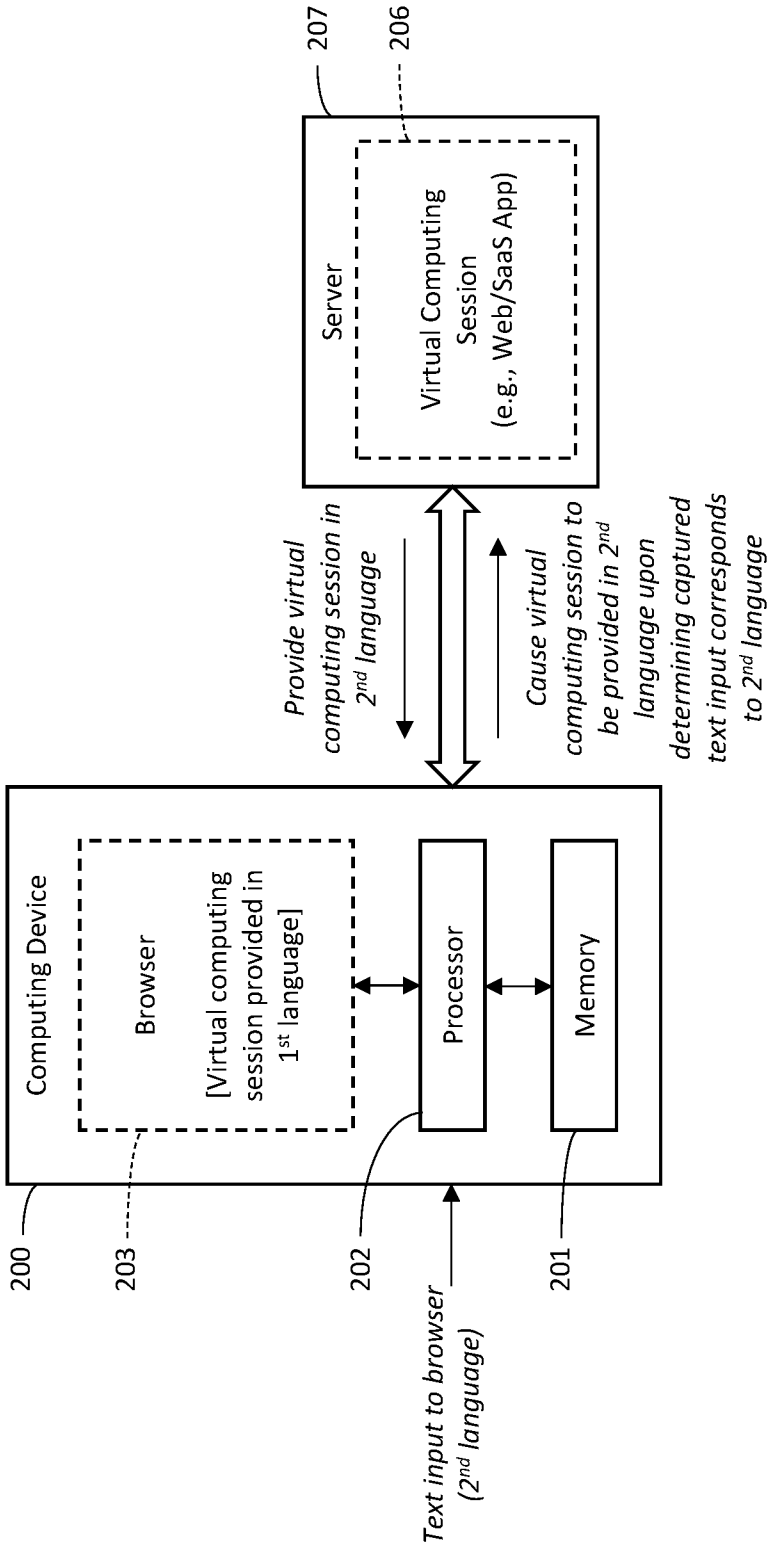
207
206
Server
Virtual Computing Session (e.g., Web/SaaS App)
Provide virtual computing session in 2nd language
Cause virtual computing session to be provided in 2nd language upon determining captured text input corresponds to 2nd language
Computing Device
Browser
[Virtual computing session provided in 1st language]
Processor
Memory
200
203
202
201
Text input to browser (2nd language)

