



US 20160011864A1

(19) **United States**

(12) **Patent Application Publication**
Fernandez-Ruiz et al.

(10) **Pub. No.: US 2016/0011864 A1**

(43) **Pub. Date: Jan. 14, 2016**

(54) **DEVELOPMENT OF PLATFORM
INDEPENDENT APPLICATIONS**

Publication Classification

(71) Applicant: **Yahoo! Inc.**, Sunnyvale, CA (US)

(72) Inventors: **Bruno Fernandez-Ruiz**, Mountain View, CA (US); **Daryl Mun-Kid Low**, San Jose, CA (US); **Martin F.N. Cooper**, Fremont, CA (US)

(51) **Int. Cl.**
G06F 9/445 (2006.01)
H04L 29/08 (2006.01)
H04L 29/06 (2006.01)
(52) **U.S. Cl.**
CPC **G06F 8/65** (2013.01); **H04L 67/42** (2013.01);
H04L 67/10 (2013.01)

(21) Appl. No.: **14/821,849**

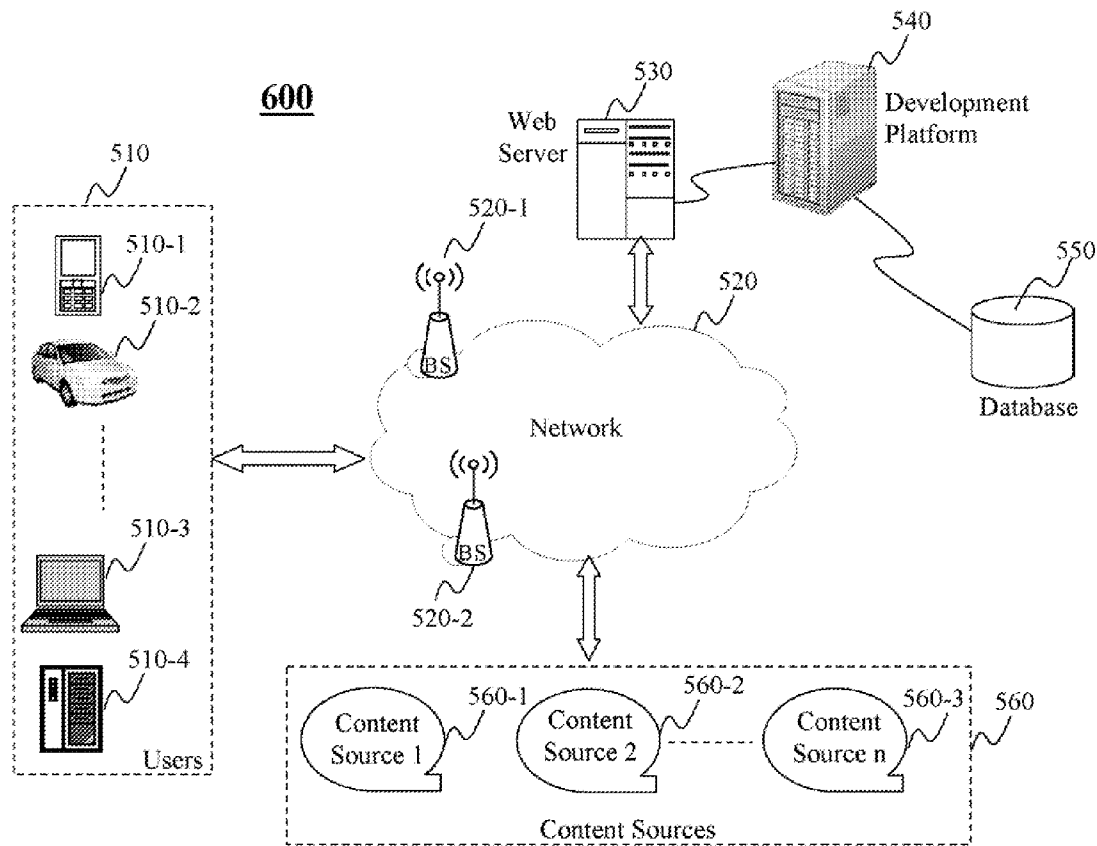
(22) Filed: **Aug. 10, 2015**

Related U.S. Application Data

(63) Continuation of application No. 13/550,914, filed on Jul. 17, 2012, now Pat. No. 9,158,520, which is a continuation-in-part of application No. 13/437,494, filed on Apr. 2, 2012.
(60) Provisional application No. 61/568,056, filed on Dec. 7, 2011.

(57) **ABSTRACT**

Method, system, and programs for providing packages to applications is disclosed. An application that is executable on a plurality of device platforms is generated. The application is deployed to a client device. A package list received from the client device is analyzed to determine package dependencies for packages on the package list. The packages on the package list and dependent packages are provided based on the package dependencies to the client device. The packages and dependent packages may be installed by the client device to ensure that the application is executable based on characteristics of the client device.



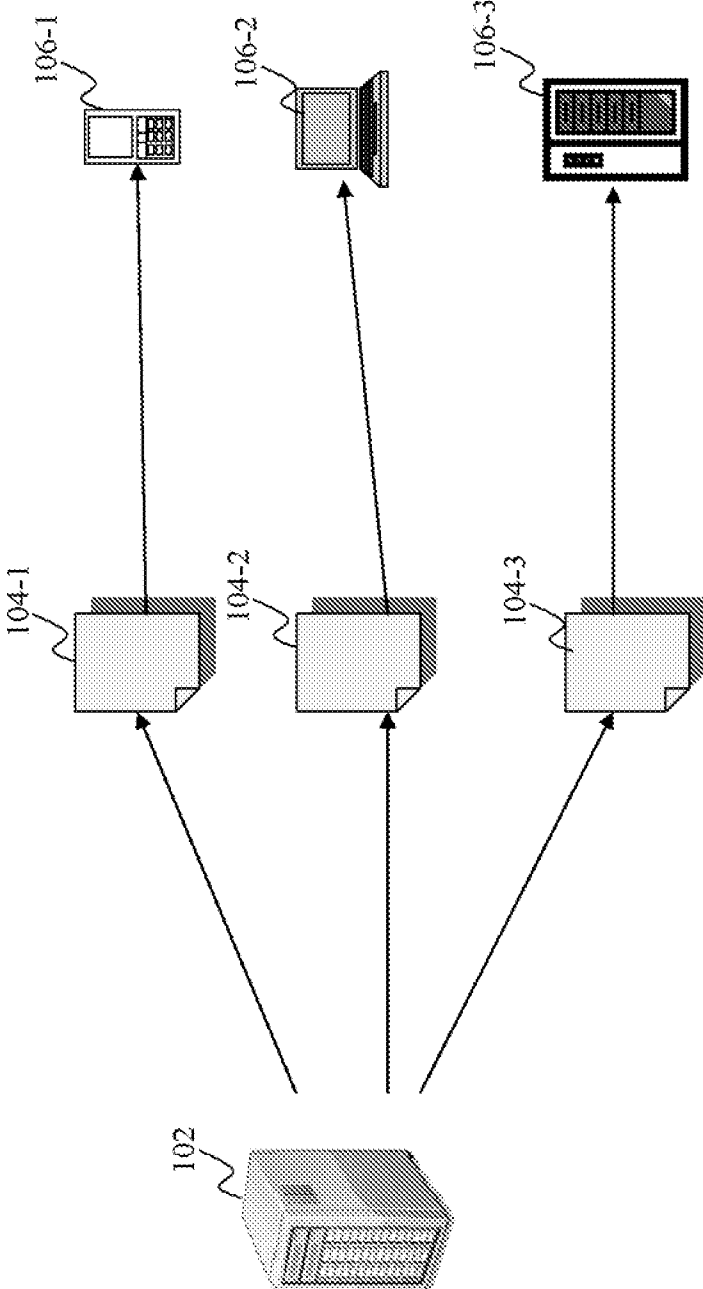


FIG. 1 (PRIOR ART)

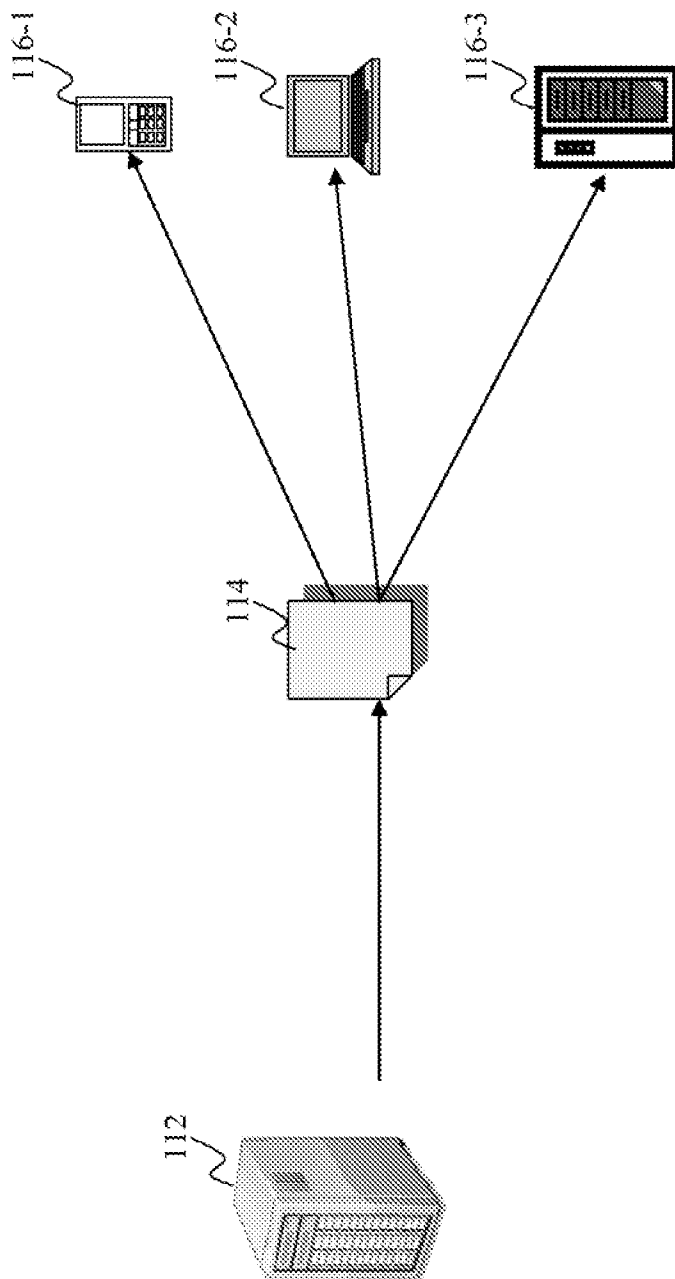


FIG. 2

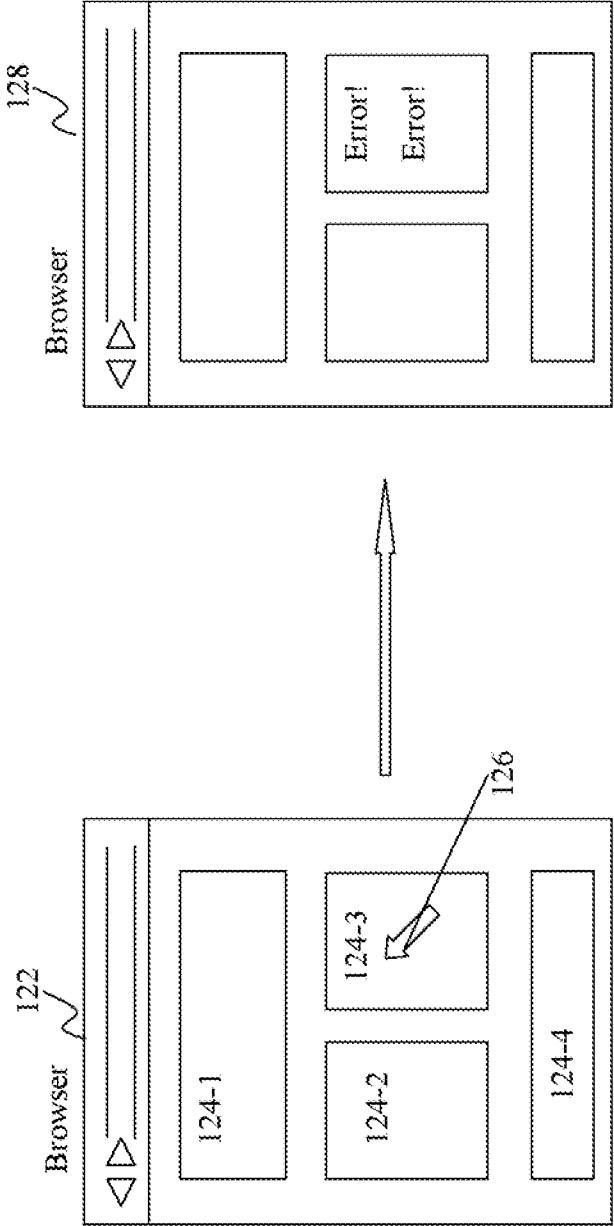


FIG. 3 (Prior Art)

