

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
7 February 2008 (07.02.2008)

PCT

(10) International Publication Number
WO 2008/017015 A2

(51) International Patent Classification:
H04L 29/06 (2006.01)

(74) Agent: **MCKENNA, Christopher, J.**; Choate, Hall & Stewart, Two International Place, Boston, MA 02110 (US).

(21) International Application Number:
PCT/US2007/075043

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

(22) International Filing Date: 2 August 2007 (02.08.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/462,350 3 August 2006 (03.08.2006) US

(71) Applicant (*for all designated States except US*): **CITRIX SYSTEMS, INC.** [US/US]; 851 West Cypress Creek Road, Fort Lauderdale, FL 33309 (US).

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

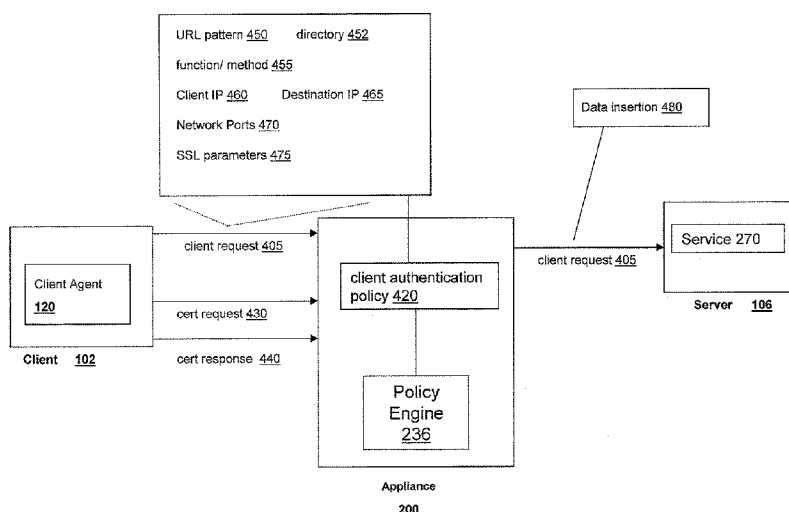
(72) Inventors; and

(75) Inventors/Applicants (*for US only*): **UDUPA, Sivaprasad** [US/US]; C/o Citrix Silicon Valley, 4988 Great America Parkway, Santa Clara, CA 95054 (US). **KANEKAR, Tushar** [IN/US]; C/o Citrix Silicon Valley, 4988 Great America Parkway, Santa Clara, CA 95054 (US). **AG, Tejus** [IN/IN]; C/o Citrix Silicon Valley, 4988 Great America Parkway, Santa Clara, CA 95054 (US).

Published:

— *without international search report and to be republished upon receipt of that report*

(54) Title: SYSTEMS AND METHODS FOR POLICY BASED TRIGGERING OF CLIENT- AUTHENTICATION AT DIRECTORY LEVEL GRANULARITY



(57) Abstract: Systems and methods are disclosed for an appliance to authenticate access of a client to a protected directory on a server via a connection, such as a secure SSL connection, established by the appliance. A method comprises the steps of: receiving, by an appliance, a first request from a client on a first network to access a server on a second network, the appliance providing the client a virtual private network connection from the first network to the second network; determining, by the appliance, the first request comprises access to a protected directory of the server; associating, by the appliance, an authentication policy with the protected directory, the authentication policy specifying an action to authenticate the client's access to the protected directory; and transmitting, by the appliance in response to the authentication policy, a second request to the client for an authentication certificate. Corresponding systems are also disclosed.

SYSTEMS AND METHODS FOR POLICY BASED TRIGGERING OF CLIENT-AUTHENTICATION AT DIRECTORY LEVEL GRANULARITY

Field of the Invention

5

The present invention generally relates to data communication networks. In particular, the present invention relates to systems and methods for providing more granular policy based triggering of client authentication for client requests via an appliance.

10

Background of the Invention

In some cases, a client accesses a resource on a network via a gateway. For example, the client may access resources provide by a server via the gateway. In one case, the client may send a request to a web server to obtain web-pages or other content, such as via a Hypertext Transfer Protocol (HTTP) request. The client request may include a Uniform Resource Locator to request content from a web-page or file located in a directory on a server. The server may include directories that are protected and unprotected. Different clients at different times may access the same resources of a server, such as the protected and unprotected directories.

Typically, a gateway or server authenticates a client on a server level. That is, a client is authenticated to access all the resources of the server, or the client is authenticated to access none of the resources of the server. However, some clients may only access the unprotected directories. In this case, in order to provide access to the unprotected directories, the client would need to be authenticated and thus giving access to the protected directories. In other cases, untrusted clients or clients connected via a virtual private network connection may try to access the protected directories. Yet, other clients may access both the protected and unprotected directories. In additional cases, different types of clients with different types

of users may try to access the protected directories. Since the gateway provides a server level of authentication access to these resources, only all or nothing access can be provided in controlling these different clients and users.

It would, therefore, be desirable to provide systems and methods to providing finer
5 control of triggering client authentication policies based on client requests.

Brief Summary of the Invention

The appliance of the present invention provides a solution to controlling access by different clients to protected directories or to resources desired to be secured. The appliance
10 provides fine-grained triggering of client authentication policy actions based on content and information identified via client requests. The appliance can apply client authentication policies on a per request basis and per client basis. The client authentication policy may indicate to request a certificate from the client or establish (or re-establish) a Secure Socket Layer (SSL) session. Any portion of the client request may be used to trigger a client
15 authentication request from the appliance to the client. For example, any URL pattern, types of functions or methods identified in the request, or internet protocol addresses and ports identified by the request may be used to trigger a client authentication policy. Using any portion of the client request allows fine-grain client authentication policies to be configured, which enables the appliance to provide fine-grain access control to protected resources.
20 Before providing access to a network or server, the appliance can trigger these authentication actions and validate the client. As such, the appliance can control, maintain and protect the level of access to resources accessed via the gateway for a wide variety of client access scenarios. The appliance can be useful in protecting against attacks and probing or zombie clients.

In one aspect, the present invention is related to a method for an appliance to authenticate access of a client to a protected resource on a server via the appliance. The method includes receiving, by an appliance, a client request to access a protected resource of a server, and determining, by the appliance, a portion of the client request matches a corresponding specification of a client authentication policy of the appliance. The method also includes queuing, by the appliance, the client request upon determining the portion of the client request matches the client authentication policy. The appliance determines the client authentication policy identifies an action to request an authentication certificate from the client in order to access the server via the appliance. In response to the client authentication policy, the appliance transmits a request to the client for an authentication certificate.

In one embodiment, the method include determining, by the appliance one or more of the following portions of the client request matches a corresponding specification of the client authentication policy: 1) a Uniform Resource Locator (URL) pattern, 2) an identifier of one of a method or function, 3) a directory, 4) a client network identifier, 5) a server network identifier, 6) a network port, and 7) a Secure Socket Layer (SSL) parameter. In some embodiments, the client authentication policy identifies a pattern for associating a portion of the client request with the client authentication policy. In another embodiment, the method also includes comprising determining, by the appliance, the client has been previously authenticated to access the protected resource, and allowing access, by the appliance, to the protected resource. In some embodiments, the method includes preventing, by the appliance, establishment of a transport layer connection with the server.

In another embodiment, the client authentication policy specifies that the authentication certificate is mandatory. In these embodiments, upon receiving an invalid authentication certificate or not receiving an authentication certificate, the appliance does not transmit the client request to the server. In other embodiments, the client authentication

policy specifies that the authentication certificate is optional. In these embodiments, the appliance transmits the client request to the server upon not receiving an authentication certificate from the client or receiving an invalid authentication certificate from the client. In one embodiment, the method includes inserting, by the appliance, into the client request a portion of the client's response to the request for the authentication certificate, and transmitting the client request to the server. In yet another embodiment, the method includes inserting, by the appliance in response to the client authentication policy, data related to the authentication certificate into a Hypertext Transfer Protocol (HTTP) header of the client request, and transmitting the client request to the server. In some embodiments, the method includes inserting, by the appliance in response to the client authentication policy, Secure Socket Layer (SSL) information into a Hypertext Transfer Protocol (HTTP) header of the client request, and transmitting the client request to the server.

In another aspect, the present invention is related to an appliance for providing finer control for authenticating access of a client to a protected resource on a server. The appliance includes means for receiving a client request to access a protected resource of a server, and for determining a portion of the client request matches a corresponding specification of a client authentication policy of the appliance. The appliance also includes means for queuing the client request upon determining the portion of the client request matches the client authentication policy, and determining the client authentication policy identifies an action to request an authentication certificate from the client in order to access the server via the appliance. The appliance also includes means for transmitting, in response to the client authentication policy, a request to the client for an authentication certificate.

In some embodiments, the appliance determines one or more of the following portions of the client request matches a corresponding specification of the client authentication policy:

a Uniform Resource Locator (URL) pattern, an identifier of one of a method or function, a

directory, a client network identifier, a server network identifier, a network port, and a Secure Socket Layer (SSL) parameter. In some embodiments, the client authentication policy identifies a pattern for associating a portion of the client request with the client authentication policy. In another embodiment, the appliance determines the client has been previously
5 authenticated to access the protected resource, and allowing access, by the appliance, to the protected resource.

In one embodiment, the client authentication policy specifies that the authentication certificate is mandatory. In these embodiments, upon receiving an invalid authentication certificate or not receiving an authentication certificate, the appliance does not transmit the
10 client request to the server. In other embodiments, the client authentication policy specifies that the authentication certificate is optional. In these embodiments, the appliance includes means for transmitting the client request to the server upon not receiving an authentication certificate from the client or receiving an invalid authentication certificate from the client. In another embodiment, the appliance includes means inserting into the client request a portion
15 of the client's response to the request for the authentication certificate, and transmitting the client request to the server. In some embodiments, the appliance prevents establishment of a transport layer connection with the server.

In yet another embodiment, the appliance includes means for inserting, in response to the client authentication policy, data related to the authentication certificate into a Hypertext
20 Transfer Protocol (HTTP) header of the client request, and transmitting the client request to the server. In some embodiments, the appliance includes means for inserting, by the appliance in response to the client authentication policy, Secure Socket Layer (SSL) information into a Hypertext Transfer Protocol (HTTP) header request of the client request, and transmitting the client request to the server

The details of various embodiments of the invention are set forth in the accompanying drawings and the description below.