FIG. 6

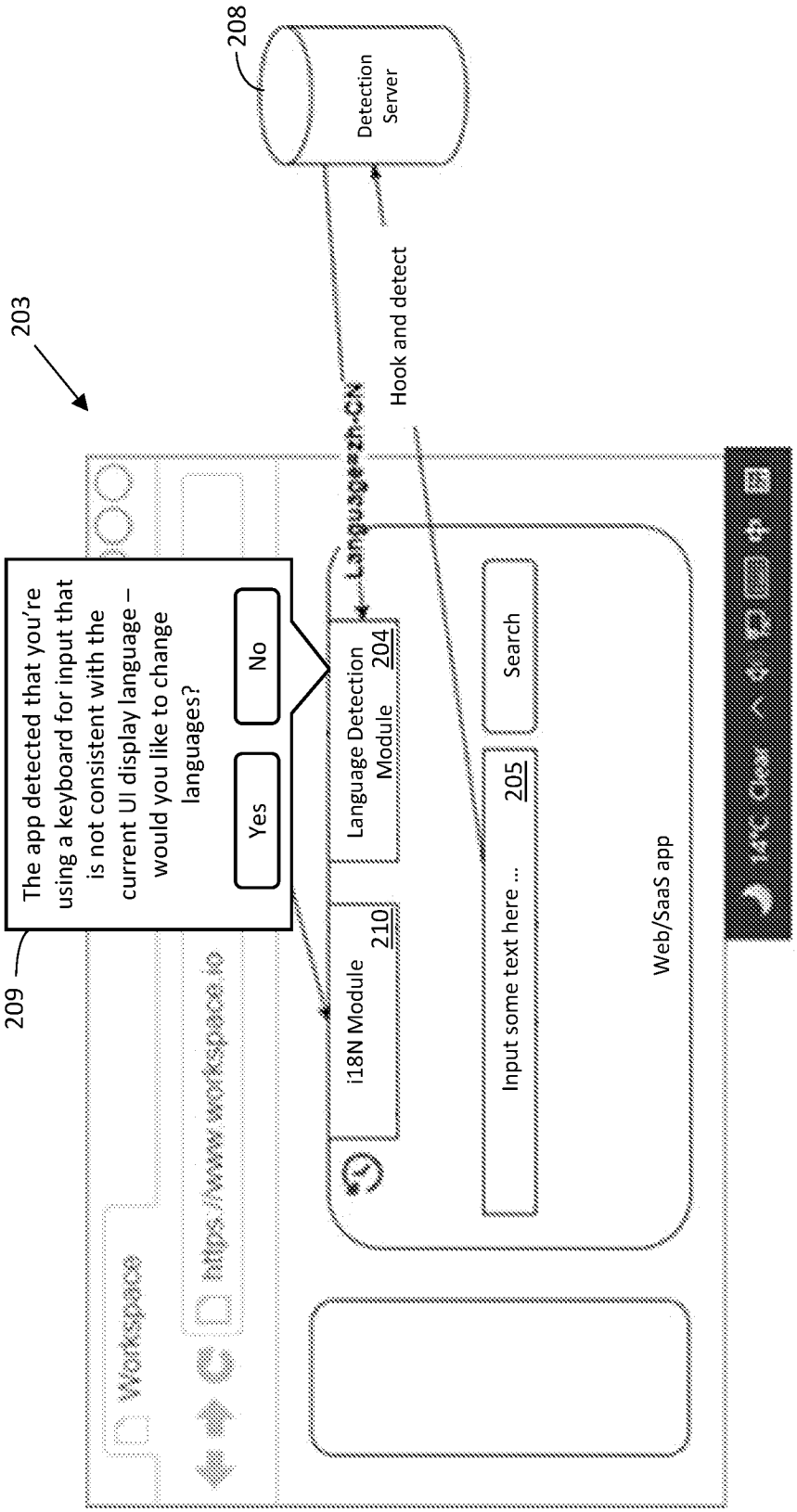
208
Detection Server
203
Hook and detect
Language Detection Module 204
The app detected that you're using a keyboard for input that is not consistent with the current UI display language – would you like to change languages?
Yes
No
209
i18N Module 210
Input some text here ... 205
Search
Web/SaaS app
Workspace
https://www.workspace.io