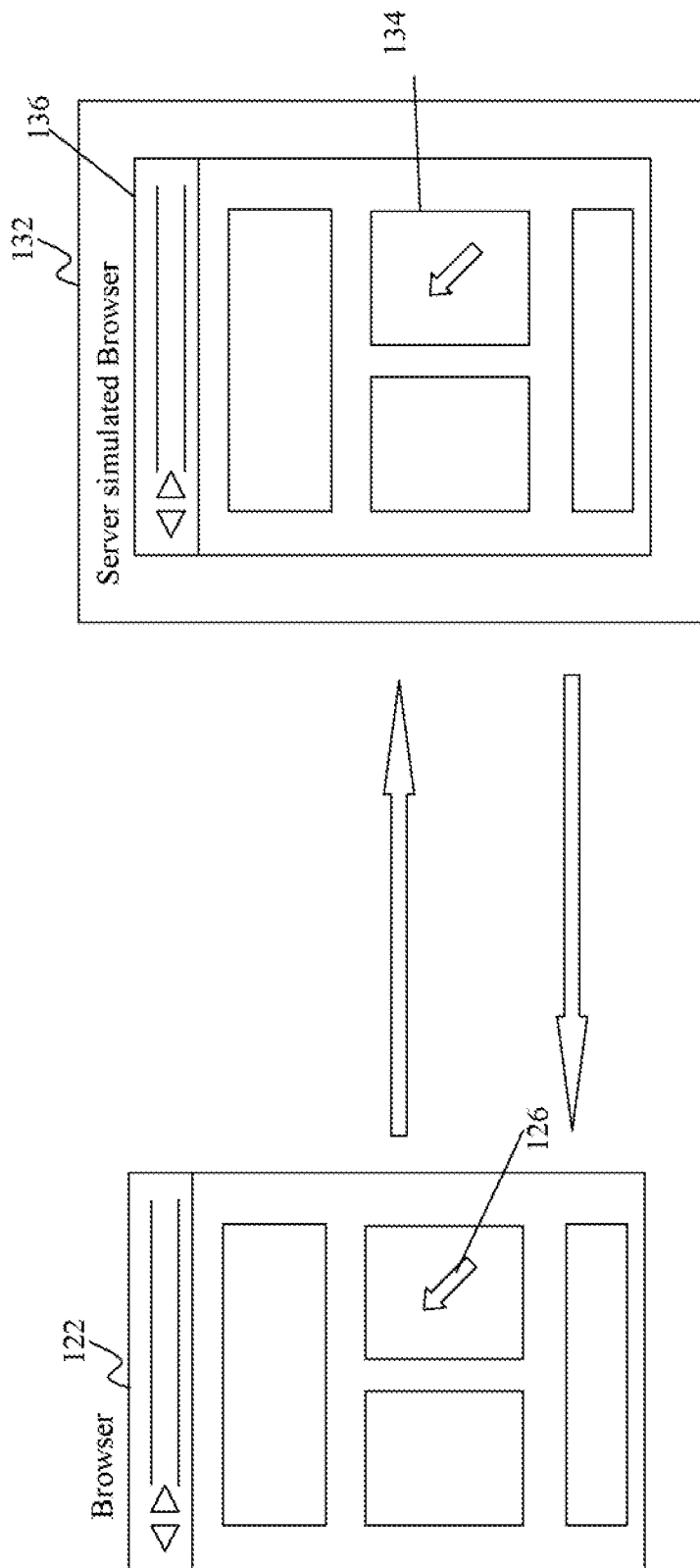


FIG. 4

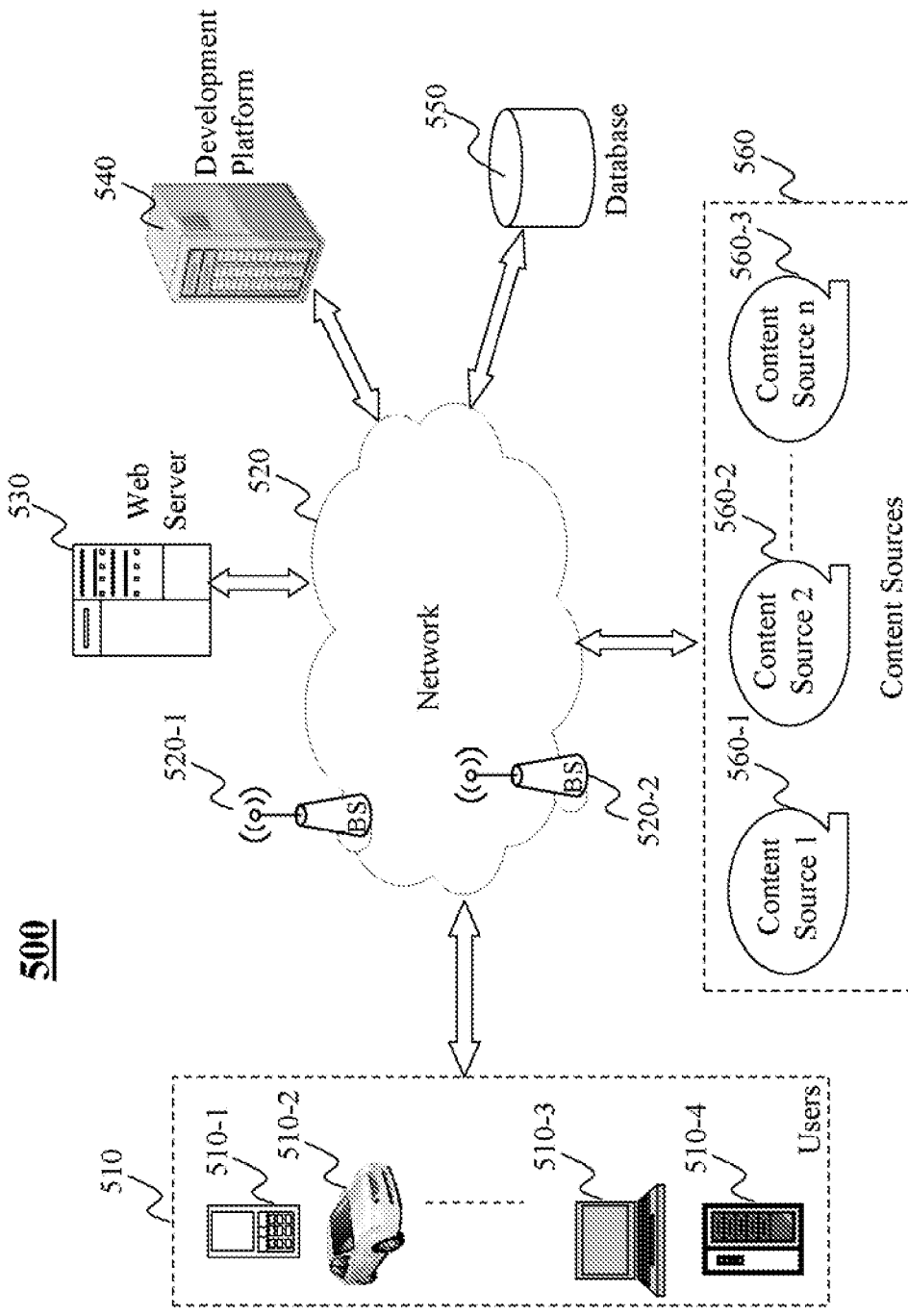


FIG. 5

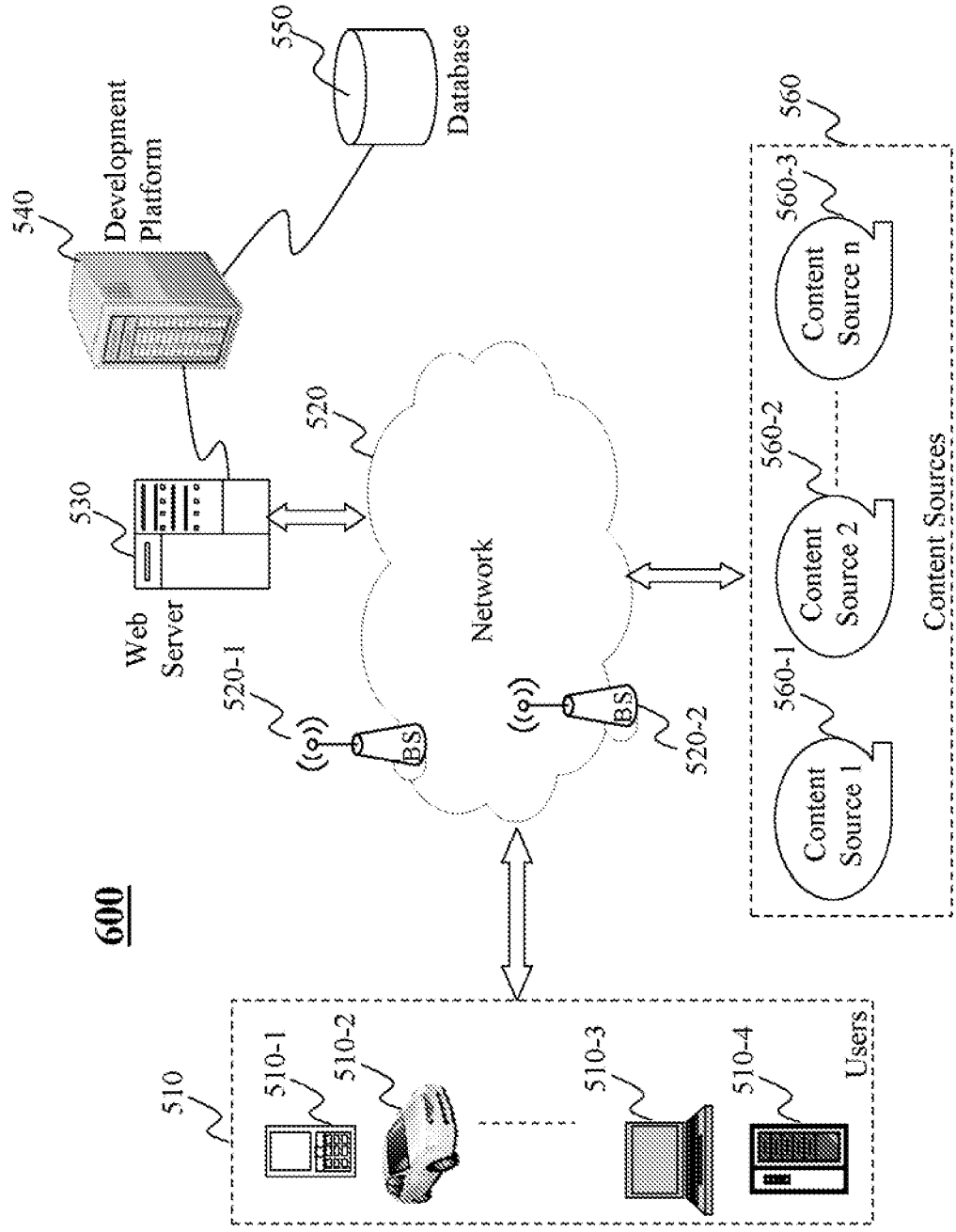


FIG. 6

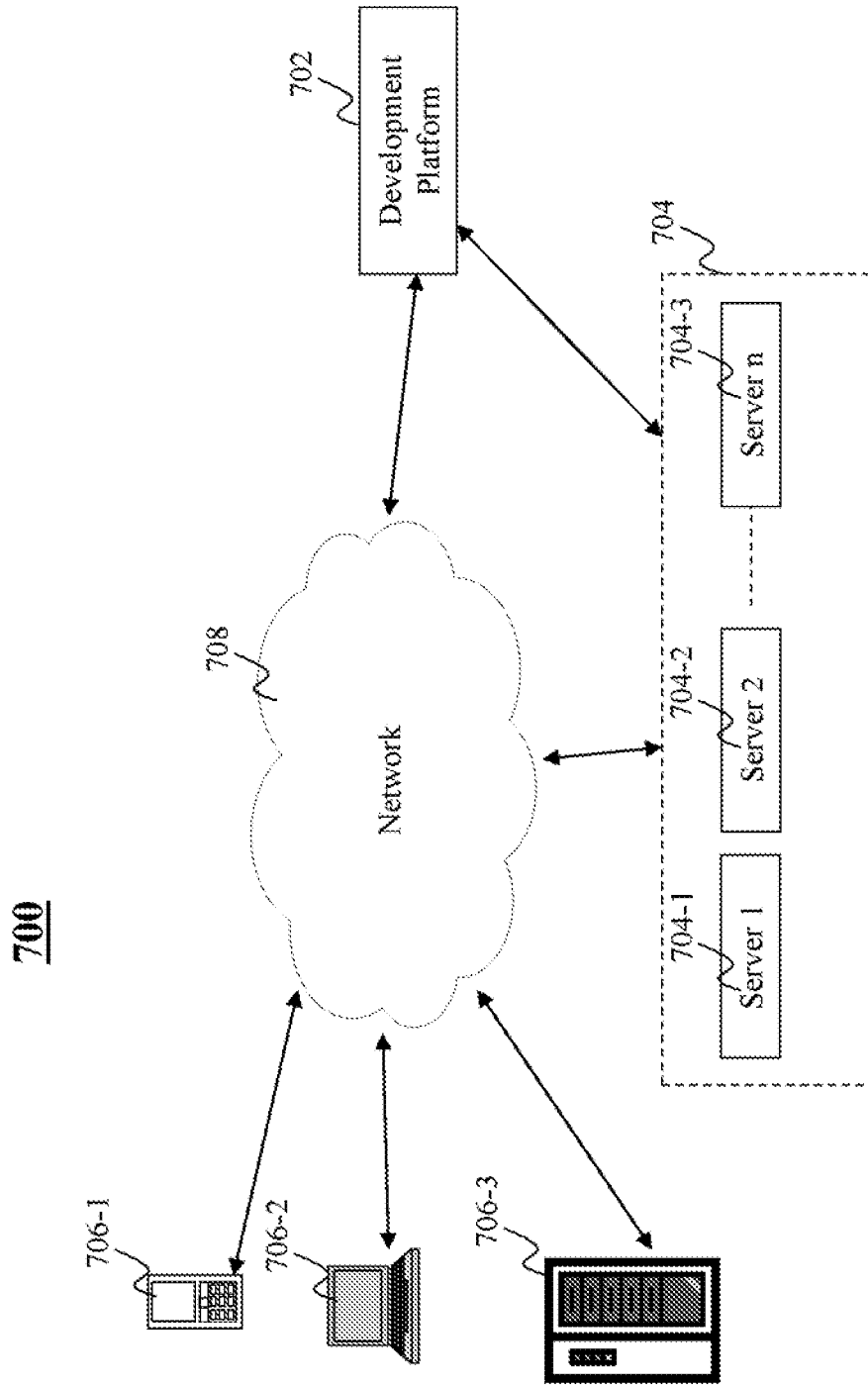