Brief Description of the Figures

The foregoing and other objects, aspects, features, and advantages of the invention will become more apparent and better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1A is a block diagram of an embodiment of a network environment for a client to access a server via an appliance;

FIG. 1B is a block diagram of an embodiment of an environment for delivering a computing environment from a server to a client via an appliance;

FIGs. 1C and 1D are block diagrams of embodiments of a computing device;

FIG. 2A is a block diagram of an embodiment of an appliance for processing communications between a client and a server;

FIG. 2B is a block diagram of another embodiment of an appliance for optimizing, accelerating, load-balancing and routing communications between a client and a server;

FIG. 3 is a block diagram of an embodiment of a client for communicating with a server via the appliance;

FIG. 4 is a block diagram of an embodiment of a system for providing policy based trigger of client authentication via an appliance based on portions of the client request; and

FIG. 5 is a flow diagram of steps of an embodiment of a method for practicing policy based triggering of client authentication based on portions of the client request.

The features and advantages of the present invention will become more apparent from the detailed description set forth below when taken in conjunction with the drawings, in which like reference characters identify corresponding elements throughout. In the drawings,

like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements.

Detailed Description of the Invention

A. Network and Computing Environment

5 Prior to discussing the specifics of embodiments of the systems and methods of an appliance and/or client, it may be helpful to discuss the network and computing environments in which such embodiments may be deployed. Referring now to Figure 1A, an embodiment of a network environment is depicted. In brief overview, the network environment comprises one or more clients 102a-102n (also generally referred to as local machine(s) 102, or client(s) 102) in communication with one or more servers 106a-106n (also generally referred to as server(s) 106, or remote machine(s) 106) via one or more networks 104, 104' (generally referred to as network 104). In some embodiments, a client 102 communicates with a server 106 via an appliance 200.

15 Although FIG. 1A shows a network 104 and a network 104' between the clients 102 and the servers 106, the clients 102 and the servers 106 may be on the same network 104. The networks 104 and 104' can be the same type of network or different types of networks. The network 104 and/or the network 104' can be a local-area network (LAN), such as a company Intranet, a metropolitan area network (MAN), or a wide area network (WAN), such as the Internet or the World Wide Web. In one embodiment, network 104' may be a private network and network 104 may be a public network. In some embodiments, network 104 may be a private network and network 104' a public network. In another embodiment, networks 20 104 and 104' may both be private networks. In some embodiments, clients 102 may be located at a branch office of a corporate enterprise communicating via a WAN connection over the network 104 to the servers 106 located at a corporate data center.

The network 104 and/or 104' be any type and/or form of network and may include any of the following: a point to point network, a broadcast network, a wide area network, a local area network, a telecommunications network, a data communication network, a computer network, an ATM (Asynchronous Transfer Mode) network, a SONET

5 (Synchronous Optical Network) network, a SDH (Synchronous Digital Hierarchy) network, a wireless network and a wireline network. In some embodiments, the network 104 may comprise a wireless link, such as an infrared channel or satellite band. The topology of the network 104 and/or 104' may be a bus, star, or ring network topology. The network 104 and/or 104' and network topology may be of any such network or network topology as
10 known to those ordinarily skilled in the art capable of supporting the operations described herein.

As shown in FIG. 1A, the appliance 200, which also may be referred to as an interface unit 200 or gateway 200, is shown between the networks 104 and 104'. In some embodiments, the appliance 200 may be located on network 104. For example, a branch
15 office of a corporate enterprise may deploy an appliance 200 at the branch office. In other embodiments, the appliance 200 may be located on network 104'. For example, an appliance 200 may be located at a corporate data center. In yet another embodiment, a plurality of appliances 200 may be deployed on network 104. In some embodiments, a plurality of appliances 200 may be deployed on network 104'. In one embodiment, a first appliance 200
20 communicates with a second appliance 200'. In other embodiments, the appliance 200 could be a part of any client 102 or server 106 on the same or different network 104, 104' as the client 102. One or more appliances 200 may be located at any point in the network or network communications path between a client 102 and a server 106.

In one embodiment, the system may include multiple, logically-grouped servers 106.
25 In these embodiments, the logical group of servers may be referred to as a server farm 38. In

some of these embodiments, the servers 106 may be geographically dispersed. In some cases, a farm 38 may be administered as a single entity. In other embodiments, the server farm 38 comprises a plurality of server farms 38. In one embodiment, the server farm executes one or more applications on behalf of one or more clients 102.

5 The servers 106 within each farm 38 can be heterogeneous. One or more of the servers 106 can operate according to one type of operating system platform (e.g., WINDOWS NT, manufactured by Microsoft Corp. of Redmond, Washington), while one or more of the other servers 106 can operate on according to another type of operating system platform (e.g., Unix or Linux). The servers 106 of each farm 38 do not need to be physically proximate to
10 another server 106 in the same farm 38. Thus, the group of servers 106 logically grouped as a farm 38 may be interconnected using a wide-area network (WAN) connection or medium-area network (MAN) connection. For example, a farm 38 may include servers 106 physically located in different continents or different regions of a continent, country, state, city, campus, or room. Data transmission speeds between servers 106 in the farm 38 can be increased if the
15 servers 106 are connected using a local-area network (LAN) connection or some form of direct connection.

 Servers 106 may be referred to as a file server, application server, web server, proxy server, or gateway server. In some embodiments, a server 106 may have the capacity to function as either an application server or as a master application server. In one embodiment,
20 a server 106 may include an Active Directory. The clients 102 may also be referred to as client nodes or endpoints. In some embodiments, a client 102 has the capacity to function as both a client node seeking access to applications on a server and as an application server providing access to hosted applications for other clients 102a-102n.

 In some embodiments, a client 102 communicates with a server 106. In one
25 embodiment, the client 102 communicates directly with one of the servers 106 in a farm 38.

In another embodiment, the client 102 executes a program neighborhood application to communicate with a server 106 in a farm 38. In still another embodiment, the server 106 provides the functionality of a master node. In some embodiments, the client 102 communicates with the server 106 in the farm 38 through a network 104. Over the network 104, the client 102 can, for example, request execution of various applications hosted by the servers 106a-106n in the farm 38 and receive output of the results of the application execution for display. In some embodiments, only the master node provides the functionality required to identify and provide address information associated with a server 106' hosting a requested application.

In one embodiment, the server 106 provides functionality of a web server. In another embodiment, the server 106a receives requests from the client 102, forwards the requests to a second server 106b and responds to the request by the client 102 with a response to the request from the server 106b. In still another embodiment, the server 106 acquires an enumeration of applications available to the client 102 and address information associated with a server 106 hosting an application identified by the enumeration of applications. In yet another embodiment, the server 106 presents the response to the request to the client 102 using a web interface. In one embodiment, the client 102 communicates directly with the server 106 to access the identified application. In another embodiment, the client 102 receives application output data, such as display data, generated by an execution of the identified application on the server 106.

Referring now to FIG. 1B, a network environment for delivering and/or operating a computing environment on a client 102 is depicted. In some embodiments, a server 106 includes an application delivery system 190 for delivering a computing environment or an application and/or data file to one or more clients 102. In brief overview, a client 10 is in communication with a server 106 via network 104, 104' and appliance 200. For example, the

client 102 may reside in a remote office of a company, e.g., a branch office, and the server 106 may reside at a corporate data center. The client 102 comprises a client agent 120, and a computing environment 15. The computing environment 15 may execute or operate an application that accesses, processes or uses a data file. The computing environment 15, application and/or data file may be delivered via the appliance 200 and/or the server 106.

In some embodiments, the appliance 200 accelerates delivery of a computing environment 15, or any portion thereof, to a client 102. In one embodiment, the appliance 200 accelerates the delivery of the computing environment 15 by the application delivery system 190. For example, the embodiments described herein may be used to accelerate delivery of a streaming application and data file processable by the application from a central corporate data center to a remote user location, such as a branch office of the company. In another embodiment, the appliance 200 accelerates transport layer traffic between a client 102 and a server 106. The appliance 200 may provide acceleration techniques for accelerating any transport layer payload from a server 106 to a client 102, such as: 1) transport layer connection pooling, 2) transport layer connection multiplexing, 3) transport control protocol buffering, 4) compression and 5) caching. In some embodiments, the appliance 200 provides load balancing of servers 106 in responding to requests from clients 102. In other embodiments, the appliance 200 acts as a proxy or access server to provide access to the one or more servers 106. In another embodiment, the appliance 200 provides a secure virtual private network connection from a first network 104 of the client 102 to the second network 104' of the server 106, such as an SSL VPN connection. In yet other embodiments, the appliance 200 provides application firewall security, control and management of the connection and communications between a client 102 and a server 106.

In some embodiments, the application delivery management system 190 provides application delivery techniques to deliver a computing environment to a desktop of a user,

remote or otherwise, based on a plurality of execution methods and based on any authentication and authorization policies applied via a policy engine 195. With these techniques, a remote user may obtain a computing environment and access to server stored applications and data files from any network connected device 100. In one embodiment, the application delivery system 190 may reside or execute on a server 106. In another embodiment, the application delivery system 190 may reside or execute on a plurality of servers 106a-106n. In some embodiments, the application delivery system 190 may execute in a server farm 38. In one embodiment, the server 106 executing the application delivery system 190 may also store or provide the application and data file. In another embodiment, a first set of one or more servers 106 may execute the application delivery system 190, and a different server 106n may store or provide the application and data file. In some embodiments, each of the application delivery system 190, the application, and data file may reside or be located on different servers. In yet another embodiment, any portion of the application delivery system 190 may reside, execute or be stored on or distributed to the appliance 200, or a plurality of appliances.

The client 102 may include a computing environment 15 for executing an application that uses or processes a data file. The client 102 via networks 104, 104' and appliance 200 may request an application and data file from the server 106. In one embodiment, the appliance 200 may forward a request from the client 102 to the server 106. For example, the client 102 may not have the application and data file stored or accessible locally. In response to the request, the application delivery system 190 and/or server 106 may deliver the application and data file to the client 102. For example, in one embodiment, the server 106 may transmit the application as an application stream to operate in computing environment 15 on client 102.

In some embodiments, the application delivery system 190 comprises any portion of the Citrix Access Suite™ by Citrix Systems, Inc., such as the MetaFrame or Citrix Presentation Server™ and/or any of the Microsoft® Windows Terminal Services manufactured by the Microsoft Corporation. In one embodiment, the application delivery system 190 may deliver one or more applications to clients 102 or users via a remote-display protocol or otherwise via remote-based or server-based computing. In another embodiment, the application delivery system 190 may deliver one or more applications to clients or users via steaming of the application.

In one embodiment, the application delivery system 190 includes a policy engine 195 for controlling and managing the access to, selection of application execution methods and the delivery of applications. In some embodiments, the policy engine 195 determines the one or more applications a user or client 102 may access. In another embodiment, the policy engine 195 determines how the application should be delivered to the user or client 102, e.g., the method of execution. In some embodiments, the application delivery system 190 provides a plurality of delivery techniques from which to select a method of application execution, such as a server-based computing, streaming or delivering the application locally to the client 120 for local execution.