FIG. 7

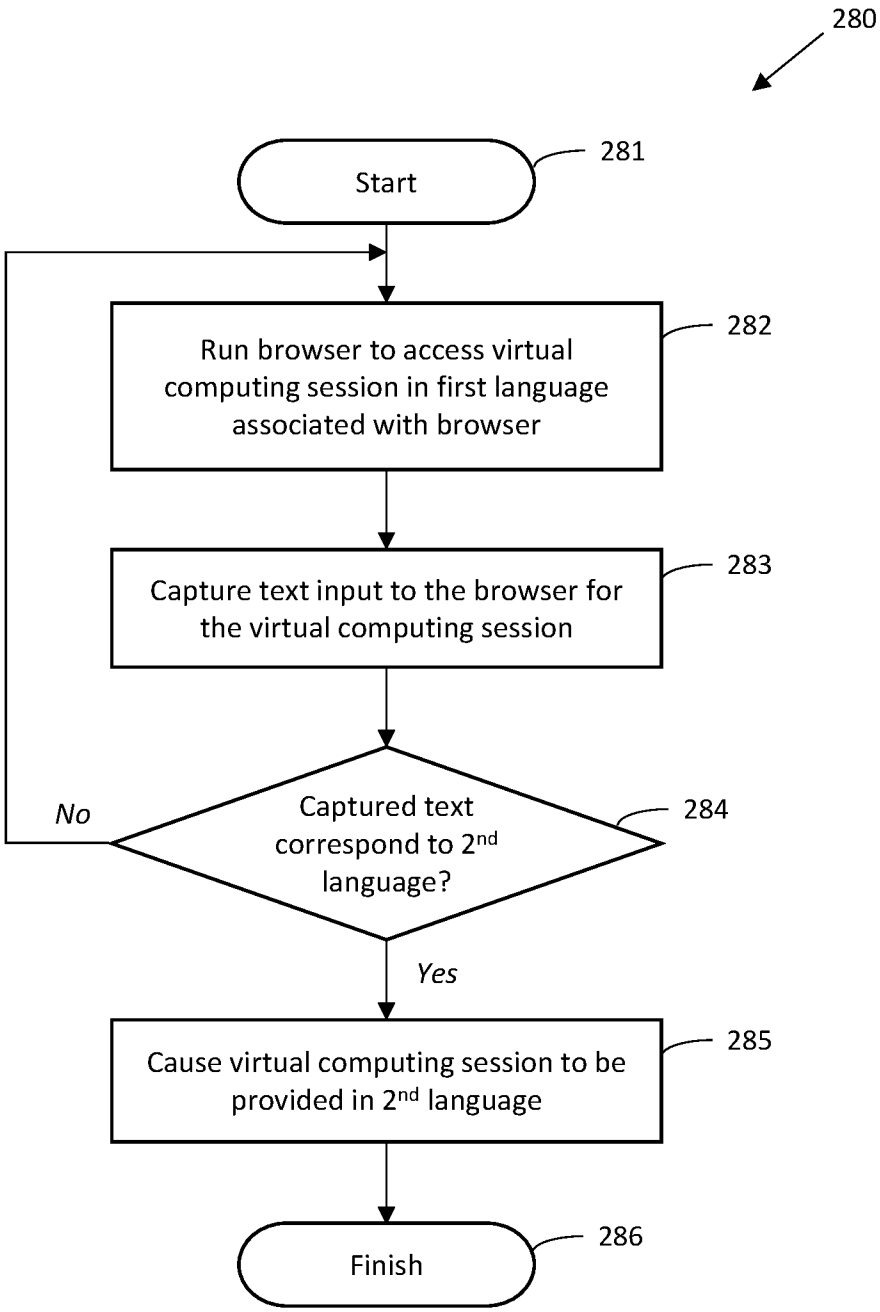
280
Start
281
Run browser to access virtual computing session in first language associated with browser
282
Capture text input to the browser for the virtual computing session
283
Captured text correspond to 2nd language?
284
No
Yes
Cause virtual computing session to be provided in 2nd language
285
Finish
286