FIG. 7

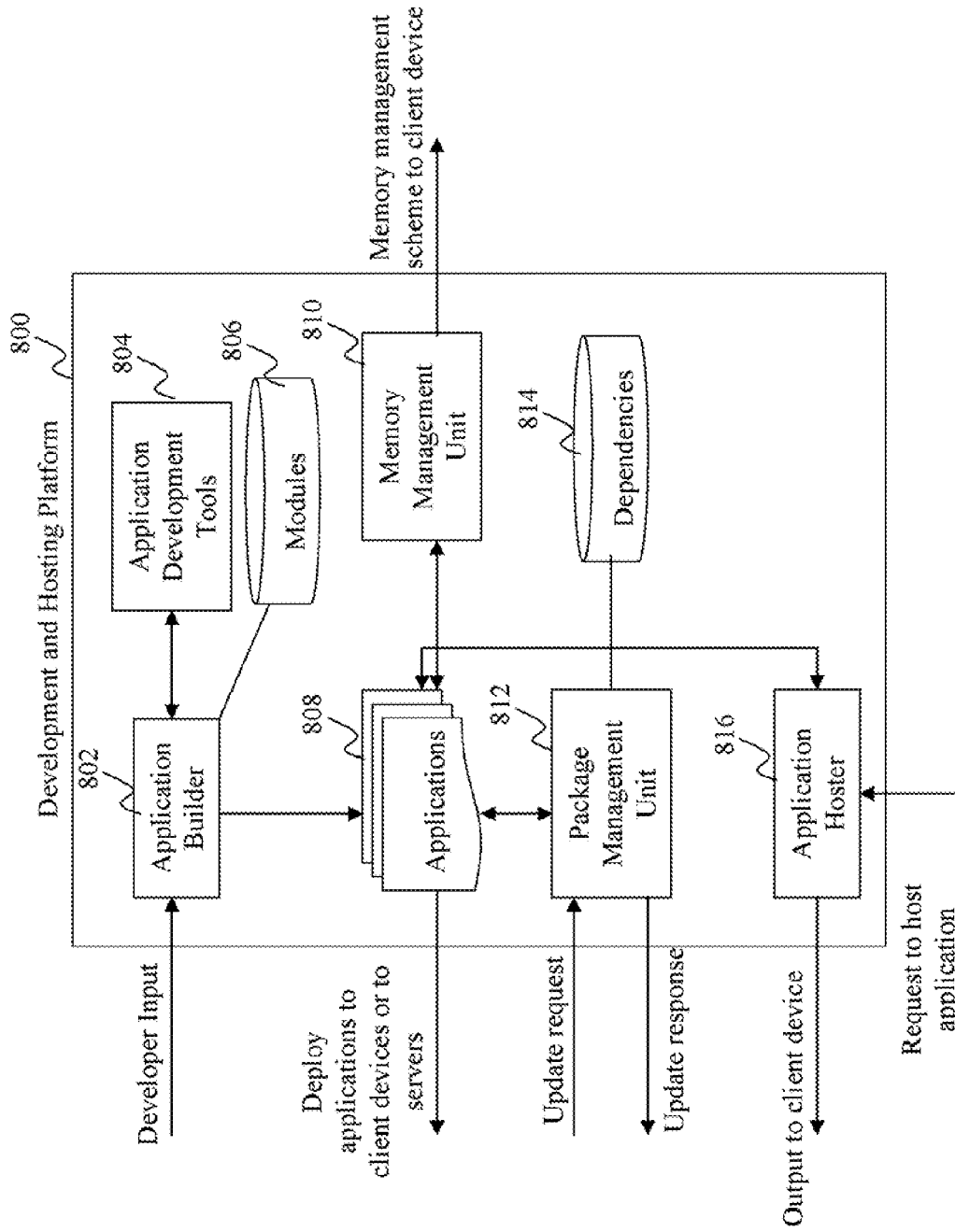


FIG. 8

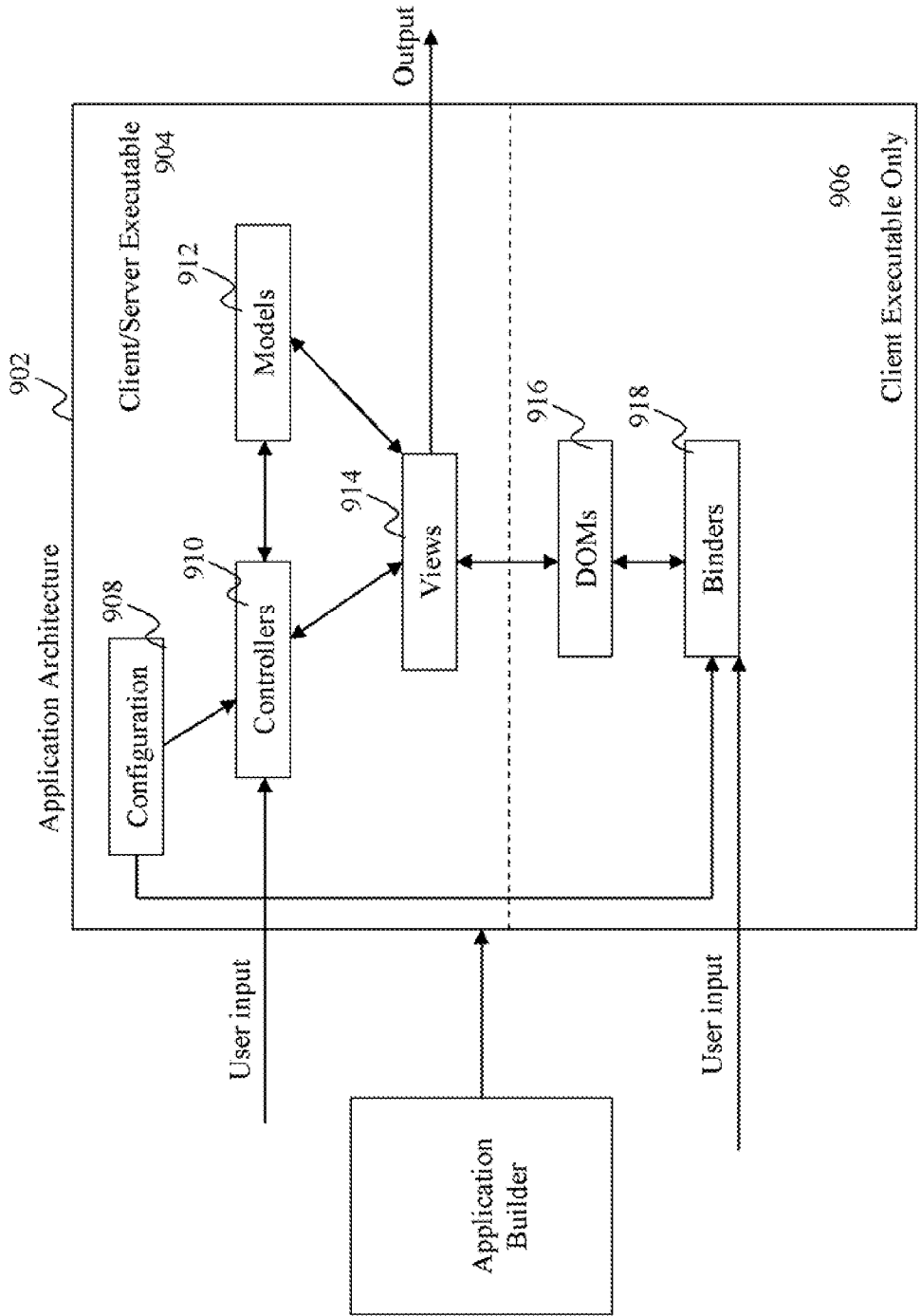
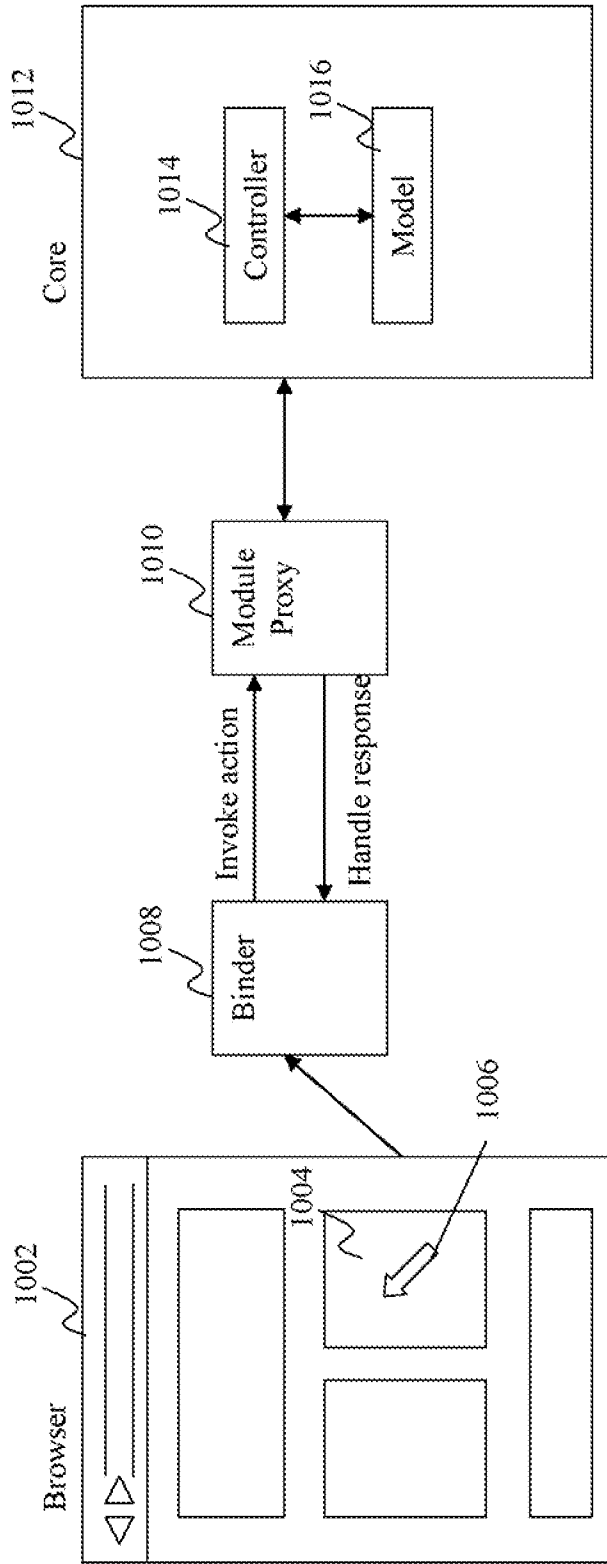
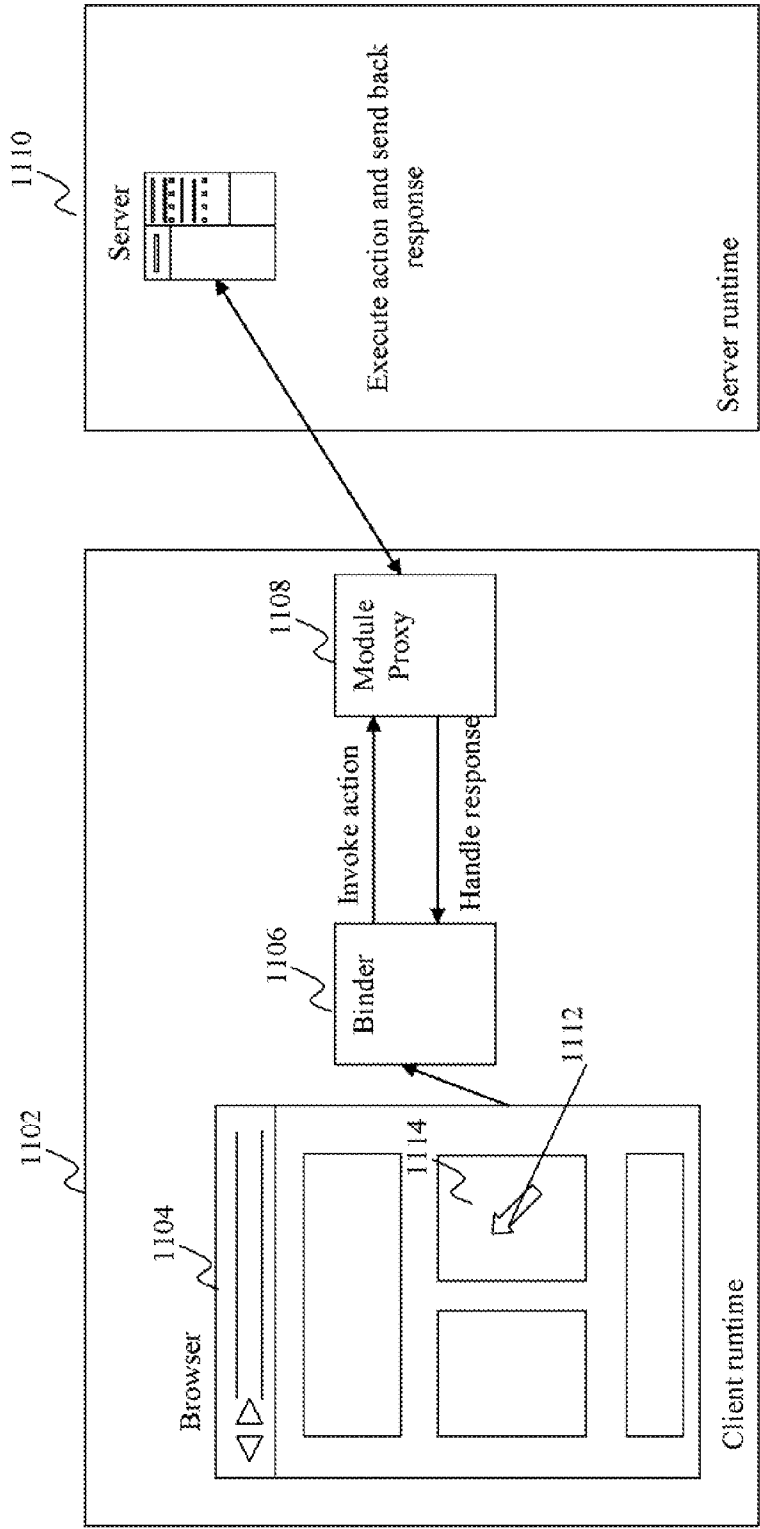