In one embodiment, a client 102 requests execution of an application program and the application delivery system 190 comprising a server 106 selects a method of executing the application program. In some embodiments, the server 106 receives credentials from the client 102. In another embodiment, the server 106 receives a request for an enumeration of available applications from the client 102. In one embodiment, in response to the request or receipt of credentials, the application delivery system 190 enumerates a plurality of application programs available to the client 102. The application delivery system 190 receives a request to execute an enumerated application. The application delivery system 190

selects one of a predetermined number of methods for executing the enumerated application, for example, responsive to a policy of a policy engine. The application delivery system 190 may select a method of execution of the application enabling the client 102 to receive application-output data generated by execution of the application program on a server 106.

- 5 The application delivery system 190 may select a method of execution of the application enabling the local machine 10 to execute the application program locally after retrieving a plurality of application files comprising the application. In yet another embodiment, the application delivery system 190 may select a method of execution of the application to stream the application via the network 104 to the client 102.

- 10 A client 102 may execute, operate or otherwise provide an application, which can be any type and/or form of software, program, or executable instructions such as any type and/or form of web browser, web-based client, client-server application, a thin-client computing client, an ActiveX control, or a Java applet, or any other type and/or form of executable instructions capable of executing on client 102. In some embodiments, the application may
- 15 be a server-based or a remote-based application executed on behalf of the client 102 on a server 106. In one embodiment the server 106 may display output to the client 102 using any thin-client or remote-display protocol, such as the Independent Computing Architecture (ICA) protocol manufactured by Citrix Systems, Inc. of Ft. Lauderdale, Florida or the Remote Desktop Protocol (RDP) manufactured by the Microsoft Corporation of Redmond,
- 20 Washington. The application can use any type of protocol and it can be, for example, an HTTP client, an FTP client, an Oscar client, or a Telnet client. In other embodiments, the application comprises any type of software related to VoIP communications, such as a soft IP telephone. In further embodiments, the application comprises any application related to real-time data communications, such as applications for streaming video and/or audio.

In some embodiments, the server 106 or a server farm 38 may be running one or more applications, such as an application providing a thin-client computing or remote display presentation application. In one embodiment, the server 106 or server farm 38 executes as an application, any portion of the Citrix Access Suite™ by Citrix Systems, Inc., such as the
5 MetaFrame or Citrix Presentation Server™, and/or any of the Microsoft® Windows Terminal Services manufactured by the Microsoft Corporation. In one embodiment, the application is an ICA client, developed by Citrix Systems, Inc. of Fort Lauderdale, Florida. In other embodiments, the application includes a Remote Desktop (RDP) client, developed by Microsoft Corporation of Redmond, Washington. Also, the server 106 may run an
10 application, which for example, may be an application server providing email services such as Microsoft Exchange manufactured by the Microsoft Corporation of Redmond, Washington, a web or Internet server, or a desktop sharing server, or a collaboration server. In some embodiments, any of the applications may comprise any type of hosted service or products, such as GoToMeeting™ provided by Citrix Online Division, Inc. of Santa Barbara,
15 California, WebEx™ provided by WebEx, Inc. of Santa Clara, California, or Microsoft Office Live Meeting provided by Microsoft Corporation of Redmond, Washington.

The client 102, server 106, and appliance 200 may be deployed as and/or executed on any type and form of computing device, such as a computer, network device or appliance capable of communicating on any type and form of network and performing the operations
20 described herein. FIGs. 1C and 1D depict block diagrams of a computing device 100 useful for practicing an embodiment of the client 102, server 106 or appliance 200. As shown in FIGs. 1C and 1D, each computing device 100 includes a central processing unit 101, and a main memory unit 122. As shown in FIG. 1C, a computing device 100 may include a visual display device 124, a keyboard 126 and/or a pointing device 127, such as a mouse. Each
25 computing device 100 may also include additional optional elements, such as one or more

input/output devices 130a-130b (generally referred to using reference numeral 130), and a cache memory 140 in communication with the central processing unit 101.

The central processing unit 101 is any logic circuitry that responds to and processes instructions fetched from the main memory unit 122. In many embodiments, the central processing unit is provided by a microprocessor unit, such as: those manufactured by Intel Corporation of Mountain View, California; those manufactured by Motorola Corporation of Schaumburg, Illinois; those manufactured by Transmeta Corporation of Santa Clara, California; the RS/6000 processor, those manufactured by International Business Machines of White Plains, New York; or those manufactured by Advanced Micro Devices of Sunnyvale, California. The computing device 100 may be based on any of these processors, or any other processor capable of operating as described herein.

Main memory unit 122 may be one or more memory chips capable of storing data and allowing any storage location to be directly accessed by the microprocessor 101, such as Static random access memory (SRAM), Burst SRAM or SynchBurst SRAM (BSRAM), Dynamic random access memory (DRAM), Fast Page Mode DRAM (FPM DRAM), Enhanced DRAM (EDRAM), Extended Data Output RAM (EDO RAM), Extended Data Output DRAM (EDO DRAM), Burst Extended Data Output DRAM (BEDO DRAM), Enhanced DRAM (EDRAM), synchronous DRAM (SDRAM), JEDEC SRAM, PC100 SDRAM, Double Data Rate SDRAM (DDR SDRAM), Enhanced SDRAM (ESDRAM), SyncLink DRAM (SLDRAM), Direct Rambus DRAM (DRDRAM), or Ferroelectric RAM (FRAM). The main memory 122 may be based on any of the above described memory chips, or any other available memory chips capable of operating as described herein. In the embodiment shown in FIG. 1C, the processor 101 communicates with main memory 122 via a system bus 150 (described in more detail below). FIG. 1C depicts an embodiment of a

computing device 100 in which the processor communicates directly with main memory 122 via a memory port 103. For example, in FIG. 1D the main memory 122 may be DRDRAM.

FIG. 1D depicts an embodiment in which the main processor 101 communicates directly with cache memory 140 via a secondary bus, sometimes referred to as a backside bus. In other embodiments, the main processor 101 communicates with cache memory 140 using the system bus 150. Cache memory 140 typically has a faster response time than main memory 122 and is typically provided by SRAM, BSRAM, or EDRAM. In the embodiment shown in FIG. 1C, the processor 101 communicates with various I/O devices 130 via a local system bus 150. Various busses may be used to connect the central processing unit 101 to any of the I/O devices 130, including a VESA VL bus, an ISA bus, an EISA bus, a MicroChannel Architecture (MCA) bus, a PCI bus, a PCI-X bus, a PCI-Express bus, or a NuBus. For embodiments in which the I/O device is a video display 124, the processor 101 may use an Advanced Graphics Port (AGP) to communicate with the display 124. FIG. 1D depicts an embodiment of a computer 100 in which the main processor 101 communicates directly with I/O device 130 via HyperTransport, Rapid I/O, or InfiniBand. FIG. 1D also depicts an embodiment in which local busses and direct communication are mixed: the processor 101 communicates with I/O device 130 using a local interconnect bus while communicating with I/O device 130 directly.

The computing device 100 may support any suitable installation device 116, such as a floppy disk drive for receiving floppy disks such as 3.5-inch, 5.25-inch disks or ZIP disks, a CD-ROM drive, a CD-R/RW drive, a DVD-ROM drive, tape drives of various formats, USB device, hard-drive or any other device suitable for installing software and programs such as any client agent 120, or portion thereof. The computing device 100 may further comprise a storage device 128, such as one or more hard disk drives or redundant arrays of independent disks, for storing an operating system and other related software, and for storing application

software programs such as any program related to the client agent 120. Optionally, any of the installation devices 116 could also be used as the storage device 128. Additionally, the operating system and the software can be run from a bootable medium, for example, a bootable CD, such as KNOPPIX®, a bootable CD for GNU/Linux that is available as a
5 GNU/Linux distribution from knoppix.net.

Furthermore, the computing device 100 may include a network interface 118 to interface to a Local Area Network (LAN), Wide Area Network (WAN) or the Internet through a variety of connections including, but not limited to, standard telephone lines, LAN or WAN links (*e.g.*, 802.11, T1, T3, 56kb, X.25), broadband connections (*e.g.*, ISDN, Frame
10 Relay, ATM), wireless connections, or some combination of any or all of the above. The network interface 118 may comprise a built-in network adapter, network interface card, PCMCIA network card, card bus network adapter, wireless network adapter, USB network adapter, modem or any other device suitable for interfacing the computing device 100 to any type of network capable of communication and performing the operations described herein.

15 A wide variety of I/O devices 130a-130n may be present in the computing device 100. Input devices include keyboards, mice, trackpads, trackballs, microphones, and drawing tablets. Output devices include video displays, speakers, inkjet printers, laser printers, and dye-sublimation printers. The I/O devices 130 may be controlled by an I/O controller 123 as shown in FIG. 1C. The I/O controller may control one or more I/O devices such as a
20 keyboard 126 and a pointing device 127, *e.g.*, a mouse or optical pen. Furthermore, an I/O device may also provide storage 128 and/or an installation medium 116 for the computing device 100. In still other embodiments, the computing device 100 may provide USB connections to receive handheld USB storage devices such as the USB Flash Drive line of devices manufactured by Twintech Industry, Inc. of Los Alamitos, California.

In some embodiments, the computing device 100 may comprise or be connected to multiple display devices 124a-124n, which each may be of the same or different type and/or form. As such, any of the I/O devices 130a-130n and/or the I/O controller 123 may comprise any type and/or form of suitable hardware, software, or combination of hardware and software to support, enable or provide for the connection and use of multiple display devices 124a-124n by the computing device 100. For example, the computing device 100 may include any type and/or form of video adapter, video card, driver, and/or library to interface, communicate, connect or otherwise use the display devices 124a-124n. In one embodiment, a video adapter may comprise multiple connectors to interface to multiple display devices 124a-124n. In other embodiments, the computing device 100 may include multiple video adapters, with each video adapter connected to one or more of the display devices 124a-124n. In some embodiments, any portion of the operating system of the computing device 100 may be configured for using multiple displays 124a-124n. In other embodiments, one or more of the display devices 124a-124n may be provided by one or more other computing devices, such as computing devices 100a and 100b connected to the computing device 100, for example, via a network. These embodiments may include any type of software designed and constructed to use another computer's display device as a second display device 124a for the computing device 100. One ordinarily skilled in the art will recognize and appreciate the various ways and embodiments that a computing device 100 may be configured to have multiple display devices 124a-124n.

In further embodiments, an I/O device 130 may be a bridge 170 between the system bus 150 and an external communication bus, such as a USB bus, an Apple Desktop Bus, an RS-232 serial connection, a SCSI bus, a FireWire bus, a FireWire 800 bus, an Ethernet bus, an AppleTalk bus, a Gigabit Ethernet bus, an Asynchronous Transfer Mode bus, a HIPPI bus,

a Super HIPPI bus, a SerialPlus bus, a SCI/LAMP bus, a FibreChannel bus, or a Serial Attached small computer system interface bus.