FIG. 8

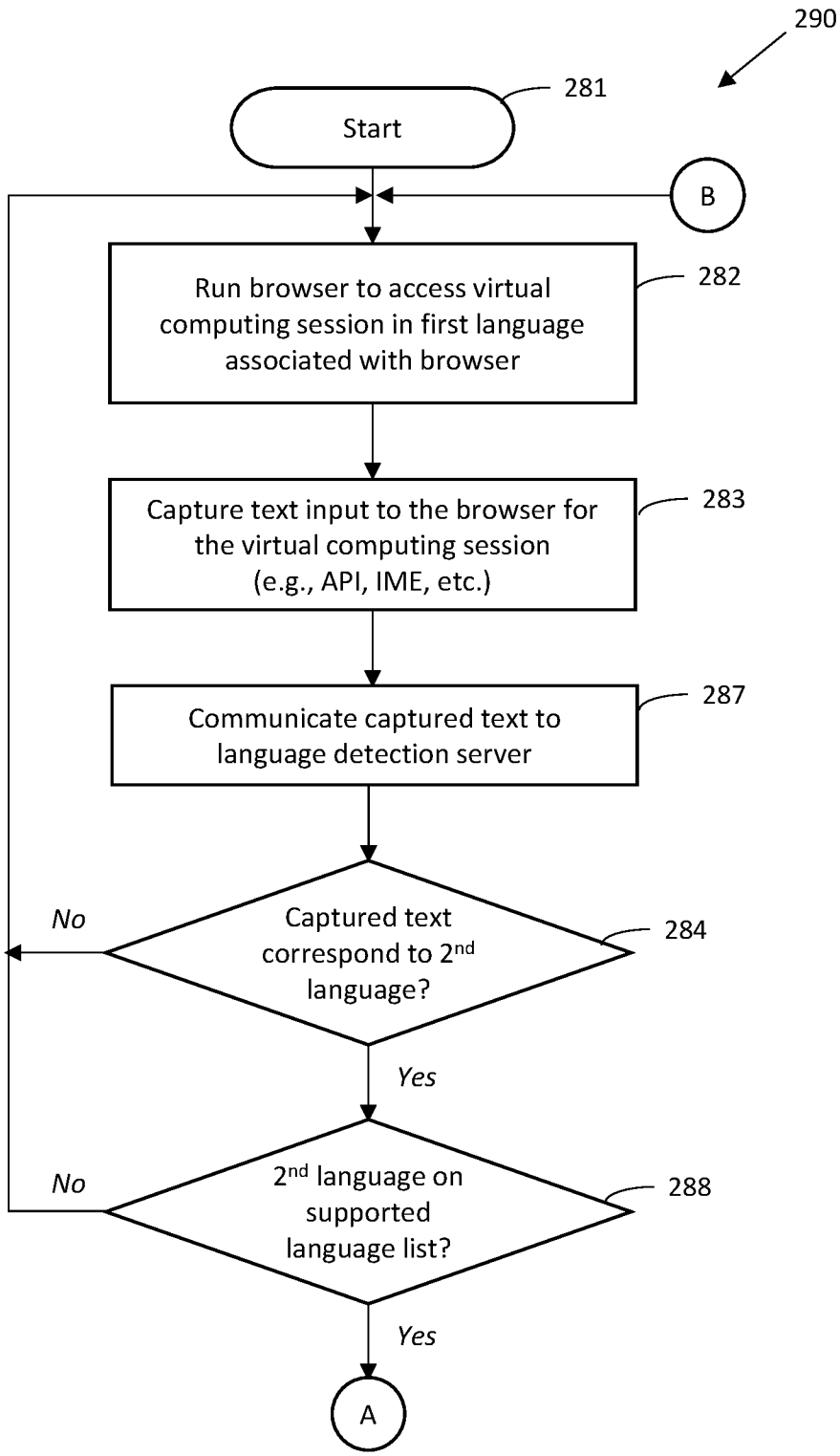
290
281
Start
B
Run browser to access virtual computing session in first language associated with browser
282
Capture text input to the browser for the virtual computing session (e.g., API, IME, etc.)
283
Communicate captured text to language detection server
287
No
Captured text correspond to 2nd language?
284
Yes
No
2nd language on supported language list?
288
Yes
A