FIG. 9



Client-side Application Execution

FIG. 10



Server-side Application Execution

FIG. 11

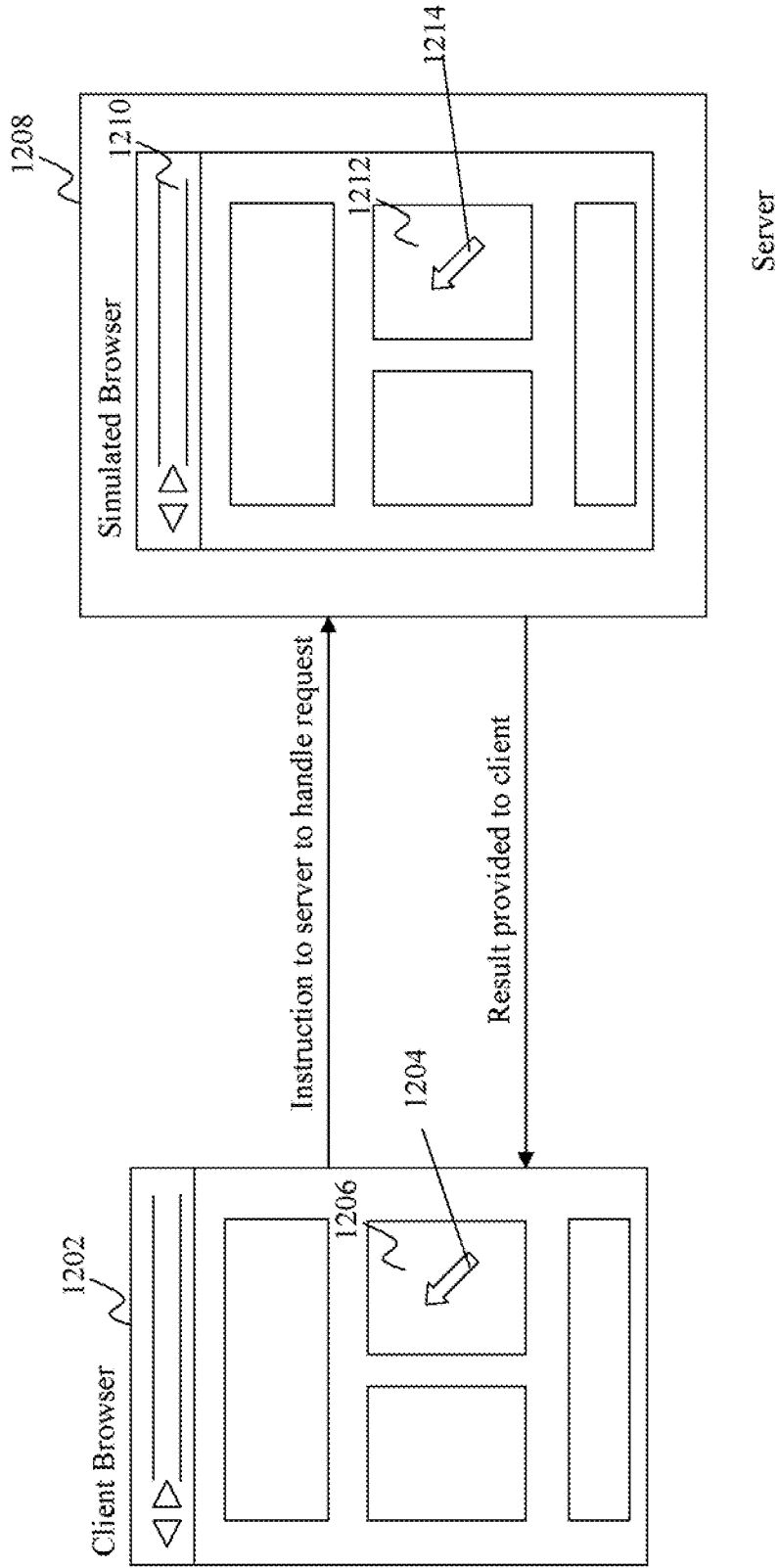


FIG. 12

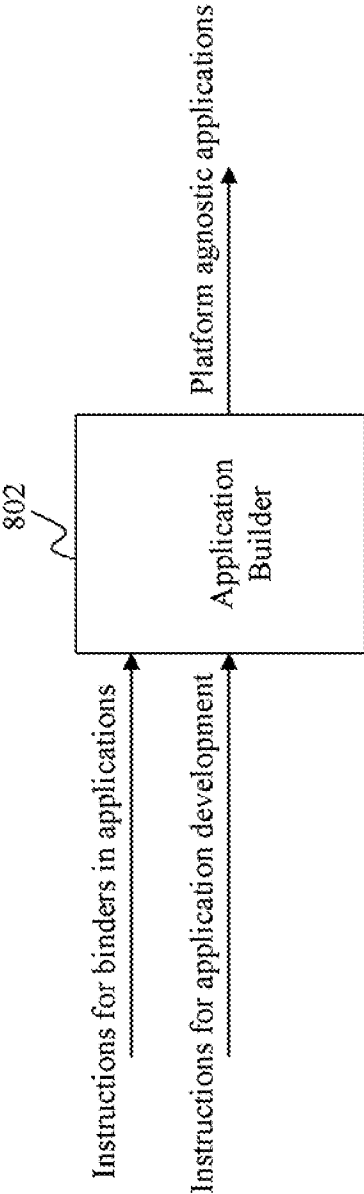


FIG. 13

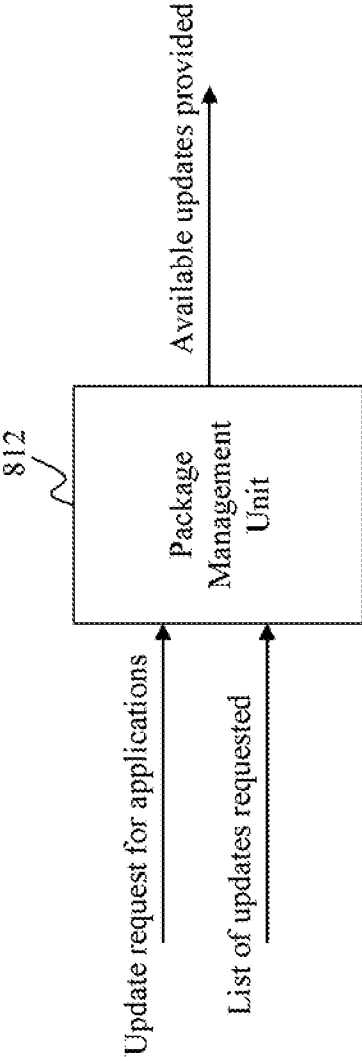


FIG. 14

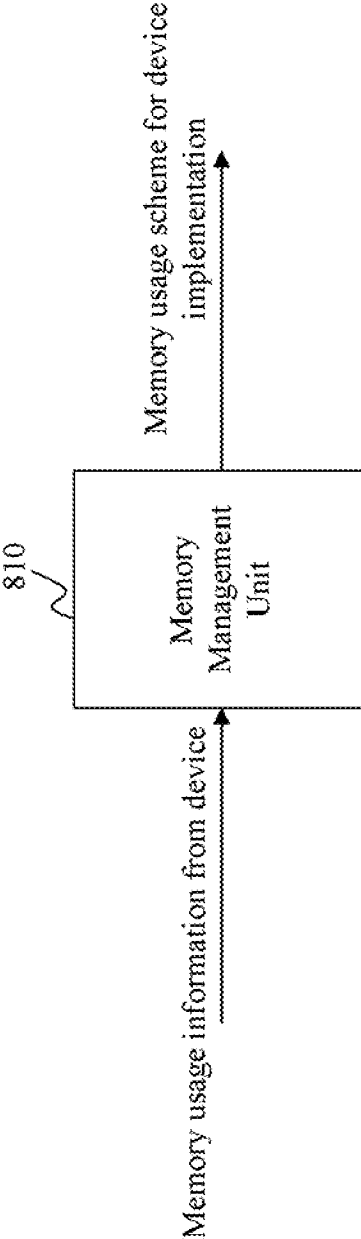


FIG. 15

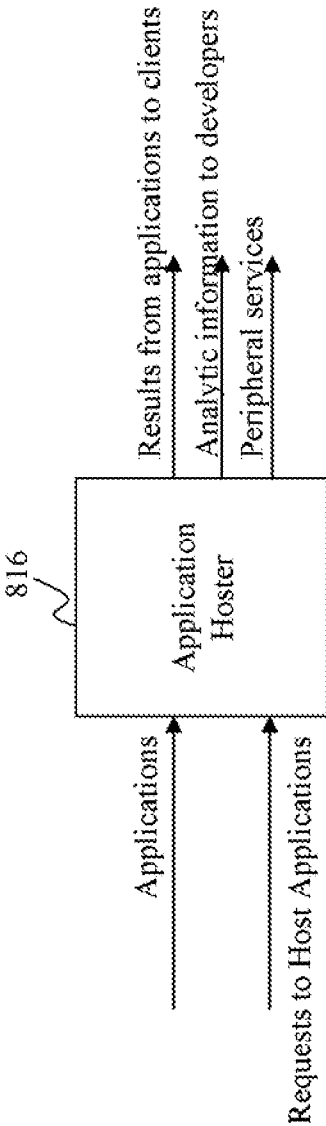


FIG. 16

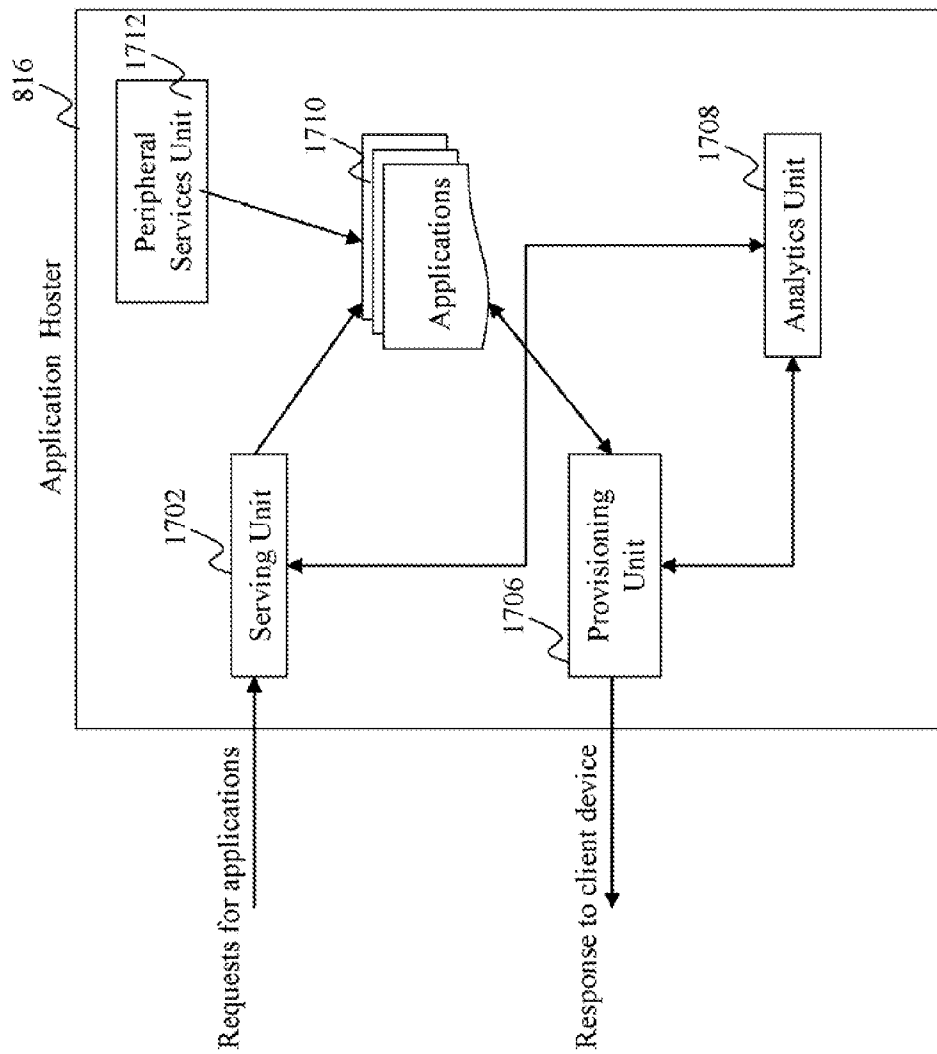


FIG. 17

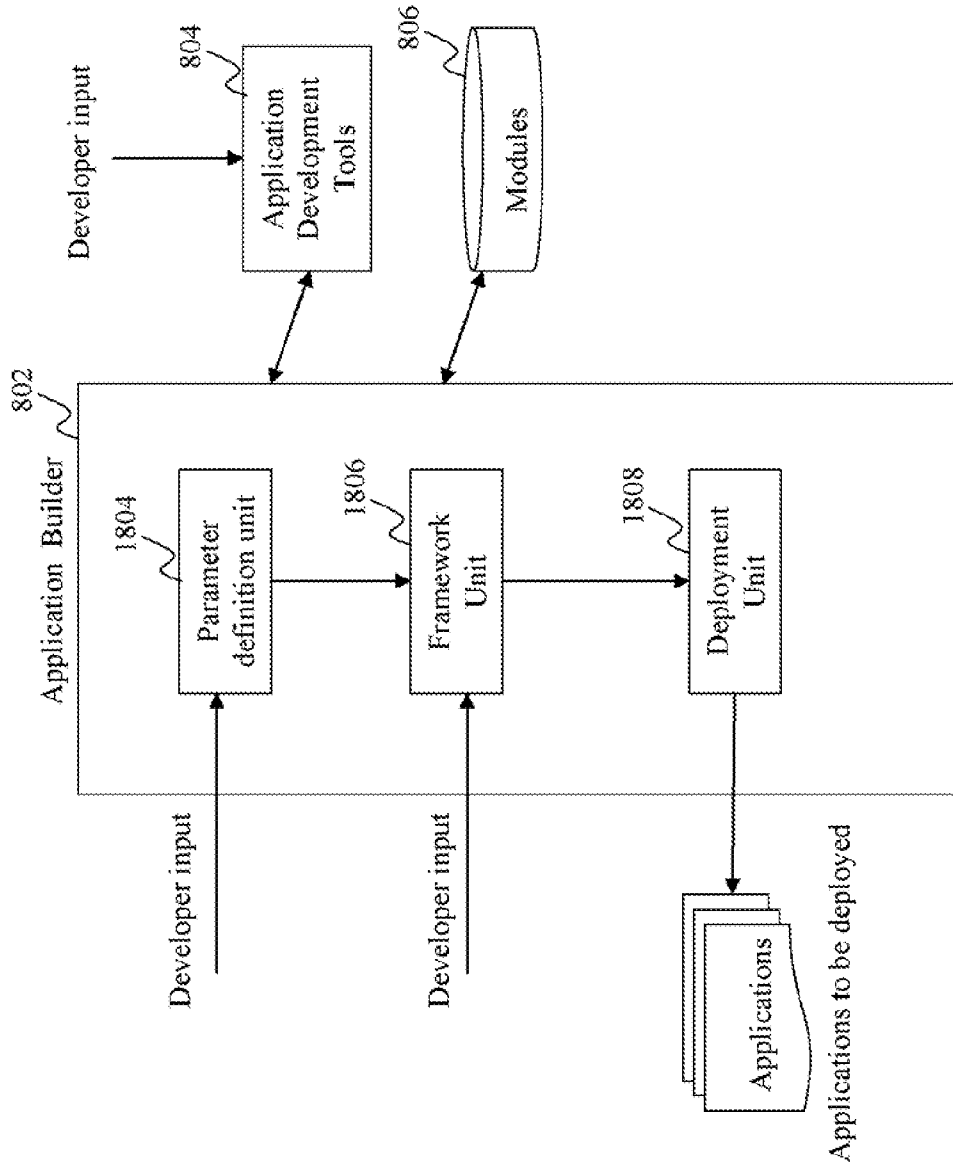


FIG. 18

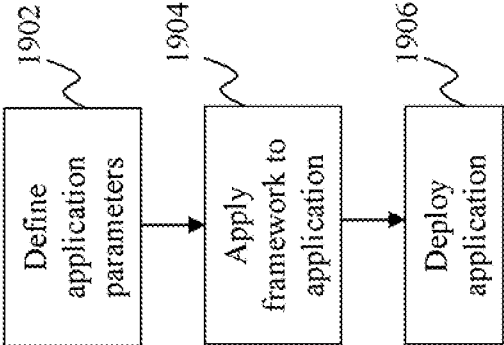


FIG. 19

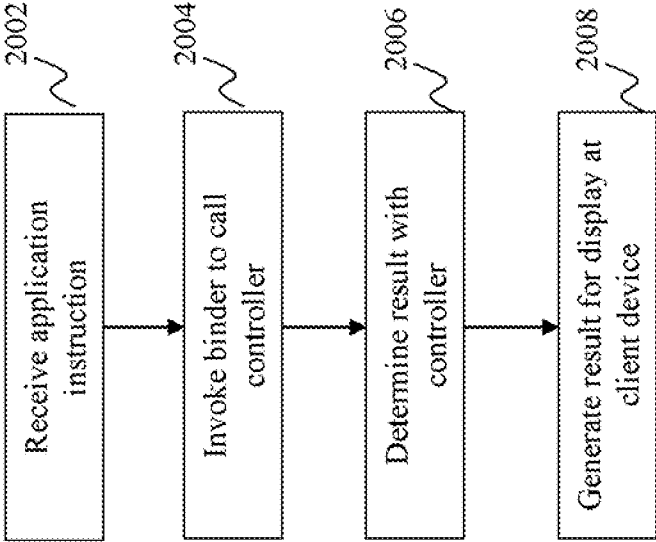


FIG. 20

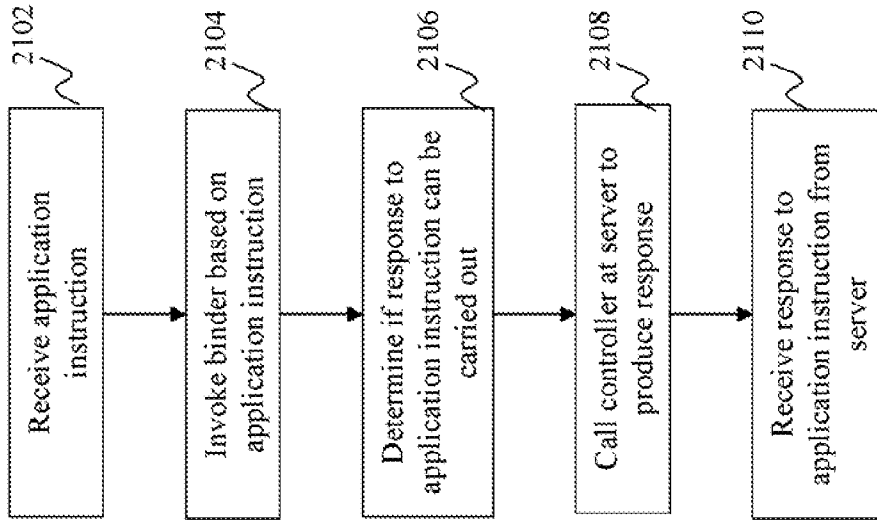
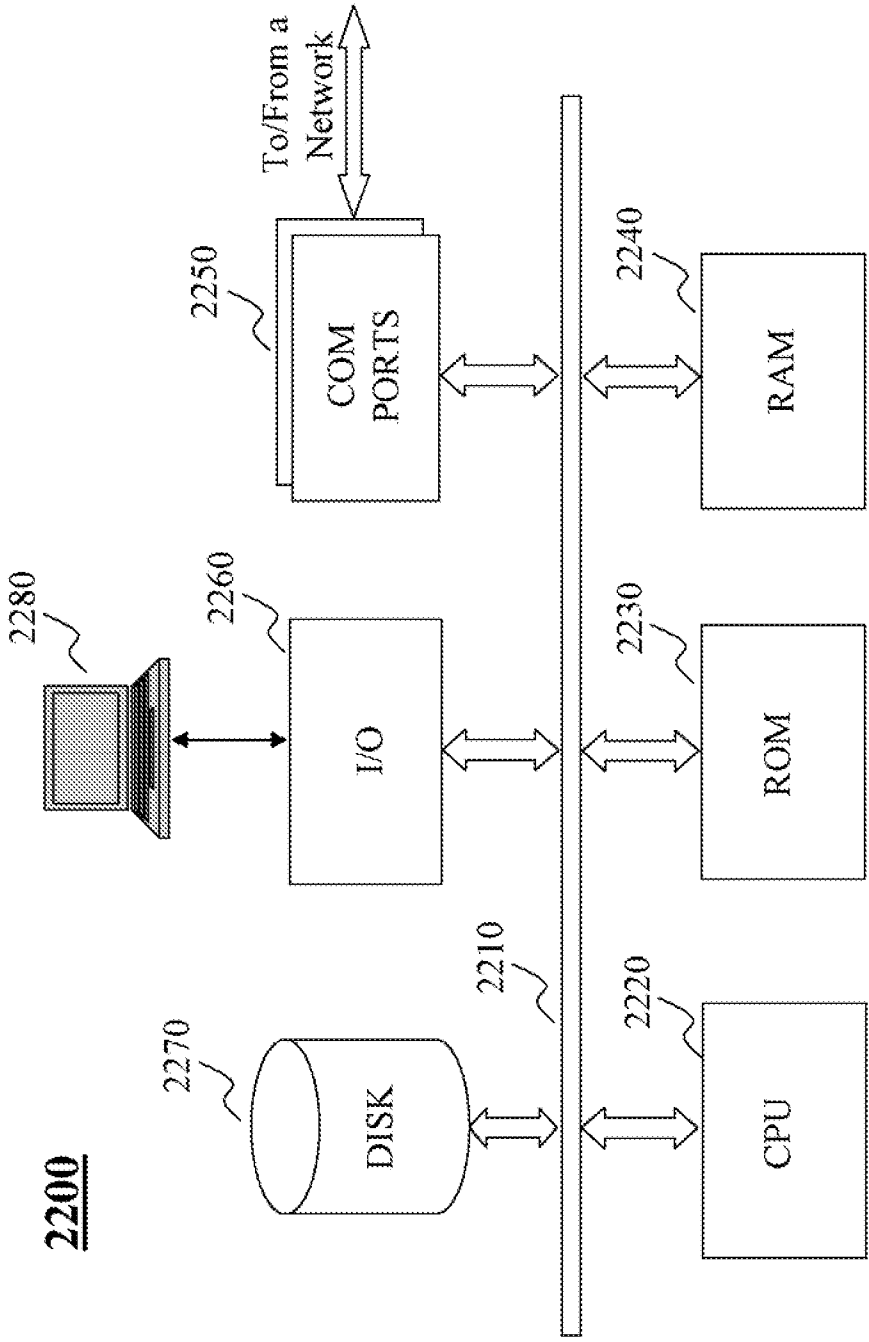


FIG. 21



2200

FIG. 22

DEVELOPMENT OF PLATFORM INDEPENDENT APPLICATIONS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is a continuation of U.S. application Ser. No. 13/550,914 filed Jul. 17, 2012, which is a continuation in part of U.S. application Ser. No. 13/437,494 filed Apr. 2 2012, which claims the benefit of priority to U.S. Provisional Application No. 61/568,056 filed Dec. 7, 2011, all of which is incorporated herein by reference in its entirety.

FIELD

[0002] The present disclosure relates to methods, systems and programming for developing platform independent applications. More particularly, the present disclosure is directed to methods, systems, and programming for developing applications that may be executed on any device platform.

BACKGROUND OF THE INVENTION

[0003] Recent advancements in technology have resulted in increased usage of different devices and operating systems. Devices such as laptops, smartphones, tablets, and computer systems may operate on a variety of operating systems, including Android, Windows, OS X, BlackBerry, and iOS. Users of these devices and operating systems have grown to expect seamless access and increased functionality and interactivity. As a result, users also expect interoperability between different devices. To adhere to the high standards set by users, developers and content providers must maintain and develop several versions and permutations of each application written for each different device and each different operating system. For example, the same smartphone application must often be written for at least each of the Android, iPhone, and Windows phone platforms, to say nothing of additional applications that may be needed for specific smartphones manufactured by Samsung, LG, HTC, T-Mobile, Blackberry, etc. Not only must these different versions of applications be developed, but they must also be tested, maintained, and stored. Furthermore, even if the applications are developed to be as similar as possible across different devices and platforms, users may still experience different results when accessing the same application from different smartphones, or from a personal computer, or laptop.

SUMMARY