A computing device 100 of the sort depicted in FIGs. 1C and 1D typically operate under the control of operating systems, which control scheduling of tasks and access to system resources. The computing device 100 can be running any operating system such as any of the versions of the Microsoft® Windows operating systems, the different releases of the Unix and Linux operating systems, any version of the Mac OS® for Macintosh computers, any embedded operating system, any real-time operating system, any open source operating system, any proprietary operating system, any operating systems for mobile computing devices, or any other operating system capable of running on the computing device and performing the operations described herein. Typical operating systems include: WINDOWS 3.x, WINDOWS 95, WINDOWS 98, WINDOWS 2000, WINDOWS NT 3.51, WINDOWS NT 4.0, WINDOWS CE, and WINDOWS XP, all of which are manufactured by Microsoft Corporation of Redmond, Washington; MacOS, manufactured by Apple Computer of Cupertino, California; OS/2, manufactured by International Business Machines of Armonk, New York; and Linux, a freely-available operating system distributed by Caldera Corp. of Salt Lake City, Utah, or any type and/or form of a Unix operating system, among others.

In other embodiments, the computing device 100 may have different processors, operating systems, and input devices consistent with the device. For example, in one embodiment the computer 100 is a Treo 180, 270, 1060, 600 or 650 smart phone manufactured by Palm, Inc. In this embodiment, the Treo smart phone is operated under the control of the PalmOS operating system and includes a stylus input device as well as a five-way navigator device. Moreover, the computing device 100 can be any workstation, desktop computer, laptop or notebook computer, server, handheld computer, mobile telephone, any

other computer, or other form of computing or telecommunications device that is capable of communication and that has sufficient processor power and memory capacity to perform the operations described herein.

5

B. Appliance Architecture

FIG. 2A illustrates an example embodiment of the appliance 200. The architecture of the appliance 200 in FIG. 2A is provided by way of illustration only and is not intended to be limiting. As shown in FIG. 2, appliance 200 comprises a hardware layer 206 and a software layer divided into a user space 202 and a kernel space 204.

Hardware layer 206 provides the hardware elements upon which programs and services within kernel space 204 and user space 202 are executed. Hardware layer 206 also provides the structures and elements which allow programs and services within kernel space 204 and user space 202 to communicate data both internally and externally with respect to appliance 200. As shown in FIG. 2, the hardware layer 206 includes a processing unit 262 for executing software programs and services, a memory 264 for storing software and data, network ports 266 for transmitting and receiving data over a network, and an encryption processor 260 for performing functions related to Secure Sockets Layer processing of data transmitted and received over the network. In some embodiments, the central processing unit 262 may perform the functions of the encryption processor 260 in a single processor. Additionally, the hardware layer 206 may comprise multiple processors for each of the processing unit 262 and the encryption processor 260. The processor 262 may include any of the processors 101 described above in connection with FIGs. 1C and 1D. In some embodiments, the central processing unit 262 may perform the functions of the encryption processor 260 in a single processor. Additionally, the hardware layer 206 may comprise multiple processors for each of the processing unit 262 and the encryption processor 260. For

example, in one embodiment, the appliance 200 comprises a first processor 262 and a second processor 262'. In other embodiments, the processor 262 or 262' comprises a multi-core processor.

Although the hardware layer 206 of appliance 200 is generally illustrated with an encryption processor 260, processor 260 may be a processor for performing functions related to any encryption protocol, such as the Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocol. In some embodiments, the processor 260 may be a general purpose processor (GPP), and in further embodiments, may be have executable instructions for performing processing of any security related protocol.

Although the hardware layer 206 of appliance 200 is illustrated with certain elements in FIG. 2, the hardware portions or components of appliance 200 may comprise any type and form of elements, hardware or software, of a computing device, such as the computing device 100 illustrated and discussed herein in conjunction with FIGs. 1C and 1D. In some embodiments, the appliance 200 may comprise a server, gateway, router, switch, bridge or other type of computing or network device, and have any hardware and/or software elements associated therewith.

The operating system of appliance 200 allocates, manages, or otherwise segregates the available system memory into kernel space 204 and user space 204. In example software architecture 200, the operating system may be any type and/or form of Unix operating system although the invention is not so limited. As such, the appliance 200 can be running any operating system such as any of the versions of the Microsoft® Windows operating systems, the different releases of the Unix and Linux operating systems, any version of the Mac OS® for Macintosh computers, any embedded operating system, any network operating system, any real-time operating system, any open source operating system, any proprietary operating system, any operating systems for mobile computing devices or network devices, or any other

operating system capable of running on the appliance 200 and performing the operations described herein.

The kernel space 204 is reserved for running the kernel 230, including any device drivers, kernel extensions or other kernel related software. As known to those skilled in the art, the kernel 230 is the core of the operating system, and provides access, control, and management of resources and hardware-related elements of the application 104. In accordance with an embodiment of the appliance 200, the kernel space 204 also includes a number of network services or processes working in conjunction with a cache manager 232. sometimes also referred to as the integrated cache, the benefits of which are described in detail further herein. Additionally, the embodiment of the kernel 230 will depend on the embodiment of the operating system installed, configured, or otherwise used by the device 200.

In one embodiment, the device 200 comprises one network stack 267, such as a TCP/IP based stack, for communicating with the client 102 and/or the server 106. In one embodiment, the network stack 267 is used to communicate with a first network, such as network 108, and a second network 110. In some embodiments, the device 200 terminates a first transport layer connection, such as a TCP connection of a client 102, and establishes a second transport layer connection to a server 106 for use by the client 102, e.g., the second transport layer connection is terminated at the appliance 200 and the server 106. The first and second transport layer connections may be established via a single network stack 267. In other embodiments, the device 200 may comprise multiple network stacks, for example 267 and 267', and the first transport layer connection may be established or terminated at one network stack 267, and the second transport layer connection on the second network stack 267'. For example, one network stack may be for receiving and transmitting network packet on a first network, and another network stack for receiving and transmitting network packets

on a second network. In one embodiment, the network stack 267 comprises a buffer 243 for queuing one or more network packets for transmission by the appliance 200.

As shown in FIG. 2, the kernel space 204 includes the cache manager 232, a high-speed layer 2-7 integrated packet engine 240, an encryption engine 234, a policy engine 236 and multi-protocol compression logic 238. Running these components or processes 232, 240, 234, 236 and 238 in kernel space 204 or kernel mode instead of the user space 202 improves the performance of each of these components, alone and in combination. Kernel operation means that these components or processes 232, 240, 234, 236 and 238 run in the core address space of the operating system of the device 200. For example, running the encryption engine 234 in kernel mode improves encryption performance by moving encryption and decryption operations to the kernel, thereby reducing the number of transitions between the memory space or a kernel thread in kernel mode and the memory space or a thread in user mode. For example, data obtained in kernel mode may not need to be passed or copied to a process or thread running in user mode, such as from a kernel level data structure to a user level data structure. In another aspect, the number of context switches between kernel mode and user mode are also reduced. Additionally, synchronization of and communications between any of the components or processes 232, 240, 235, 236 and 238 can be performed more efficiently in the kernel space 204.

In some embodiments, any portion of the components 232, 240, 234, 236 and 238 may run or operate in the kernel space 204, while other portions of these components 232, 240, 234, 236 and 238 may run or operate in user space 202. In one embodiment, the appliance 200 uses a kernel-level data structure providing access to any portion of one or more network packets, for example, a network packet comprising a request from a client 102 or a response from a server 106. In some embodiments, the kernel-level data structure may be obtained by the packet engine 240 via a transport layer driver interface or filter to the

network stack 267. The kernel-level data structure may comprise any interface and/or data accessible via the kernel space 204 related to the network stack 267, network traffic or packets received or transmitted by the network stack 267. In other embodiments, the kernel-level data structure may be used by any of the components or processes 232, 240, 234, 236 and 238 to perform the desired operation of the component or process. In one embodiment, a component 232, 240, 234, 236 and 238 is running in kernel mode 204 when using the kernel-level data structure, while in another embodiment, the component 232, 240, 234, 236 and 238 is running in user mode when using the kernel-level data structure. In some embodiments, the kernel-level data structure may be copied or passed to a second kernel-level data structure, or any desired user-level data structure.

The cache manager 232 may comprise software, hardware or any combination of software and hardware to provide cache access, control and management of any type and form of content, such as objects or dynamically generated objects served by the originating servers 106. The data, objects or content processed and stored by the cache manager 232 may comprise data in any format, such as a markup language, or communicated via any protocol. In some embodiments, the cache manager 232 duplicates original data stored elsewhere or data previously computed, generated or transmitted, in which the original data may require longer access time to fetch, compute or otherwise obtain relative to reading a cache memory element. Once the data is stored in the cache memory element, future use can be made by accessing the cached copy rather than refetching or recomputing the original data, thereby reducing the access time. In some embodiments, the cache memory element may comprise a data object in memory 264 of device 200. In other embodiments, the cache memory element may comprise memory having a faster access time than memory 264. In another embodiment, the cache memory element may comprise any type and form of storage element of the device 200, such as a portion of a hard disk. In some embodiments, the

processing unit 262 may provide cache memory for use by the cache manager 232. In yet further embodiments, the cache manager 232 may use any portion and combination of memory, storage, or the processing unit for caching data, objects, and other content.

Furthermore, the cache manager 232 includes any logic, functions, rules, or operations to perform any embodiments of the techniques of the appliance 200 described herein. For example, the cache manager 232 includes logic or functionality to invalidate objects based on the expiration of an invalidation time period or upon receipt of an invalidation command from a client 102 or server 106. In some embodiments, the cache manager 232 may operate as a program, service, process or task executing in the kernel space 204, and in other embodiments, in the user space 202. In one embodiment, a first portion of the cache manager 232 executes in the user space 202 while a second portion executes in the kernel space 204. In some embodiments, the cache manager 232 can comprise any type of general purpose processor (GPP), or any other type of integrated circuit, such as a Field Programmable Gate Array (FPGA), Programmable Logic Device (PLD), or Application Specific Integrated Circuit (ASIC).

The policy engine 236 may include, for example, an intelligent statistical engine or other programmable application(s). In one embodiment, the policy engine 236 provides a configuration mechanism to allow a user to identifying, specify, define or configure a caching policy. Policy engine 236, in some embodiments, also has access to memory to support data structures such as lookup tables or hash tables to enable user-selected caching policy decisions. In other embodiments, the policy engine 236 may comprise any logic, rules, functions or operations to determine and provide access, control and management of objects, data or content being cached by the appliance 200 in addition to access, control and management of security, network traffic, network access, compression or any other function

or operation performed by the appliance 200. Further examples of specific caching policies are further described herein.