FIG. 9A

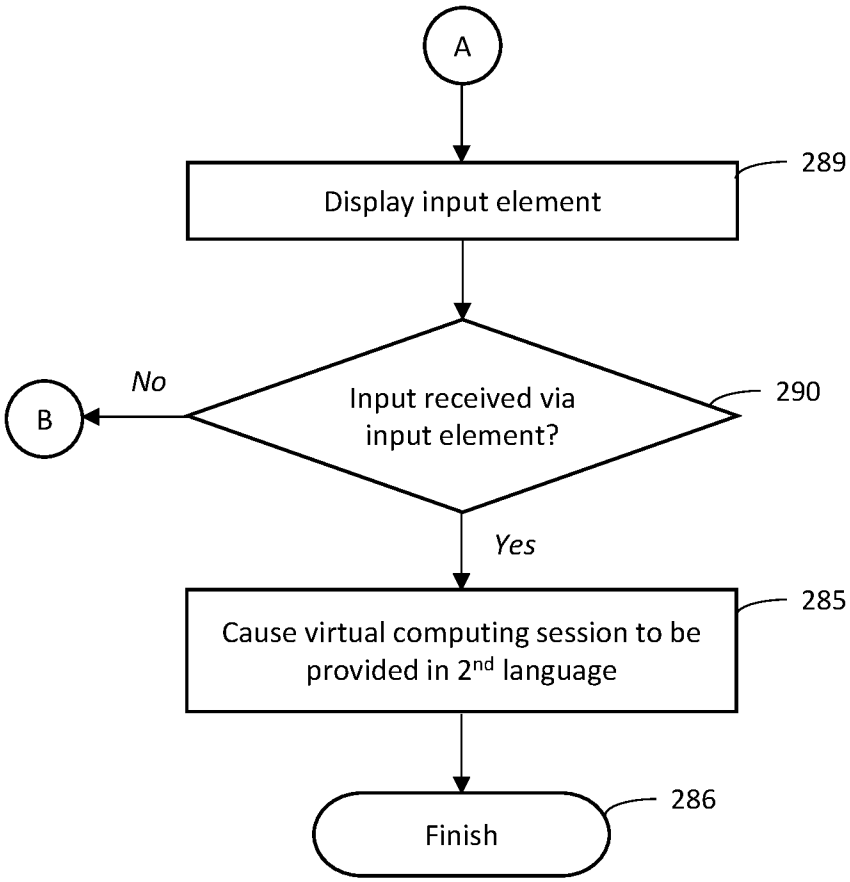
A
Display input element
289
Input received via input element?
290
No
B
Yes
Cause virtual computing session to be provided in 2nd language
285
Finish
286

FIG. 9B

INTERNATIONAL SEARCH REPORT

International application No
PCT/CN2022/085449

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/451
ADD.

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)
G06F

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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2015/142771 A1 (BHAGAT RAHUL H [US] ET AL) 21 May 2015 (2015-05-21) paragraphs [0022], [0034], [0040], [0041] figures 1, 2, 4, 5, 8	1-20
A	US 2015/161114 A1 (BURYAK KIRILL [US] ET AL) 11 June 2015 (2015-06-11) paragraphs [0005], [0028], [0029], [0033], [0041], [0075] figures 3, 4	1-20

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

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
1 September 2022	14/09/2022

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Hoisl, Bernhard

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/CN2022/085449

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2015142771 A1	21-05-2015	US 8949107 B1	03-02-2015
		US 2015142771 A1	21-05-2015
		US 2017200214 A1	13-07-2017
US 2015161114 A1	11-06-2015	NONE	