[0004] The embodiments described herein relate to methods, systems, and programming for providing packages to applications.

[0005] In an embodiment, a method implemented on at least one computing device, each computing device having at least one processor, storage, and a communication platform connected to a network for providing packages to applications is disclosed. An application that is executable on a plurality of device platforms is generated. The application is deployed to a client device. A package list received from the client device is analyzed to determine package dependencies for packages on the package list. The packages on the package list and dependent packages are provided based on the package dependencies to the client device. The packages and dependent packages may be installed by the client device to ensure that the application is executable based on characteristics of the client device.

[0006] In an embodiment, a method implemented on at least one computing device, each computing device having at least one processor, storage, and a communication platform connected to a network for providing an application for execution on a plurality of device platforms is disclosed. An application that is executable on a plurality of device platforms is generated. Capabilities of the client device are detected. Packages to be applied to the application are determined to ensure that the application is executable based on the capabilities of the client device. The application and the packages are deployed to the client device.

[0007] In an embodiment, a method implemented on at least one computing device, each computing device having at least one processor, storage, and a communication platform connected to a network for implementing server side caching of packages is disclosed. A package list is received from a client device for an application that is executable on a plurality of device platforms. A determination is made as to which packages from the package list are required by the client device. Required packages are adaptively cached for delivery to the client device based on the determination.

[0008] In an embodiment, a system for providing packages to applications is disclosed. The system may comprise an application builder for generating an application that is executable on a plurality of device platforms and deploying the application to a client device; and a package management unit for analyzing a package list received from the client device to determine package dependencies for packages on the package list, and providing the packages on the package list and dependent packages based on the package dependencies to the client device, wherein the packages and dependent packages may be installed by the client device to ensure that the application is executable based on characteristics of the client device.