The encryption engine 234 comprises any logic, business rules, functions or operations for handling the processing of any security related protocol, such as SSL or TLS, or any function related thereto. For example, the encryption engine 234 encrypts and decrypts network packets, or any portion thereof, communicated via the appliance 200. The encryption engine 234 may also setup or establish SSL or TLS connections on behalf of the client 102a-102n, server 106a-106n, or appliance 200. As such, the encryption engine 234 provides offloading and acceleration of SSL processing. In one embodiment, the encryption engine 234 uses a tunneling protocol to provide a virtual private network between a client 102a-102n and a server 106a-106n. In some embodiments, the encryption engine 234 is in communication with the Encryption processor 260. In other embodiments, the encryption engine 234 comprises executable instructions running on the Encryption processor 260.

The multi-protocol compression engine 238 comprises any logic, business rules, function or operations for compressing one or more protocols of a network packet, such as any of the protocols used by the network stack 267 of the device 200. In one embodiment, multi-protocol compression engine 238 compresses bi-directionally between clients 102a-102n and servers 106a-106n any TCP/IP based protocol, including Messaging Application Programming Interface (MAPI) (email), File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), Common Internet File System (CIFS) protocol (file transfer), Independent Computing Architecture (ICA) protocol, Remote Desktop Protocol (RDP), Wireless Application Protocol (WAP), Mobile IP protocol, and Voice Over IP (VoIP) protocol. In other embodiments, multi-protocol compression engine 238 provides compression of Hypertext Markup Language (HTML) based protocols and in some embodiments, provides compression of any markup languages, such as the Extensible Markup Language (XML). In

one embodiment, the multi-protocol compression engine 238 provides compression of any high-performance protocol, such as any protocol designed for appliance 200 to appliance 200 communications. In another embodiment, the multi-protocol compression engine 238 compresses any payload of or any communication using a modified transport control
5 protocol, such as Transaction TCP (T/TCP), TCP with selective acknowledgements (TCP-SACK), TCP with large windows (TCP-LW), a congestion prediction protocol such as the TCP-Vegas protocol, and a TCP spoofing protocol.

As such, the multi-protocol compression engine 238 accelerates performance for users accessing applications via desktop clients, e.g., Microsoft Outlook and non-Web thin clients,
10 such as any client launched by popular enterprise applications like Oracle, SAP and Siebel, and even mobile clients, such as the Pocket PC. In some embodiments, the multi-protocol compression engine 238 by executing in the kernel mode 204 and integrating with packet processing engine 240 accessing the network stack 267 is able to compress any of the protocols carried by the TCP/IP protocol, such as any application layer protocol.

15 High speed layer 2-7 integrated packet engine 240, also generally referred to as a packet processing engine or packet engine, is responsible for managing the kernel-level processing of packets received and transmitted by appliance 200 via network ports 266. The high speed layer 2-7 integrated packet engine 240 may comprise a buffer for queuing one or more network packets during processing, such as for receipt of a network packet or
20 transmission of a network packet. Additionally, the high speed layer 2-7 integrated packet engine 240 is in communication with one or more network stacks 267 to send and receive network packets via network ports 266. The high speed layer 2-7 integrated packet engine 240 works in conjunction with encryption engine 234, cache manager 232, policy engine 236 and multi-protocol compression logic 238. In particular, encryption engine 234 is configured
25 to perform SSL processing of packets, policy engine 236 is configured to perform functions

related to traffic management such as request-level content switching and request-level cache redirection, and multi-protocol compression logic 238 is configured to perform functions related to compression and decompression of data.

The high speed layer 2-7 integrated packet engine 240 includes a packet processing timer 242. In one embodiment, the packet processing timer 242 provides one or more time intervals to trigger the processing of incoming, i.e., received, or outgoing, i.e., transmitted, network packets. In some embodiments, the high speed layer 2-7 integrated packet engine 240 processes network packets responsive to the timer 242. The packet processing timer 242 provides any type and form of signal to the packet engine 240 to notify, trigger, or

communicate a time related event, interval or occurrence. In many embodiments, the packet processing timer 242 operates in the order of milliseconds, such as for example 100ms, 50ms or 25ms. For example, in some embodiments, the packet processing timer 242 provides time intervals or otherwise causes a network packet to be processed by the high speed layer 2-7 integrated packet engine 240 at a 10 ms time interval, while in other embodiments, at a 5 ms time interval, and still yet in further embodiments, as short as a 3, 2, or 1 ms time interval.

The high speed layer 2-7 integrated packet engine 240 may be interfaced, integrated or in communication with the encryption engine 234, cache manager 232, policy engine 236 and multi-protocol compression engine 238 during operation. As such, any of the logic,

functions, or operations of the encryption engine 234, cache manager 232, policy engine 236 and multi-protocol compression logic 238 may be performed responsive to the packet processing timer 242 and/or the packet engine 240. Therefore, any of the logic, functions, or operations of the encryption engine 234, cache manager 232, policy engine 236 and multi-protocol compression logic 238 may be performed at the granularity of time intervals provided via the packet processing timer 242, for example, at a time interval of less than or equal to 10ms. For example, in one embodiment, the cache manager 232 may perform

invalidation of any cached objects responsive to the high speed layer 2-7 integrated packet engine 240 and/or the packet processing timer 242. In another embodiment, the expiry or invalidation time of a cached object can be set to the same order of granularity as the time interval of the packet processing timer 242, such as at every 10 ms.

5 In contrast to kernel space 204, user space 202 is the memory area or portion of the operating system used by user mode applications or programs otherwise running in user mode. A user mode application may not access kernel space 204 directly and uses service calls in order to access kernel services. As shown in FIG. 2, user space 202 of appliance 200 includes a graphical user interface (GUI) 210, a command line interface (CLI) 212, shell
10 services 214, health monitoring program 216, and daemon services 218. GUI 210 and CLI 212 provide a means by which a system administrator or other user can interact with and control the operation of appliance 200, such as via the operating system of the appliance 200 and either is user space 202 or kernel space 204. The GUI 210 may be any type and form of graphical user interface and may be presented via text, graphical or otherwise, by any type of
15 program or application, such as a browser. The CLI 212 may be any type and form of command line or text-based interface, such as a command line provided by the operating system. For example, the CLI 212 may comprise a shell, which is a tool to enable users to interact with the operating system. In some embodiments, the CLI 212 may be provided via a bash, csh, tcsh, or ksh type shell. The shell services 214 comprises the programs, services,
20 tasks, processes or executable instructions to support interaction with the appliance 200 or operating system by a user via the GUI 210 and/or CLI 212.

Health monitoring program 216 is used to monitor, check, report and ensure that network systems are functioning properly and that users are receiving requested content over a network. Health monitoring program 216 comprises one or more programs, services, tasks,
25 processes or executable instructions to provide logic, rules, functions or operations for

monitoring any activity of the appliance 200. In some embodiments, the health monitoring program 216 intercepts and inspects any network traffic passed via the appliance 200. In other embodiments, the health monitoring program 216 interfaces by any suitable means and/or mechanisms with one or more of the following: the encryption engine 234, cache manager 232, policy engine 236, multi-protocol compression logic 238, packet engine 240, daemon services 218, and shell services 214. As such, the health monitoring program 216 may call any application programming interface (API) to determine a state, status, or health of any portion of the appliance 200. For example, the health monitoring program 216 may ping or send a status inquiry on a periodic basis to check if a program, process, service or task is active and currently running. In another example, the health monitoring program 216 may check any status, error or history logs provided by any program, process, service or task to determine any condition, status or error with any portion of the appliance 200.

Daemon services 218 are programs that run continuously or in the background and handle periodic service requests received by appliance 200. In some embodiments, a daemon service may forward the requests to other programs or processes, such as another daemon service 218 as appropriate. As known to those skilled in the art, a daemon service 218 may run unattended to perform continuous or periodic system wide functions, such as network control, or to perform any desired task. In some embodiments, one or more daemon services 218 run in the user space 202, while in other embodiments, one or more daemon services 218 run in the kernel space.

Referring now to FIG. 2B, another embodiment of the appliance 200 is depicted. In brief overview, the appliance 200 provides one or more of the following services, functionality or operations: SSL VPN connectivity 280, switching/load balancing 284, Domain Name Service resolution 286, acceleration 288 and an application firewall 290 for communications between one or more clients 102 and one or more servers 106. In one

embodiment, the appliance 200 comprises any of the network devices manufactured by Citrix Systems, Inc. of Ft. Lauderdale Florida, referred to as Citrix NetScaler devices. Each of the servers 106 may provide one or more network related services 270a-270n (referred to as services 270). For example, a server 106 may provide an http service 270. The appliance
5 200 comprises one or more virtual servers or virtual internet protocol servers, referred to as a vServer, VIP server, or just VIP 275a-275n (also referred herein as vServer 275). The vServer 275 receives, intercepts or otherwise processes communications between a client 102 and a server 106 in accordance with the configuration and operations of the appliance 200.

The vServer 275 may comprise software, hardware or any combination of software
10 and hardware. The vServer 275 may comprise any type and form of program, service, task, process or executable instructions operating in user mode 202, kernel mode 204 or any combination thereof in the appliance 200. The vServer 275 includes any logic, functions, rules, or operations to perform any embodiments of the techniques described herein, such as SSL VPN 280, switching/load balancing 284, Domain Name Service resolution 286,
15 acceleration 288 and an application firewall 290. In some embodiments, the vServer 275 establishes a connection to a service 270 of a server 106. The service 275 may comprise any program, application, process, task or set of executable instructions capable of connecting to and communicating to the appliance 200, client 102 or vServer 275. For example, the service 275 may comprise a web server, http server, ftp, email or database server. In some
20 embodiments, the service 270 is a daemon process or network driver for listening, receiving and/or sending communications for an application, such as email, database or an enterprise application. In some embodiments, the service 270 may communicate on a specific IP address, or IP address and port.

In some embodiments, the vServer 275 applies one or more policies of the policy
25 engine 236 to network communications between the client 102 and server 106. In one

embodiment, the policies are associated with a VServer 275. In another embodiment, the policies are based on a user, or a group of users. In yet another embodiment, a policy is global and applies to one or more vServers 275a-275n, and any user or group of users communicating via the appliance 200. In some embodiments, the policies of the policy engine have conditions upon which the policy is applied based on any content of the communication, such as internet protocol address, port, protocol type, header or fields in a packet, or the context of the communication, such as user, group of the user, vServer 275, transport layer connection, and/or identification or attributes of the client 102 or server 106.

In other embodiments, the appliance 200 communicates or interfaces with the policy engine 236 to determine authentication and/or authorization of a remote user or a remote client 102 to access the computing environment 15, application, and/or data file from a server 106. In another embodiment, the appliance 200 communicates or interfaces with the policy engine 236 to determine authentication and/or authorization of a remote user or a remote client 102 to have the application delivery system 190 deliver one or more of the computing environment 15, application, and/or data file. In yet another embodiment, the appliance 200 establishes a VPN or SSL VPN connection based on the policy engine's 236 authentication and/or authorization of a remote user or a remote client 103. In one embodiment, the appliance 102 controls the flow of network traffic and communication sessions based on policies of the policy engine 236. For example, the appliance 200 may control the access to a computing environment 15, application or data file based on the policy engine 236.

In some embodiments, the vServer 275 establishes a transport layer connection, such as a TCP or UDP connection with a client 102 via the client agent 120. In one embodiment, the vServer 275 listens for and receives communications from the client 102. In other embodiments, the vServer 275 establishes a transport layer connection, such as a TCP or UDP connection with a client server 106. In one embodiment, the vServer 275 establishes

the transport layer connection to an internet protocol address and port of a server 270 running on the server 106. In another embodiment, the vServer 275 associates a first transport layer connection to a client 102 with a second transport layer connection to the server 106. In some embodiments, a vServer 275 establishes a pool of transport layer connections to a server 106 and multiplexes client requests via the pooled transport layer connections.

In some embodiments, the appliance 200 provides a SSL VPN connection 280 between a client 102 and a server 106. For example, a client 102 on a first network 102 requests to establish a connection to a server 106 on a second network 104'. In some embodiments, the second network 104' is not routable from the first network 104. In other embodiments, the client 102 is on a public network 104 and the server 106 is on a private network 104', such as a corporate network. In one embodiment, the client agent 120 intercepts communications of the client 102 on the first network 104, encrypts the communications, and transmits the communications via a first transport layer connection to the appliance 200. The appliance 200 associates the first transport layer connection on the first network 104 to a second transport layer connection to the server 106 on the second network 104. The appliance 200 receives the intercepted communication from the client agent 102, decrypts the communications, and transmits the communication to the server 106 on the second network 104 via the second transport layer connection. The second transport layer connection may be a pooled transport layer connection. As such, the appliance 200 provides an end-to-end secure transport layer connection for the client 102 between the two networks 104, 104'.

In one embodiment, the appliance 200 hosts an intranet internet protocol or intranetIP 282 address of the client 102 on the virtual private network 104. The client 102 has a local network identifier, such as an internet protocol (IP) address and/or host name on the first network 104. When connected to the second network 104' via the appliance 200, the

appliance 200 establishes, assigns or otherwise provides an IntranetIP, which is network identifier, such as IP address and/or host name, for the client 102 on the second network 104'.

The appliance 200 listens for and receives on the second or private network 104' for any communications directed towards the client 102 using the client's established IntranetIP 282.

5 In one embodiment, the appliance 200 acts as or on behalf of the client 102 on the second private network 104. For example, in another embodiment, a vServer 275 listens for and responds to communications to the IntranetIP 282 of the client 102. In some embodiments, if a computing device 100 on the second network 104' transmits a request, the appliance 200 processes the request as if it were the client 102. For example, the appliance 200 may
10 respond to a ping to the client's IntranetIP 282. In another example, the appliance may establish a connection, such as a TCP or UDP connection, with computing device 100 on the second network 104 requesting a connection with the client's IntranetIP 282.

In some embodiments, the appliance 200 provides one or more of the following acceleration techniques 288 to communications between the client 102 and server 106: 1)
15 compression; 2) decompression; 3) Transmission Control Protocol pooling; 4) Transmission Control Protocol multiplexing; 5) Transmission Control Protocol buffering; and 6) caching. In one embodiment, the appliance 200 relieves servers 106 of much of the processing load caused by repeatedly opening and closing transport layers connections to clients 102 by opening one or more transport layer connections with each server 106 and maintaining these
20 connections to allow repeated data accesses by clients via the Internet. This technique is referred to herein as "connection pooling".

In some embodiments, in order to seamlessly splice communications from a client 102 to a server 106 via a pooled transport layer connection, the appliance 200 translates or multiplexes communications by modifying sequence number and acknowledgment numbers
25 at the transport layer protocol level. This is referred to as "connection multiplexing". In some

embodiments, no application layer protocol interaction is required. For example, in the case of an in-bound packet (that is, a packet received from a client 102), the source network address of the packet is changed to that of an output port of appliance 200, and the destination network address is changed to that of the intended server. In the case of an outbound packet (that is, one received from a server 106), the source network address is changed from that of the server 106 to that of an output port of appliance 200 and the destination address is changed from that of appliance 200 to that of the requesting client 102. The sequence numbers and acknowledgment numbers of the packet are also translated to sequence numbers and acknowledgement expected by the client 102 on the appliance's 200 transport layer connection to the client 102. In some embodiments, the packet checksum of the transport layer protocol is recalculated to account for these translations.

In another embodiment, the appliance 200 provides switching or load-balancing functionality 284 for communications between the client 102 and server 106. In some embodiments, the appliance 200 distributes traffic and directs client requests to a server 106 based on layer 4 or application-layer request data. In one embodiment, although the network layer or layer 2 of the network packet identifies a destination server 106, the appliance 200 determines the server 106 to distribute the network packet by application information and data carried as payload of the transport layer packet. In one embodiment, the health monitoring programs 216 of the appliance 200 monitor the health of servers to determine the server 106 for which to distribute a client's request. In some embodiments, if the appliance 200 detects a server 106 is not available or has a load over a predetermined threshold, the appliance 200 can direct or distribute client requests to another server 106.

In some embodiments, the appliance 200 acts as a Domain Name Service (DNS) resolver or otherwise provides resolution of a DNS request from clients 102. In some embodiments, the appliance intercepts' a DNS request transmitted by the client 102. In one

embodiment, the appliance 200 responds to a client's DNS request with an IP address of or hosted by the appliance 200. In this embodiment, the client 102 transmits network communication for the domain name to the appliance 200. In another embodiment, the appliance 200 responds to a client's DNS request with an IP address of or hosted by a second appliance 200'. In some embodiments, the appliance 200 responds to a client's DNS request with an IP address of a server 106 determined by the appliance 200.

In yet another embodiment, the appliance 200 provides application firewall functionality 290 for communications between the client 102 and server 106. In one embodiment, the policy engine 236 provides rules for detecting and blocking illegitimate requests. In some embodiments, the application firewall 290 protects against denial of service (DoS) attacks. In other embodiments, the appliance inspects the content of intercepted requests to identify and block application-based attacks. In some embodiments, the rules/policy engine 236 comprises one or more application firewall or security control policies for providing protections against various classes and types of web or Internet based vulnerabilities, such as one or more of the following: 1) buffer overflow, 2) CGI-BIN parameter manipulation, 3) form/hidden field manipulation, 4) forceful browsing, 5) cookie or session poisoning, 6) broken access control list (ACLs) or weak passwords, 7) cross-site scripting (XSS), 8) command injection, 9) SQL injection, 10) error triggering sensitive information leak, 11) insecure use of cryptography, 12) server misconfiguration, 13) back doors and debug options, 14) website defacement, 15) platform or operating systems vulnerabilities, and 16) zero-day exploits. In an embodiment, the application firewall 290 provides HTML form field protection in the form of inspecting or analyzing the network communication for one or more of the following: 1) required fields are returned, 2) no added field allowed, 3) read-only and hidden field enforcement, 4) drop-down list and radio button field conformance, and 5) form-field max-length enforcement. In some embodiments, the

application firewall 290 ensures cookies are not modified. In other embodiments, the application firewall 290 protects against forceful browsing by enforcing legal URLs.

In still yet other embodiments, the application firewall 290 protects any confidential information contained in the network communication. The application firewall 290 may inspect or analyze any network communication in accordance with the rules or policies of the engine 236 to identify any confidential information in any field of the network packet. In some embodiments, the application firewall 290 identifies in the network communication one or more occurrences of a credit card number, password, social security number, name, patient code, contact information, and age. The encoded portion of the network communication may comprise these occurrences or the confidential information. Based on these occurrences, in one embodiment, the application firewall 290 may take a policy action on the network communication, such as prevent transmission of the network communication. In another embodiment, the application firewall 290 may rewrite, remove or otherwise mask such identified occurrence or confidential information.

C. Client Agent

Referring now to FIG. 3, an embodiment of the client agent 120 is depicted. The client 102 includes a client agent 120 for establishing and exchanging communications with the appliance 200 and/or server 106 via a network 104. In brief overview, the client 102 operates on computing device 100 having an operating system with a kernel mode 302 and a user mode 303, and a network stack 310 with one or more layers 310a-310b. The client 102 may have installed and/or execute one or more applications. In some embodiments, one or more applications may communicate via the network stack 310 to a network 104. One of the applications, such as a web browser, may also include a first program 322. For example, the

first program 322 may be used in some embodiments to install and/or execute the client agent 120, or any portion thereof. The client agent 120 includes an interception mechanism, or interceptor 350, for intercepting network communications from the network stack 310 from the one or more applications.

5 The network stack 310 of the client 102 may comprise any type and form of software, or hardware, or any combinations thereof, for providing connectivity to and communications with a network. In one embodiment, the network stack 310 comprises a software implementation for a network protocol suite. The network stack 310 may comprise one or more network layers, such as any networks layers of the Open Systems Interconnection (OSI) communications model as those skilled in the art recognize and appreciate. As such, the network stack 310 may comprise any type and form of protocols for any of the following layers of the OSI model: 1) physical link layer, 2) data link layer, 3) network layer, 4) transport layer, 5) session layer, 6) presentation layer, and 7) application layer. In one embodiment, the network stack 310 may comprise a transport control protocol (TCP) over the network layer protocol of the internet protocol (IP), generally referred to as TCP/IP. In some embodiments, the TCP/IP protocol may be carried over the Ethernet protocol, which may comprise any of the family of IEEE wide-area-network (WAN) or local-area-network (LAN) protocols, such as those protocols covered by the IEEE 802.3. In some embodiments, the network stack 310 comprises any type and form of a wireless protocol, such as IEEE 802.11 and/or mobile internet protocol.

 In view of a TCP/IP based network, any TCP/IP based protocol may be used, including Messaging Application Programming Interface (MAPI) (email), File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), Common Internet File System (CIFS) protocol (file transfer), Independent Computing Architecture (ICA) protocol, Remote Desktop Protocol (RDP), Wireless Application Protocol (WAP), Mobile IP protocol, and

Voice Over IP (VoIP) protocol. In another embodiment, the network stack 310 comprises any type and form of transport control protocol, such as a modified transport control protocol, for example a Transaction TCP (T/TCP), TCP with selection acknowledgements (TCP-SACK), TCP with large windows (TCP-LW), a congestion prediction protocol such as the

5 TCP-Vegas protocol, and a TCP spoofing protocol. In other embodiments, any type and form of user datagram protocol (UDP), such as UDP over IP, may be used by the network stack 310, such as for voice communications or real-time data communications.