[0009] Other concepts relate to software for implementing provision of packages to applications is disclosed. A software product, in accord with this concept, includes at least one machine-readable non-transitory medium and information carried by the medium. The information carried by the medium may be executable program code data regarding parameters in association with a request or operational parameters.

[0010] In an embodiment, a machine readable and non-transitory medium having information recorded thereon for providing packages to applications, where when the information is read by the machine, causes the machine to perform the steps comprising: generating an application that is executable on a plurality of device platforms; deploying the application to a client device; analyzing a package list received from the client device to determine package dependencies for packages on the package list; and providing the packages on the package list and dependent packages based on the package dependencies to the client device, wherein the packages and dependent packages may be installed by the client device to ensure that the application is executable based on characteristics of the client device.

[0011] Additional advantages and novel features will be set forth in part in the description which follows, and in part will become apparent to those skilled in the art upon examination of the following and the accompanying drawings or may be learned by production or operation of the disclosed embodiments. The advantages of the present embodiments may be realized and attained by practice or use of various aspects of

the methodologies, instrumentalities and combinations set forth in the detailed description set forth below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The methods, systems, and/or programming described herein are further described in terms of exemplary embodiments. These exemplary embodiments are described in detail with reference to the drawings. These embodiments are non-limiting exemplary embodiments, in which like reference numerals represent similar structures throughout the several views of the drawings.

[0013] FIG. 1 depicts an exemplary prior art system for developing and deploying applications to different devices.

[0014] FIG. 2 depicts a system for developing and deploying a single application that is interoperable with a plurality of device platforms in accordance with an embodiment of the present disclosure.

[0015] FIG. 3 depicts an exemplary prior art browser interaction.

[0016] FIG. 4 depicts an exemplary results of a browser interaction with a server browser executing an application in accordance with an embodiment of the present disclosure.

[0017] FIG. 5 is a high level depiction of an exemplary system 500 in which a development platform is deployed to provide development and hosting of applications that are interoperable between different device platforms and operating systems.

[0018] FIG. 6 is a high level depiction of an exemplary system 600 in which a development platform is employed to provide and host applications that are interoperable between different device platforms and operating systems, in accordance with an embodiment of the present disclosure.

[0019] FIG. 7 is a high level depiction of an exemplary system 700 in which development platform 702 directs distributed hosting of applications, in accordance with an embodiment of the present disclosure.

[0020] FIG. 8 is a high level depiction of an exemplary development and hosting platform in accordance with an embodiment of the present disclosure.

[0021] FIG. 9 depicts an exemplary application MVC architecture in accordance with an embodiment of the present disclosure.

[0022] FIG. 10 depicts an exemplary client-side application execution in accordance with an embodiment of the present disclosure.

[0023] FIG. 11 depicts an exemplary server-side application execution in accordance with an embodiment of the present disclosure.

[0024] FIG. 12 depicts an exemplary view of a client browser running an application at a client, and a simulated browser at a server running the same application, in accordance with an embodiment of the present disclosure.

[0025] FIG. 13 depicts an exemplary high level system diagram of application builder 802, in accordance with an embodiment of the present disclosure.

[0026] FIG. 14 depicts an exemplary high level system diagram of package management unit 812, in accordance with an embodiment of the present disclosure.

[0027] FIG. 15 depicts an exemplary high level system diagram of memory management unit 810, in accordance with an embodiment of the present disclosure.

[0028] FIG. 16 depicts an exemplary high level system diagram of application hoster 816, in accordance with an embodiment of the present disclosure.

[0029] FIG. 17 depicts an exemplary high level system diagram of application hoster 816, in accordance with an embodiment of the present disclosure.

[0030] FIG. 18 depicts an exemplary high level system diagram of application builder 802, in accordance with an embodiment of the present disclosure.

[0031] FIG. 19 depicts a flowchart of an exemplary process in which application builder 802 is utilized to develop an application, in accordance with an embodiment of the present disclosure.

[0032] FIG. 20 depicts a flowchart of an exemplary process in which a user 510 of a device can execute an application that is interoperable across a plurality of device platforms, in accordance with an embodiment of the present disclosure.

[0033] FIG. 21 depicts a flowchart of an exemplary process in which a user 510 of a device can execute an application that is interoperable across a plurality of device platforms, in accordance with an embodiment of the present disclosure.

[0034] FIG. 22 depicts a general computer architecture in which the present embodiments can be implemented and has a functional block diagram illustration of a computer hardware platform which includes user interface elements.

DETAILED DESCRIPTION

[0035] In the following detailed description, numerous specific details are set forth by way of example in order to provide a thorough understanding of the relevant embodiments described herein. However, it should be apparent to those skilled in the art that the present embodiments may be practiced without such details. In other instances, well known methods, procedures, components and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the embodiments described herein.

[0036] The present disclosure relates to methods, systems and programming developing and hosting platform independent applications. More particularly, the present disclosure is directed to methods, systems, and programming for developing applications that may be executed on any device platform. The embodiments described herein provide a platform that facilitates development and hosting of applications using a cloud infrastructure. The development and hosting platform may deploy applications to client devices to be run locally at the client device or at a server. The deployed applications may also run off-line locally at the client device. The platform provides a web application framework that allows an application to execute both at a client or browser side or at a server side. As a result, application developers do not need to write different code for a server backend and a browser backend. Additionally, an application may run at the server side even if the client side encounters problems executing the application, because the applications all use a single code base. Furthermore, applications may run in a hybrid runtime utilizing resources of both a client device or client browser and a server.

[0037] FIG. 1 depicts an exemplary prior art system for developing and deploying applications to different devices. Server platform 102 is responsible for deploying different applications 104-1, 104-2, and 104-3 to different devices 106-1, 106-2, and 106-3. Each application 104-1, 104-2, and 104-3 represents a different version of the "same application that has been programmed to operate and execute specifically on a corresponding device 106-1, 106-2, and 106-3. As shown by FIG. 1, each device requires its own corresponding appli-

cation to be served to it. For example, application 104-1 corresponds to device 106-1, application 104-2 corresponds to device 106-2, and application 104-3 corresponds to device 106-3. Thus, development of applications can be inefficient, and due to time and cost constraints for developers, applications tailored for certain devices may simply not be developed due to these constraints.

[0038] FIG. 2 depicts a system for developing and deploying a single application that is interoperable with a plurality of device platforms in accordance with an embodiment of the present disclosure. Development and hosting platform 112, described in further detail in the paragraphs below, allows developers to develop a single application 114, that may be deployed to different devices 116-1, 116-2, and 116-3. Application 114 is written in a fashion that allows application 114 to be interoperable between different devices. Thus, developers only need to develop the application once and serve it across a network to any number of devices running any type of operating system.

[0039] FIG. 3 depicts an exemplary prior art browser interaction. In FIG. 3, browser 122 displays an application having four view screens 124-1-4. Each view screen represents a part of the application that is either viewable and/or actionable in response to user input. Cursor 126, shown in view screen 124-3 may be clicked or activated by a user. If activation of cursor 126 in view screen 124-3 requests a result from the application that browser 122 cannot execute, view screen 124-3 of browser 122 will show an error state, which is shown by browser 128 of FIG. 3. In certain circumstances, the entire application may crash or cease to operate, simply because a user may have attempted to execute an action in a view screen of the application that is not supported by either the operating system or the device in which the browser resides. For example, if clicking in view screen 124-3 represented an attempt to start a JavaScript application, and the browser, operating system, or device does not support JavaScript, then an error will result, such as that shown in browser 128. Thus, a user is unable to complete their task or use the application to its full capacity.

[0040] FIG. 4 depicts an exemplary result of a browser interaction with a simulated server browser executing an application in accordance with an embodiment of the present disclosure. Browser 122 is the same as the one shown in FIG. 3, except in this instance, browser 122 is executing an application that has been developed to be interoperable between different device platforms and operating systems, in accordance with the embodiments of the present disclosure. The main difference depicted by FIG. 4 is the result of a user interacting with cursor 126. When a user activates cursor 126, even if browser 122 is unable to produce a response due to incompatibility or a problem with its operating system or device, browser 122 is able to send the user instruction representing an activation of cursor 126 to server 132. Server 132 may host or run a hosted version of the same application, and carry out the user instruction, as shown in view screen 134 of simulated browser 136. The result of the user instruction may be transmitted back to browser 122 and shown within the corresponding view screen of browser 122. Thus, even if browser 122 itself, or its operating system or device, is incapable of producing a response to the user interaction with cursor 126, browser 122 may contact server 132 to produce the desired result. Server 132 may transmit a view screen showing the exact results to browser 122 so a user of browser 122 can continue using the application without realizing that

the application may have suffered from compatibility issues when executed at a client browser.

[0041] FIG. 5 is a high level depiction of an exemplary system 500 in which a development platform is deployed to provide development and hosting of applications that are interoperable between different device platforms and operating systems, in accordance with an embodiment of the present disclosure. Exemplary system 500 includes users 510, network 520, web server 530, content sources 560, development platform 540, and a database 550. Network 520 can be a single network or a combination of different networks. For example, a network may be a local area network (LAN), a wide area network (WAN), a public network, a private network, a proprietary network, a Public Telephone Switched Network (PTSN), the Internet, a wireless network, a virtual network, or any combination thereof. A network may also include various network access points, e.g., wired or wireless access points such as base stations or Internet exchange points 520-1, . . . , 520-2, through which a data source may connect to in order to transmit information via the network.

[0042] Users 510 may be of different types such as users connected to the network via desktop connections (510-4), users connecting to the network via wireless connections such as through a laptop (510-3), a handheld device (510-1), or a built-in device in a motor vehicle (510-2). A user may run applications provided by web server 530, development platform 540, or any of content sources 560. Thus, applications may be provided from web server 530, development platform 540, or any of content sources 560 through network 520. Once a user is running an application on their device, user may send instructions or requests via the application to web server 530, development platform 540, or any of content sources 560 through network 520. The application may also independently communicate with web server 530, development platform 540, or any of content sources 560 through network 520 as needed to ensure that the application can execute properly.

[0043] For example, an application that is interoperable across different platforms and operating systems may be provided to any of users 510 by web server 530, development platform 540, or any of content sources 560 through network 520. The application may be developed at development platform 540 but may also be deployed through any of the aforementioned components. The application may also be hosted by development platform 540 so that the entire application is executed remotely, with all results of user interactions with the application provided to the users 510.

[0044] The content sources 560 include multiple content sources 560-1, 560-2, . . . , 560-3. A content source may correspond to a web page host corresponding to an entity, whether an individual, a business, or an organization such as the USPTO represented by USPTO.gov, a content provider such as Yahoo.com, or a content feed source. It is understood that any of these content sources may store and host applications that have been developed using development platform 540 in accordance with an embodiment of the present disclosure. A content source may also include, for example, an application store that provides applications to personal computers, laptops, or mobile devices. Web server 530 and development platform 540 may access information from any of content sources 560 and rely on such information to respond to application requests and vice versa. Development platform 540 may also access additional information, via network 520,

stored in database 550, which may contain modules and/or application tools for developing applications.

[0045] In exemplary system 500, an application is developed at development platform 540. The application is developed such that the application is interoperable across different device platforms and operating systems. Additionally, the application is developed such that if the application is one requiring a hybrid client device and server runtime, the application may execute on its own at a client or user device independent of the device's connection to a server over a network. The application is also developed so that it may adaptively utilize the resources of a server if a server is better suited to execute certain aspects of the application or the entire application. Finally, the application is developed so that it may be hosted completely at a server, which simply provides response to application user input to a user or client device, while the server carries out all instructions and requests received by the application via user input.

[0046] The application need only be developed once and the same application may be deployed across network 520 to any of users 510. The application may be deployed to users 510 from any of web server 530, development platform 540, or content sources 560.

[0047] In an embodiment, the application is deployed to user 510-1 using a mobile device. If, for example, user 510-1 of the mobile device enters an area of limited or no mobile broadband connectivity, the application is able to continue executing even though the application cannot establish a connection with a web server or content source from which it is deployed.

[0048] In another embodiment, the application is deployed to user 510-3 using a laptop that is incompatible with the application because the laptop is not capable of supporting JavaScript. However, development platform 540, from which the application is deployed, may host the application and carry out any requests of user 510-3 at their laptop. Development platform 540 provides user 510-3 responses to user inputs of the application that is viewable and actionable by user 510-3 so that user 510-3 does not notice any difference or suffer any consequence from their laptop being incompatible with the application.

[0049] FIG. 6 is a high level depiction of an exemplary system 600 in which a development platform is deployed to provide development and hosting of applications that are interoperable between different device platforms and operating systems, in accordance with an embodiment of the present disclosure. In this embodiment, development platform 540 serves as a backend system of web server 530. All communication to and from development platform 540 are sent and received through web server 530.

[0050] FIG. 7 is a high level depiction of an exemplary system 700 in which development platform 702 directs distributed hosting of applications, in accordance with an embodiment of the present disclosure. Development platform 702 is a platform that may be used to develop applications that are interoperable across different device platforms. Applications developed by development platform 702 may be deployed to any of servers 704, which then host the applications that are eventually distributed for use by devices 706 over network 708. For example, a developer develops an application at development platform 702. The completed application is deployed to server 704-1. Server 704-1 is then responsible for deploying the application to any of devices 706 over network 708 for usage.

[0051] FIG. 8 depicts a high level depiction of an exemplary development and hosting platform in accordance with an embodiment of the present disclosure. Development and hosting platform 800 includes an application builder 802, application development tools 804, modules database 806, applications 808, memory management unit 810, package management unit 812, dependencies database 814, and application hoster 816. Application builder 802, receives developer input, and generates applications 808. Each application that is generated includes at least one module that may be retrieved from, for example, modules database 806. Application development tools 804 are used to assist in developing the applications.