Furthermore, the network stack 310 may include one or more network drivers supporting the one or more layers, such as a TCP driver or a network layer driver. The

10 network drivers may be included as part of the operating system of the computing device 100 or as part of any network interface cards or other network access components of the computing device 100. In some embodiments, any of the network drivers of the network stack 310 may be customized, modified or adapted to provide a custom or modified portion of the network stack 310 in support of any of the techniques described herein. In other

15 embodiments, the acceleration program 120 is designed and constructed to operate with or work in conjunction with the network stack 310 installed or otherwise provided by the operating system of the client 102.

The network stack 310 comprises any type and form of interfaces for receiving, obtaining, providing or otherwise accessing any information and data related to network

20 communications of the client 102. In one embodiment, an interface to the network stack 310 comprises an application programming interface (API). The interface may also comprise any function call, hooking or filtering mechanism, event or call back mechanism, or any type of interfacing technique. The network stack 310 via the interface may receive or provide any type and form of data structure, such as an object, related to functionality or operation of the

25 network stack 310. For example, the data structure may comprise information and data

related to a network packet or one or more network packets. In some embodiments, the data structure comprises a portion of the network packet processed at a protocol layer of the network stack 310, such as a network packet of the transport layer. In some embodiments, the data structure 325 comprises a kernel-level data structure, while in other embodiments, the data structure 325 comprises a user-mode data structure. A kernel-level data structure may comprise a data structure obtained or related to a portion of the network stack 310 operating in kernel-mode 302, or a network driver or other software running in kernel-mode 302, or any data structure obtained or received by a service, process, task, thread or other executable instructions running or operating in kernel-mode of the operating system.

Additionally, some portions of the network stack 310 may execute or operate in kernel-mode 302, for example, the data link or network layer, while other portions execute or operate in user-mode 303, such as an application layer of the network stack 310. For example, a first portion 310a of the network stack may provide user-mode access to the network stack 310 to an application while a second portion 310b of the network stack 310 provides access to a network. In some embodiments, a first portion 310a of the network stack may comprise one or more upper layers of the network stack 310, such as any of layers 5-7. In other embodiments, a second portion 310b of the network stack 310 comprises one or more lower layers, such as any of layers 1-4. Each of the first portion 310a and second portion 310b of the network stack 310 may comprise any portion of the network stack 310, at any one or more network layers, in user-mode 203, kernel-mode, 202, or combinations thereof, or at any portion of a network layer or interface point to a network layer or any portion of or interface point to the user-mode 203 and kernel-mode 203. .

The interceptor 350 may comprise software, hardware, or any combination of software and hardware. In one embodiment, the interceptor 350 intercept a network communication at any point in the network stack 310, and redirects or transmits the network

communication to a destination desired, managed or controlled by the interceptor 350 or client agent 120. For example, the interceptor 350 may intercept a network communication of a network stack 310 of a first network and transmit the network communication to the appliance 200 for transmission on a second network 104. In some embodiments, the

5 interceptor 350 comprises any type of interceptor 350 comprising a driver, such as a network driver constructed and designed to interface and work with the network stack 310. In some embodiments, the client agent 120 and/or interceptor 350 operates at one or more layers of the network stack 310, such as at the transport layer. In one embodiment, the interceptor 350 comprises a filter driver, hooking mechanism, or any form and type of suitable network
10 driver interface that interfaces to the transport layer of the network stack, such as via the transport driver interface (TDI). In some embodiments, the interceptor 350 interfaces to a first protocol layer, such as the transport layer and another protocol layer, such as any layer above the transport protocol layer, for example, an application protocol layer. In one embodiment, the interceptor 350 may comprise a driver complying with the Network Driver
15 Interface Specification (NDIS), or a NDIS driver. In another embodiment, the interceptor 350 may comprise a min-filter or a mini-port driver. In one embodiment, the interceptor 350, or portion thereof, operates in kernel-mode 202. In another embodiment, the interceptor 350, or portion thereof, operates in user-mode 203. In some embodiments, a portion of the interceptor 350 operates in kernel-mode 202 while another portion of the interceptor 350
20 operates in user-mode 203. In other embodiments, the client agent 120 operates in user-mode 203 but interfaces via the interceptor 350 to a kernel-mode driver, process, service, task or portion of the operating system, such as to obtain a kernel-level data structure 225. In further embodiments, the interceptor 350 is a user-mode application or program, such as application.

In one embodiment, the interceptor 350 intercepts any transport layer connection
25 requests. In these embodiments, the interceptor 350 executes transport layer application

programming interface (API) calls to set the destination information, such as destination IP address and/or port to a desired location for the location. In this manner, the interceptor 350 intercepts and redirects the transport layer connection to a IP address and port controlled or managed by the interceptor 350 or client agent 120. In one embodiment, the interceptor 350 sets the destination information for the connection to a local IP address and port of the client 102 on which the client agent 120 is listening. For example, the client agent 120 may comprise a proxy service listening on a local IP address and port for redirected transport layer communications. In some embodiments, the client agent 120 then communicates the redirected transport layer communication to the appliance 200.

In some embodiments, the interceptor 350 intercepts a Domain Name Service (DNS) request. In one embodiment, the client agent 120 and/or interceptor 350 resolves the DNS request. In another embodiment, the interceptor transmits the intercepted DNS request to the appliance 200 for DNS resolution. In one embodiment, the appliance 200 resolves the DNS request and communicates the DNS response to the client agent 120. In some embodiments, the appliance 200 resolves the DNS request via another appliance 200' or a DNS server 106.

In yet another embodiment, the client agent 120 may comprise two agents 120 and 120'. In one embodiment, a first agent 120 may comprise an interceptor 350 operating at the network layer of the network stack 310. In some embodiments, the first agent 120 intercepts network layer requests such as Internet Control Message Protocol (ICMP) requests (e.g., ping and traceroute). In other embodiments, the second agent 120' may operate at the transport layer and intercept transport layer communications. In some embodiments, the first agent 120 intercepts communications at one layer of the network stack 210 and interfaces with or communicates the intercepted communication to the second agent 120'.

The client agent 120 and/or interceptor 350 may operate at or interface with a protocol layer in a manner transparent to any other protocol layer of the network stack 310. For

example, in one embodiment, the interceptor 350 operates or interfaces with the transport layer of the network stack 310 transparently to any protocol layer below the transport layer, such as the network layer, and any protocol layer above the transport layer, such as the session, presentation or application layer protocols. This allows the other protocol layers of the network stack 310 to operate as desired and without modification for using the interceptor 350. As such, the client agent 120 and/or interceptor 350 can interface with the transport layer to secure, optimize, accelerate, route or load-balance any communications provided via any protocol carried by the transport layer, such as any application layer protocol over TCP/IP.

Furthermore, the client agent 120 and/or interceptor may operate at or interface with the network stack 310 in a manner transparent to any application, a user of the client 102, and any other computing device, such as a server, in communications with the client 102. The client agent 120 and/or interceptor 350 may be installed and/or executed on the client 102 in a manner without modification of an application. In some embodiments, the user of the client 102 or a computing device in communications with the client 102 are not aware of the existence, execution or operation of the client agent 120 and/or interceptor 350. As such, in some embodiments, the client agent 120 and/or interceptor 350 is installed, executed, and/or operated transparently to an application, user of the client 102, another computing device, such as a server, or any of the protocol layers above and/or below the protocol layer interfaced to by the interceptor 350.

The client agent 120 includes an acceleration program 302, a streaming client 306, and/or a collection agent 304. In one embodiment, the client agent 120 comprises an Independent Computing Architecture (ICA) client, or any portion thereof, developed by Citrix Systems, Inc. of Fort Lauderdale, Florida, and is also referred to as an ICA client. In some embodiments, the client 120 comprises an application streaming client 306 for

streaming an application from a server 106 to a client 102. In some embodiments, the client agent 120 comprises an acceleration program 302 for accelerating communications between client 102 and server 106. In another embodiment, the client agent 120 includes a collection agent 304 for performing end-point detection/scanning and collecting end-point information
5 for the appliance 200 and/or server 106.

In some embodiments, the acceleration program 302 comprises a client-side acceleration program for performing one or more acceleration techniques to accelerate, enhance or otherwise improve a client's communications with and/or access to a server 106, such as accessing an application provided by a server 106. The logic, functions, and/or
10 operations of the executable instructions of the acceleration program 302 may perform one or more of the following acceleration techniques: 1) multi-protocol compression, 2) transport control protocol pooling, 3) transport control protocol multiplexing, 4) transport control protocol buffering, and 5) caching via a cache manager. Additionally, the acceleration program 302 may perform encryption and/or decryption of any communications received
15 and/or transmitted by the client 102. In some embodiments, the acceleration program 302 performs one or more of the acceleration techniques in an integrated manner or fashion. Additionally, the acceleration program 302 can perform compression on any of the protocols, or multiple-protocols, carried as a payload of a network packet of the transport layer protocol.

The streaming client 306 comprises an application, program, process, service, task or
20 executable instructions for receiving and executing a streamed application from a server 106. A server 106 may stream one or more application data files to the streaming client 306 for playing, executing or otherwise causing to be executed the application on the client 102. In some embodiments, the server 106 transmits a set of compressed or packaged application data files to the streaming client 306. In some embodiments, the plurality of application files
25 are compressed and stored on a file server within an archive file such as a CAB, ZIP, SIT,

TAR, JAR or other archive. In one embodiment, the server 106 decompresses, unpackages or unarchives the application files and transmits the files to the client 102. In another embodiment, the client 102 decompresses, unpackages or unarchives the application files.

The streaming client 306 dynamically installs the application, or portion thereof, and executes the application. In one embodiment, the streaming client 306 may be an executable program. In some embodiments, the streaming client 306 may be able to launch another executable program.

The collection agent 304 comprises an application, program, process, service, task or executable instructions for identifying, obtaining and/or collecting information about the client 102. In some embodiments, the appliance 200 transmits the collection agent 304 to the client 102 or client agent 120. The collection agent 304 may be configured according to one or more policies of the policy engine 236 of the appliance. In other embodiments, the collection agent 304 transmits collected information on the client 102 to the appliance 200. In one embodiment, the policy engine 236 of the appliance 200 uses the collected information to determine and provide access, authentication and authorization control of the client's connection to a network 104.

In one embodiment, the collection agent 304 comprises an end-point detection and scanning mechanism, which identifies and determines one or more attributes or characteristics of the client. For example, the collection agent 304 may identify and determine any one or more of the following client-side attributes: 1) the operating system an/or a version of an operating system, 2) a service pack of the operating system, 3) a running service, 4) a running process, and 5) a file. The collection agent 304 may also identify and determine the presence or versions of any one or more of the following on the client: 1) antivirus software, 2) personal firewall software, 3) anti-spam software, and 4) internet

security software. The policy engine 236 may have one or more policies based on any one or more of the attributes or characteristics of the client or client-side attributes.