[0052] All applications that are developed are based on a browser based or scripting based application framework. In an embodiment, this browser based or scripting based application framework is a JavaScript application framework. Thus, in an embodiment, the applications are written or coded completely using JavaScript. The JavaScript application framework facilitates development of applications that can be executed in both client-side and server-side environments. The JavaScript application framework ensures that developers using application builder 802 do not have to write a different set of code for a server backend and a browser frontend. The JavaScript application framework also ensures that even if a browser attempting to execute the application is not capable of running JavaScript applications or is not compatible with JavaScript, the application will adapt and execute instead at a server side. Since all applications may be JavaScript based, a single code base may be used. A module/widget application framework is also used along with the JavaScript application platform by application builder 802 to develop applications that may be deployed to devices. The application framework facilitates the development of transportable code between client devices or browsers at client devices and server. The application framework provides a software architecture pattern that isolates application logic for a user from a user interface, thus permitting independent development, testing, and maintenance of each.

[0053] Thus, as discussed above, each application generated by application builder 802 allows developers to develop one application that is interoperable between different device platforms and operating systems. The same single application may be deployed to a variety of devices and be executable on all of them. The applications may be deployed to any device that is capable of running or having a browser environment capable of executing the applications.

[0054] Application development tools 804 facilitates continuous integration of tasks for developers using application builder 802 to develop applications. The development of applications may be monitored, and output from the applications reviewed. Application development tools 804 may also be used in conjunction with application builder 802 to add device specific presentation capabilities to applications.

[0055] Memory management unit 810 may be used to add yet another level of functionality to the applications developed by application builder 802. Before, or after deployment of an application to a device, memory management unit 810 may detect a current or expected memory usage of the device. If the application will require more memory than is available at the device, memory management unit 810 may deploy a memory management scheme to the device. The memory management scheme ensures that the device will always have

enough memory to support execution of the application. This prevents the application from encountering memory related errors.

[0056] Package management unit **812** is responsible for ensuring that deployed applications are kept updated. Deployed applications may periodically or at user request, submit an update request to package management unit **812** to serve updated packages to the applications. The update request includes a list of currently installed and additional packages it wants to acquire. Package management unit **812** checks the package list and obtains package dependencies from dependencies database **814**. Package dependencies ensure that the client device receives not only the packages it requests, but also additional, and dependent packages that should be served to ensure that the application is fully upgraded and able to operate. Package management unit **812** is then able to deploy or serve all updated and required packages to the client device autonomously.

[0057] Package management unit **812** may also be responsible for ensuring that the correct packages are delivered to a client device based on the client device type. For example, if a client device is a smartphone operating on iOS, package management unit **812**, ensures that the packages received by the client device are compliant with iOS and are optimized such that the application will execute taking full advantage of the capabilities of the client device.

[0058] In another embodiment, package management unit **812** may facilitate retrieval of partial previews of compatible remote packages not currently installed on the client device. Partial previews provide a preview mode for “try before you buy” scenarios which may apply to up-to-date and older version of applications. Compatibly previews may be adaptively served by package management unit **812** based on a current version of an application executing on a client device.

[0059] For example, if a client device is running an older version of an application, the server may need to determine a more appropriate package corresponding to an older version of the application. The client device may submit a request to package management unit **812** for packages it wishes to preview. Package management unit **812** determines the client device is executing an older version of an application and searches for files which have been tagged to represent their association with a particular version of the application. Once package management unit **812** finds the appropriate files, it responds to the client device with the location of the files so that the client device may retrieve them.

[0060] In another embodiment, package management unit **812** facilitates server side caching. Server side caching involves package management unit **812** making an independent determination of which packages are required at a client device. This allows package management unit **812** to adaptively cache certain packages to ensure that required packages are deployed in an efficient manner to prevent any slow down of application execution. Intermediate computations may be cached to aid future package requests.

[0061] In another embodiment, package management unit **812** performs a determination of a minimal set of packages required in order to update an application at the client device. This minimal set of packages represents only the packages that are not present at the application but are required. This ensures that when a client device sends a list of packages to the package management unit **812**, package management unit **812** will perform an independent determination to determine the minimal set of packages, representing a barebones set of

packages necessary to update the application at the client device. Thus, this facilitates an efficient usage of resources and ensures that the application will not be bogged down by receiving too many unnecessary updates at once.

[0062] In another embodiment, package management unit **812** may send instructions to a client device to delete, remove, or clean up packages residing on the client device related to the application that are no longer needed. Package management unit **812** may also prepare updates to the client device based on asset of transactions defined by package management unit **812**, with instructions allowing the client device to implement a rollback feature in cases where updated packages fail or do not operate as expected on the client device.

[0063] Application hoster **816** is a hosting environment for all applications developed by application builder **802**. More specifically, application hoster **816** is a server-side JavaScript hosting environment for applications, allowing application developers to deploy, un-deploy, and manage multiple versions of applications. Applications deployed from application hoster **816** are able to be executed on any device platform due to the platform agnostic nature of the applications. Application hoster **816** provides hosting of applications in a network environment. Thus, the servers shown in FIG. 7 are not necessary for hosting developer created applications.

[0064] FIG. 9 depicts an exemplary application architecture in accordance with an embodiment of the present disclosure. Application architecture **902** represents the application framework used by application builder **802** to develop applications. Since applications developed using application builder **802** may be written completely in a browser oriented language or a scripting language. For example, the language may be JavaScript. It follows that both client components and server components of any application developed using application builder **802** are written in JavaScript as well. This facilitates the execution of applications on both a client device or browser at a client device, or at a server. FIG. 9 depicts elements of an application architecture **902** and corresponding components.

[0065] Application architecture **902** includes client/server executable components **904** and client executable components **906**. Client/server executable components **904** include configuration files **908**, controllers **910**, models **912**, and views **914**. Client executable components **906** includes document object models (DOMs) **916** and binders **918**.

[0066] Configuration files **908** define relationships between code components, assets, routing paths, and defaults. When an application is first created by a developer using application builder **802**, configuration files are generated. Controllers **910** react to user input and initiate responses to user input by calling models **912**. For example, a particular controller associated with a particular model may receive an indication of a user input and instruct a corresponding model and corresponding view from views **914** to perform actions based on the user input. Models **912** manage the behavior and data of an application domain, and responds to requests for information about its state (from the views), and responds to instructions to change state (from the controllers). More specifically, models **912** notify views **914** about changes in information so that views **914** can react accordingly. Views **914** renders models **912** into forms suitable for interaction, typically in the form of a user interface element. Multiple views may exist for a single model. Views **914** represent what a user of an application sees on a device or browser.

[0067] For an application operating at a server side or within a browser at a client device, client requests for data, or user input, is sent to controllers 910, which fetch data from models 912 to pass to views 914. The appropriate output from the views may then be passed back to the client device.

[0068] DOMs 916 assist in rendering views. DOMs 916 are conventions that represent and interact with objects such as views, models, and controllers. DOMs 916 and binders 918 are deployed to a client device only since they are only executable at a client device. Binders 918 interact with event handlers based on user interface input to an application from a user. The event handlers interact with the DOMs 916 which in turn invoke functions in the controllers 910 to update content in views 914. Since DOMs 916 and binders 918 are only client executable, binders receive user input in the form of user interface interactions. The binders 918 interact with the DOMs 916, which communicates with the controllers to facilitate client requests.

[0069] The user input to controllers 910 at a server side occurs when client device binders 918 cannot interact with controllers 910 via DOMs 916. This causes controllers 910 at a server side to run remote network requests that are sent to models 912. The results from models 912 are sent to views 914 to transmit back to the client device for rendering at a client browser.

[0070] FIG. 10 depicts an exemplary client-side application execution in accordance with an embodiment of the present disclosure. Browser 1002 represents a client side browser instance running an application in accordance with the embodiments described herein. Upon activation or clicking of view panel 1004 using cursor 1006, an instruction bound to binder 1008 invoke an action corresponding with view panel 1004. The instruction is sent through module proxy 1010, which serves as a conduit for sending requests to Core 1012 of the application. The instruction is transmitted through module proxy 1010 to controller 1014, which executes the instruction at the client side browser based on sending a request for information to model 1016. The response to execution of the instruction at controller 1014 is transmitted through module proxy 1010, back to binder 1008, which interacts with view panel 1004 that will show the appropriate response.

[0071] FIG. 11 depicts an exemplary server-side application execution in accordance with an embodiment of the present disclosure. Client runtime box 1102 shows all interactions between browser 1104, binder 1106, and module proxy 1108 with respect to a client-side execution of an application. Server runtime box 1110 shows a server responding to instructions sent from module proxy 1108. In this particular example, a certain user input interaction based on cursor 1112 in view panel 1114 is unable to be executed at the client side. Thus, an instruction representing the user input interaction is bound to binder 1106. An action is invoked by binder 1106, which sends the instruction through module proxy 1108, which serves as a conduit for sending requests to different servers that may be hosting or executing the application. Server 1116 receives the instruction, executes the action at the application executing at the server side and transmits a response back through module proxy 1108 to binder 1106, which interacts with view panel 1114 to show the appropriate response to the user input interaction or instruction.

[0072] FIG. 12 depicts an exemplary view of a client browser running an application at a client, and a simulated browser at a server running the same application, in accor-

dance with an embodiment of the present disclosure. Client browser 1202 includes a cursor 1204 for receiving user input to a view panel 1206. Since client browser 1202 cannot execute an instruction related to the user input, the instruction is sent to server 1208. Server 1208, which is executing the same application at a server side will execute the instruction sent by the client browser 1202 and provide a response that is viewable within view panel 1206 of client browser 1202. The simulated browser 1210 includes a view panel 1212, as well as simulated cursor 1214 which corresponds to the activities happening at client browser 1202.

[0073] FIG. 13 depicts an exemplary high level system diagram of application builder 802, in accordance with an embodiment of the present disclosure. Application builder 802, which is responsible for developing and generating platform independent applications, receives instructions for application development, including application parameters, and modules to include in the application. Application builder 802 also receives instructions for adding binders to the application. Binders, as discussed above, are set by an application developer. The binders dictate which event handlers (tied to user interface inputs) may attach with DOMs that further communicate within the architecture of the application. Taking all of the input into account, as well as using application development tools 804, platform agnostic applications are output from application builder 802. These applications may then be deployed to directly to a client device, deployed for serving through a different server, such as, for example, one linked or associated with a mobile application store, or deployed to application hoster 816 for hosting through a cloud based network.

[0074] FIG. 14 depicts an exemplary high level system diagram of package management unit 812, in accordance with an embodiment of the present disclosure. Package management unit 812 is responsible for providing updates to applications in the form of packages to keep deployed applications updated. Package management unit 812 receives as input, lists of required packages and currently installed packages. Package management 812 then uses this received input to cross reference a dependencies database to determine what other packages may be required in addition to those required for the requested updates. Once the entire package list is determined by package management unit 812, the updates in the form of packages are deployed to the applications. This ensures that applications will to the extent possible reflect the latest version.

[0075] In an embodiment, a large set of packages may be a part of a predefined grouping which may be referenced with a small amount of information. For example, a list of required packages may simply reference a predefined grouping of packages as opposed to each package piecemeal.

[0076] In another embodiment, package management unit 812 may receive an update request for a past version of an application. In this scenario, package management unit searches for a corresponding past update package that is compatible with the update request, and provides the relevant packages to the application as required.

[0077] FIG. 15 depicts an exemplary high level system diagram of memory management unit 810, in accordance with an embodiment of the present disclosure. Memory management unit 810 receives as input, memory usage information from a device that an application has been deployed or transmitted to. Prior to execution of the application at the device, memory management unit 810 receives memory

usage information and analyzes the memory usage information. If the memory usage information shows that the device does not have enough free memory to execute the application, memory management unit determines a memory usage scheme for the device to ensure that the application may be executed. The memory usage scheme is then transmitted to the device by memory management unit **810** for implementation at the device.

[0078] FIG. 16 depicts an exemplary high level system diagram of application hoster **816**, in accordance with an embodiment of the present disclosure. Application hoster **816** receives applications developed by application builder **802**. Application hoster **816** can then host these applications and deploy them as necessary, while acting as a server for the applications. Application hoster **816** may also receive requests from client devices or other sources, such as developers to host applications. Application hoster **816**, while hosting applications, can provide results of executing instructions provided to the hosted applications, to client devices. Also, as will be discussed in further detail below, application hoster **816** can analyze analytic information relating to deployment and execution of applications. This information may be provided to developers to assist developers in determining how to further develop applications and tweak existing applications. Since application hoster **816** is a hosting platform that hosts applications developed using application builder **802** in a cloud based environment, application hoster **816** may also host peripheral services, such as privacy controls and parental controls. These peripheral services may be deployed with any of the applications hosted by application hoster **816**.

[0079] FIG. 17 depicts an exemplary high level system diagram of application hoster **816**, in accordance with an embodiment of the present disclosure. Application hoster **816** includes serving unit **1702**, provisioning unit **1706**, analytics unit **1708**, applications **1710**, and peripheral services unit **1712**. Serving unit **1702** receives requests for applications that have been deployed by application hoster **816**. Any requests for applications or to perform particular instructions at a particular application are processed by serving unit **1702** which routes the requests appropriately. Provisioning unit **1706** is responsible for deploying the correct version of applications to client devices, and for transmitting responses to client devices based on instructions received. Since multiple versions of applications exist and may be stored by application hoster **816** (i.e. development application, test application, staging application), application hoster **816** allows developers to deploy different applications that may be hosted, for example, on a distributed network of servers. The distributed network of servers may comprise multiple routers fronted by a hardware load balancer to publish changes to applications as they occur. Analytics unit **1708** may extract metrics from both serving unit **1702** and provisioning unit **1706**. These metrics may be packaged as a set of information and provided to developers to assist in their application development efforts.

[0080] Peripheral services unit **1712** provides peripheral services such as privacy controls and parental controls to applications deployed by application hoster **816**. For example, a hosted application may also receive the benefit of a peripheral service allowing parental controls to be set for the application. If the application were not hosted at application hoster **816**, this option would not be available.

[0081] FIG. 18 depicts an exemplary high level system diagram of application builder **802**, in accordance with an

embodiment of the present disclosure. Application builder **802** includes a parameter definition unit **1804**, framework unit **1806**, and deployment unit **1808**. Parameter definition unit **1804** allows developers to define application parameters of an application allowing the application to be executable on a plurality of device platforms. Parameter definition unit **1804** also allows developers to define configuration parameters of the application and define modules to be included in the application. Framework unit **1808** allows developers to apply a framework to the application facilitating transportable code between a client device and a server to execute the application. Framework unit **1808** may also apply binders to the application to react to user interface events and apply controllers to the application that are bound to the binders, the controllers facilitating a response to the user interface events. The controllers serve data representing responses to user interface events to a view which corresponds to a display element of the application to facilitate display of the responses. Deployment unit **1808** deploys applications that are eventually served to client devices. The deployment unit may deploy the application to an application store, hosting platform, web server, or client device.

[0082] FIG. 19 depicts a flowchart of an exemplary process in which application builder **802** is utilized to develop an application, in accordance with an embodiment of the present disclosure. At **1902**, application builder **802** defines application parameters of an application allowing the application to be executable on a plurality of device parameters. Definition of application parameters is based upon developer input to application builder **802**. Defining the application parameters includes defining configuration parameters of the application dictating how the application will appear and execute on a plurality of devices. Which modules to be included in the application are also defined. At **1904**, application builder **802** applies a framework to the application facilitating transportable code between a client device and a server to execute the application. Application of the framework includes applying binders to the application to react to user interface events. Each binder may, for example, bind to some type of user interface such as user input to the application. Controllers are also applied to the application. These controllers are bound to the binders and facilitate a response to the user interface events. The response may be provided to a client device executing the application. The controllers may serve data representing responses to user interface events to a view which corresponds to a display element of the application facilitating display of the responses. At **1906**, the application is deployed. The deployed application may be served or deployed to at least one of an application store, a hosting platform, a web server, or a client device.

[0083] FIG. 20 depicts a flowchart of an exemplary process in which a user **510** of a device can execute an application that is interoperable across a plurality of device platforms, in accordance with an embodiment of the present disclosure. At **2002**, an application instruction representing a user interface input is received. At **2004**, a binder is invoked based on the application instruction to call a controller. A binder associated with the user interface input is determined, and an action based on the binder is invoked to call an associated controller to handle producing a response to the application instruction. At **2006**, a result is determined with the controller. At **2008**, the result is generated for display at a client device. Generation of the result for display may take place at at least one of the client device or a server executing the application.

[0084] FIG. 21 depicts a flowchart of an exemplary process in which a user 510 of a device can execute an application that is interoperable across a plurality of device platforms, in accordance with an embodiment of the present disclosure. At 2102, an application instruction representing a user interface input is received. At 2104, a binder is invoked based on the application instruction to call a controller. At 2106, a determination is made whether a response to the application instruction can be carried out by a controller at a client device. This may entail determining if the client device has a requisite capability to execute the application instruction locally. The requisite capability may be based upon current network connection with a server, or the ability of the client device platform or operating system to handle the application instruction. At 1808, a controller associated with the binder at a server is called to produce the response to the application instruction. This may entail transmitting the application instruction over a network to the server instructing the server to direct the controller to execute the application instruction and produce a response viewable by the client device. At 2110, the response to the application instruction from the server is received for display at the client device.

[0085] To implement the embodiments set forth herein, computer hardware platforms may be used as hardware platform(s) for one or more of the elements described herein (e.g., application builder 802, application hoster 816, memory management unit 810, and package management unit 812). The hardware elements, operating systems and programming languages of such computer hardware platforms are conventional in nature, and it is presumed that those skilled in the art are adequately familiar therewith to adapt those technologies to implement any of the elements described herein. A computer with user interface elements may be used to implement a personal computer (PC) or other type of workstation or terminal device, although a computer may also act as a server if appropriately programmed. It is believed that those skilled in the art are familiar with the structure, programming, and general operation of such computer equipment, and as a result the drawings are self-explanatory.

[0086] FIG. 22 depicts a general computer architecture on which the present teaching can be implemented and has a functional block diagram illustration of a computer hardware platform which includes user interface elements. The computer may be a general purpose computer or a special purpose computer. This computer 1900 can be used to implement any components of the development and hosting platform described herein. For example, the application builder 802 that develops and generates applications, application hoster 816 which deploys and hosts applications, memory management unit 810 which determines memory schemes, and package management unit 812 which handles updating of packages to applications can all be implemented on a computer such as computer 2200, via its hardware, software program, firmware, or a combination thereof. Although only one such computer is shown, for convenience, the computer functions relating to development and hosting of applications may be implemented in a distributed fashion on a number of similar platforms, to distribute the processing load.

[0087] The computer 2200, for example, includes COM ports 2250 connected to and from a network connected thereto to facilitate data communications. The computer 2200 also includes a central processing unit (CPU) 2220, in the form of one or more processors, for executing program instructions. The exemplary computer platform includes an

internal communication bus 2210, program storage and data storage of different forms, e.g., disk 2270, read only memory (ROM) 2230, or random access memory (RAM) 2240, for various data files to be processed and/or communicated by the computer, as well as possibly program instructions to be executed by the CPU. The computer 2200 also includes an I/O component 2260, supporting input/output flows between the computer and other components therein such as user interface elements 2280. The computer 2200 may also receive programming and data via network communications.

[0088] Hence, aspects of the methods of developing, deploying, and hosting applications that are interoperable across a plurality of device platforms, as outlined above, may be embodied in programming. Program aspects of the technology may be thought of as “products” or “articles of manufacture” typically in the form of executable code and/or associated data that is carried on or embodied in a type of machine readable medium. Tangible non-transitory “storage” type media include any or all of the memory or other storage for the computers, processors or the like, or associated schedules thereof, such as various semiconductor memories, tape drives, disk drives and the like, which may provide storage at any time for the software programming.

[0089] All or portions of the software may at times be communicated through a network such as the Internet or various other telecommunication networks. Such communications, for example, may enable loading of the software from one computer or processor into another, for example, from a server or host computer into the hardware platform(s) of a computing environment or other system implementing a computing environment or similar functionalities in connection with generating explanations based on user inquiries. Thus, another type of media that may bear the software elements includes optical, electrical and electromagnetic waves, such as used across physical inter-faces between local devices, through wired and optical landline networks and over various air-links. The physical elements that carry such waves, such as wired or wireless links, optical links or the like, also may be considered as media bearing the software. As used herein, unless restricted to tangible “storage” media, terms such as computer or machine “readable medium” refer to any medium that participates in providing, instructions to a processor for execution.

[0090] Hence, a machine readable medium may take many forms, including but not limited to, a tangible storage medium, a carrier wave medium or physical transmission medium. Non-volatile storage media include, for example, optical or magnetic disks, such as any of the storage devices in any computer(s) or the like, which may be used to implement the system or any of its components as shown in the drawings. Volatile storage media includes dynamic memory, such as a main memory of such a computer platform. Tangible transmission media includes coaxial cables, copper wire, and fiber optics, including wires that form a bus within a computer system. Carrier-wave transmission media can take the form of electric or electromagnetic signals, or acoustic or light waves such as those generated during radio frequency (RF) and infrared (IR) data communications. Common forms of computer-readable media therefore include for example: a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, DVD or DVD-ROM, any other optical media, punch card paper tapes, any other physical storage medium with patterns of holes, a RAM, a PROM and EPROM, a FLASH-EPROM, any other memory

chip or cartridge, a carrier wave transporting data or instructions, cables or links transporting such a carrier wave, or any other medium from which a computer can read programming code and/or data. Many of these forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to a processor for execution.

[0091] Those skilled in the art will recognize that the embodiments of the present disclosure are amenable to a variety of modifications and/or enhancements. For example, although the implementation of various components described above may be embodied in a hardware device, it can also be implemented as a software only solution—e.g., an installation on an existing server. In addition, the dynamic relation/event detector and its components as disclosed herein can be implemented as firmware, a firmware/software combination, a firmware/hardware combination, or a hardware/firmware/software combination.

[0092] While the foregoing has described what are considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that the teachings may be applied in numerous applications, only some of which have been described herein. It is intended by the following claims to claim and all applications, modifications and variations that fall within the true scope of the present teachings.

What is claimed:

1. A method implemented on at least one computing device, each computing device having at least one processor, storage, and a communication platform connected to a network for providing packages to a client device, the method comprising:

deploying an application on both a client device and a server device;

deploying an updated version of the application on the sever device;

receiving from the client device, at the server device, a request to update the application;

identifying at the server device, one or more packages associated with the updated application deployed on the server device;

prioritizing at the server device, the one or more packages; selecting at the server device, at least one of the one or more packages based on one or more parameters associated with the client device; and

deploying the at least one of the one or more packages on the client device to update the application.

2. The method of claim 1, further comprising:

determining at the server device, a version of the application deployed on the client device from the request; and identifying at the server device, the one or more packages associated with the updated application based on the version of the application deployed on the client device.

3. The method of claim 1, further comprising:

receiving from the client device, at the server device, a preview request related to one of the one or more packages;

obtaining at the server device, a preview resource directed to the one of the one or more packages;

sending the preview resource to be presented on the client device; and

when a selection to install the one of the one or more packages is received from the client device at the service device, deploying the one of the one or more packages on the client device.

4. The method of claim 1, further comprising: receiving at the server device, a current memory usage of the client device; and

selecting the at least one of the one or more packages based on the current memory usage of the client device.

5. The method of claim 1, further comprising: receiving at the server device, a processing capacity of the client device; and

selecting the at least one of the one or more packages based on the processing capacity of the client device.

6. The method of claim 1, further comprising:

incorporating an instruction to the at least one of the one or more packages to be deployed on the client device, wherein the instruction allows the client device to delete the at least one of the one or more packages from the client device upon detecting that the updated application fails to execute on the client device, and roll back to the application before updating.

7. A system for providing packages to a client device comprising at least one processor, a storage, a communication platform connected to a network, wherein the at least one processor is configured by machine-readable instructions to: deploy an application on both a client device and a server device;

deploy an updated version of the application on the sever device;

receive from the client device, at the server device, a request to update the application;

identify at the server device, one or more packages associated with the updated application deployed on the server device;

prioritize at the server device, the one or more packages;

select at the server device, at least one of the one or more packages based on one or more parameters associated with the client device; and

deploy the at least one of the one or more packages on the client device to update the application.

8. The system of claim 7, wherein the at least one processor is further configured to

determine at the server device, a version of the application deployed on the client device from the request; and

identify at the server device, the one or more packages associated with the updated application based on the version of the application deployed on the client device.

9. The system of claim 7, wherein the at least one processor is further configured to

receive from the client device, at the server device, a preview request related to one of the one or more packages;

obtain at the server device, a preview resource directed to the one of the one or more packages;

send the preview resource to be presented on the client device; and

when a selection to install the one of the one or more packages is received from the client device at the service device, deploy the one of the one or more packages on the client device.

10. The system of claim 7, wherein the at least one processor is further configured to

receive at the server device, a current memory usage of the client device; and

select the at least one of the one or more packages based on the current memory usage of the client device.

11. The system of claim 7, wherein the at least one processor is further configured to

- receive at the server device, a processing capacity of the client device; and
- select the at least one of the one or more packages based on the processing capacity of the client device.

12. The system of claim 7, wherein the at least one processor is further configured to

- incorporate an instruction to the at least one of the one or more packages to be deployed on the client device, wherein the instruction allows the client device to delete the at least one of the one or more packages from the client device upon detecting that the updated application fails to execute on the client device, and to roll back to the application before updating.

13. A non-transitory machine readable medium having information recorded for providing packages to a client device, wherein the recorded information, when read by a machine, causes the machine to:

- deploy an application on both a client device and a server device;
- deploy an updated version of the application on the sever device;
- receive from the client device, at the server device, a request to update the application;
- identify at the server device, one or more packages associated with the updated application deployed on the server device;
- prioritize at the server device, the one or more packages;
- select at the server device, at least one of the one or more packages based on one or more parameters associated with the client device; and
- deploy the at least one of the one or more packages on the client device to update the application.

14. The non-transitory machine readable medium of claim 13, wherein the recorded information, when read by the machine, further causes the machine to:

- determine at the server device, a version of the application deployed on the client device from the request; and

- identify at the server device, the one or more packages associated with the updated application based on the version of the application deployed on the client device.

15. The non-transitory machine readable medium of claim 13, wherein the recorded information, when read by the machine, further causes the machine to:

- receive from the client device, at the server device, a preview request related to one of the one or more packages;
- obtain at the server device, a preview resource directed to the one of the one or more packages;
- send the preview resource to be presented on the client device; and
- when a selection to install the one of the one or more packages is received from the client device at the service device, deploy the one of the one or more packages on the client device.

16. The non-transitory machine readable medium of claim 13, wherein the recorded information, when read by the machine, further causes the machine to:

- receive at the server device, a current memory usage of the client device; and
- select the at least one of the one or more packages based on the current memory usage of the client device.

17. The non-transitory machine readable medium of claim 13, wherein the recorded information, when read by the machine, further causes the machine to:

- receive at the server device, a processing capacity of the client device; and
- select the at least one of the one or more packages based on the processing capacity of the client device.

18. The non-transitory machine readable medium of claim 13, wherein the recorded information, when read by the machine, further causes the machine to:

- incorporate an instruction to the at least one of the one or more packages to be deployed on the client device, wherein the instruction allows the client device to delete the at least one of the one or more packages from the client device upon detecting that the updated application fails to execute on the client device, and to roll back to the application before updating.

* * * * *