In some embodiments and still referring to FIG. 3, a first program 322 may be used to install and/or execute the client agent 120, or portion thereof, such as the interceptor 350, automatically, silently, transparently, or otherwise. In one embodiment, the first program 322 comprises a plugin component, such an ActiveX control or Java control or script that is loaded into and executed by an application. For example, the first program comprises an ActiveX control loaded and run by a web browser application, such as in the memory space or context of the application. In another embodiment, the first program 322 comprises a set of executable instructions loaded into and run by the application, such as a browser. In one embodiment, the first program 322 comprises a designed and constructed program to install the client agent 120. In some embodiments, the first program 322 obtains, downloads, or receives the client agent 120 via the network from another computing device. In another embodiment, the first program 322 is an installer program or a plug and play manager for installing programs, such as network drivers, on the operating system of the client 102.

D. POLICY-BASED TRIGGERING OF CLIENT AUTHENTICATION

Referring now to FIG. 4, an embodiment of an appliance 200 for providing policy based triggering of client authentication based on any portion or content of a client request is depicted. In brief overview, a client 102 establishes a connection with an appliance 200, via client agent 120. The client 102 transmits a request 405 received or intercepted by the appliance 200. The request 405 may comprises a request for a resource. For example, the client request 405 may comprise an HTTP request to access a Uniform Resource Locator (URL) of a directory, such as a protected directory, on a web server 106, such as a web server provided via service 270. The appliance 405 may associate any portion or element of the

request with a client authentication policy 420. Based on the client authentication policy 420, the policy engine 236 may request 430 from the client a client certificate, or authentication certificate. Upon receiving the request and before authentication the client 102, the appliance 200 may prevent access to the network 104, server 106 or resource. Based on the response 5 440 from the client 102, the policy engine 236 may deny or grant a level of access to the resource.

The client request 405 may comprises any type and form of content or information. In one embodiment, the client request comprises a URL 450, including any parts of a URL, such a suffix, prefix, token, or query string. The URL may identify any type of resource, 10 such as a directory, file, web-page, location, network address, database, program, service or server. In one embodiment, the URL 450 identifies a directory 452 on a server 106. In some embodiments, the URL 450 identifies a protected directory 452. In other embodiments, the URL 450 identifies an unprotected directory 452. The client request 405 may comprise any type and form of protocol, such as HyperText Transfer Protocol (HTTP), secure HyperText 15 Transfer Protocol (HTTPS), or Extended Markup Language (XML). In some embodiments, the client request 405 identifies a type or name of a function, method or operation 455. For example, the client request 405 may identify an HTTP Get or Post method 455. In other embodiments, the client request 405 comprises a set of one or more name-value pairs.

In other embodiments, the client request 405 identifies any network or IP layer 20 information. For example, the client request 405 may be transmitted via a transport layer connection, such as via TCP/IP. With a TCP/IP packet, the appliance 200 can determine any network layer information, such as network identifiers (IP addresses and/or host names) and ports. In some embodiments, the client request 405 comprises a client IP address and/or host name 460. In other embodiments, the client request 405 comprises a destination IP address 25 and/or host name 465. In another embodiment, the client request 405 comprises information

identifying any of the network ports 470 for the client 102 and/or server 106. For example, the client request 405 may identify the source port 470 of transmission from the client 102. The client request 405 may identify the destination port 470 the request is destined for, such as a network port 470 of a server 106 or service 270. In some embodiments, the client
5 request 405 includes, identifies or is associated with one or more SSL or TLS parameters 475. In one embodiment, the client request 405 includes, identifies or is associated with and SSL or TLS version 475. In another embodiment, the client request 405 includes, identifies or is associated with a cipher 475. In some embodiments, the client request 405 includes, identifies or is associated with a cipher strength 475, which may be specified in bits.

10 The client authentication policy 420 may be associated with, specify or identify any portion or element of the client request 405. For example, if the client request 405 has a portion or element identified or used by the policy 420, then the appliance 200 via the policy engine 236 takes an action on the request 405 based on the policy 420. In some
15 embodiments, the elements of the client request 405 can be used as the specification, conditions or parameters of the client authentication policy 420. In one embodiment, the client authentication policy 420 is associated with a URL pattern 450, such as any suffix, prefix, token or query string. In another embodiment, the client authentication policy 420 is associated with a directory 452, such as a protected directory. In other embodiments, the client authentication policy 420 is associated with a function or method 455, such as an
20 HTTP Get or Post. In one embodiment, the client authentication policy 420 is associated with a client IP address or host name 460. In another embodiment, the client authentication policy 420 is associated with a destination IP address or host name 465. In some embodiments, the client authentication policy 420 is associated with one or more network ports 470, such as a source port and/or destination port. In yet another embodiment, the

client authentication policy 420 is associated with any one or more SSL or TLS parameters
475.

In some embodiments, the client authentication policy 420 may identify any one or more actions for the policy engine 236 to take with regards to authentication of the client 102
5 and/or authorization to access the resource by the client 102, or otherwise to process and transmit the client request 405. In one embodiment, the client authentication policy 420 may identify or specify to authenticate the client 102 before further processing or transmitting the request by the appliance 200. In some embodiments, the client authentication policy 420 identifies that an authentication certificate, such as an SSL or TLS certificate, should be
10 requested from the client 102. In one embodiment, the authentication certificate comprises a client certificate. In another embodiment, the policy 420 indicates the certificate is required in order to transmit the client request 405. In other embodiments, the policy 420 indicates the certificate should be requested but is optional. In yet another embodiment, the client authentication policy 420 indicates authentication credentials of the user are required. In
15 some embodiments, the client authentication policy 420 indicates that user authentication credentials should be requested but are optional

In other embodiments, the client authentication policy 420 indicates to only request the client certificate once and for other requests for the client 102 to not request the certificate. In some embodiments, the client authentication policy 420 indicates to request a
20 certificate for every client request 405. In another embodiment, the client authentication policy 420 indicates to request the client certificate for any new resource requested by the client 102. In other embodiments, the client authentication policy 420 indicates to request a certificate only on the first request for a resource, and for subsequent requests, a certificate may not be requested.

In yet other embodiments, the client authentication policy 420 may request authentication of the client 102, for example, via a client certificate, based on any of the other portions of the client request 405. In one embodiment, the client authentication policy 420 may request authentication of the client 102 if the client request 405 has a URL pattern 450 that matches a URL pattern specified by the policy 420. In another embodiment, the client authentication policy 420 may request authentication of the client 102 if the client request 405 identifies a directory 452 that matches a directory specified by the policy 420. In some embodiments, the client authentication policy 420 may request authentication of the client 102 if the client request 405 identifies a method or function 455 that matches a method or function specified by the policy 420. In other embodiments, the client authentication policy 420 may request authentication of the client 102 if the client request 405 identifies a client IP and/or destination IP 465 that matches IP addresses or host names specified by the policy 420. In one embodiment, the client authentication policy 420 may request authentication of the client 102 if the client request 405 identifies a network port 270 that matches a network port specified by the policy 420. In another embodiment, the client authentication policy 420 may request authentication of the client 102 if the client request 405 is associated with a SSL parameters 475 that matches SSL parameters specified by the policy 420.

In yet another embodiment, the client authentication policy 420 may request authentication of the client 102 if a portion 450, 452, 455, 460, 465, 470 and/or 475 of the client request 405 does not match the corresponding specification of the policy 420. For example, if the URL pattern 450 does not have a certain prefix or suffix, then the client 102 should be authenticated. In another example, if the client IP address 460 of the request 405 does not match a subnet mask, then the client 102 should be authenticated.

In some embodiments, the client authentication policy 420 may be associated with one or more authorization policies or a level of access of the client 102 or user of the client

102. In other embodiments, the client authentication policy 420 indicates an action to take based on authentication requested by the policy. In one embodiment, the client authentication policy 420 indicates that the client 102 is not to be allowed or granted a level of access based on a result or verification of the authentication via certificate, or otherwise.

5 For example, if the client 102 provides a valid certificate in response 440 to a request 430 by the appliance 200, the appliance 200 transmits the request to the intended destination, such as server 106, according to the policy 420. In another embodiment, the client authentication policy 420 indicates that the client 102 should be denied access or a level of access. For example, if the client 102 transmits an invalid certificate in response 440 to the certificate
10 request 430, the appliance 200 may deny access by the client 102 to the resource. In some cases, the appliance 200 may drop the request 405.

In some embodiments, the client authentication policy 420 may identify or specify an action to insert any data, e.g. data insertion 480 as illustrated in FIG. 4, into the client request 405, or communications by the appliance 200 to the client 102, or communications by the
15 appliance 200 to the server 106 when transmitting the client request 405. In one embodiment, an action of the client authentication policy 420 may identify or specify to add an SSL or TLS data to the communication. In some embodiments, the action may identify to add any SSL or TLS related session parameters to any header or field in the communication, such as a certificate header, certificate subject, certificate hash, certificate issuer, session
20 header, and/or cipher header. In other embodiments, the action may identify the appliance 200 to transmit an SSL certificate request to the client 102 having specified SSL parameters.

In still other embodiments, the appliance 200, such as via policy engine 236, may insert the client's response to the certificate request into the client request (data insertion 480) transmitted to the server 106. In one embodiment, when the authentication policy 420
25 specifies authentication is optional, the appliance 200 may allow the client request 405 to be

transmitted to the server 106 even if the client 102 does not provide a certificate or provides a bad or expired certificate. However, in some embodiments, the appliance 200 inserts the result of the authentication, i.e., response from the client, as an HTTP header in the client request transmitted to the server 106. In this manner, the server 106 can take corrective
5 action, or decide not take corrective action, based on the client's response to authentication. This optional authentication mechanism can be useful to provide variable access control to the end user. If a user or client 102 properly authenticates by responding with a valid certification, the user or client may be provided with full access to resources via the appliance 200 and/or server 106. Otherwise, if the user or client does not provide a valid certificate, the
10 user or client's access may be restricted to certain resources only.

In operation, the appliance 200 uses any of the portions or any granularity of information of the client request 405 to provide authentication policies regarding access by the client 102 to any resources via the appliance 200. Referring now to FIG. 5, steps of an embodiment of a method 500 of operations of the appliance 200 in applying policy based
15 triggering of client authentication for a client request is depicted. In brief overview, the appliance 200 receives a request 405 from a client 102 to access a resource. At step 510, the appliance determines from any portion of the request 405 an identification of a client authentication policy 420. For example, the appliance 200 determines an element of the client request matches a corresponding specification of the client authentication policy 420.
20 At step 515, the appliance 200 requests 430 from the client 102 a certificate for the client to access the resource based on the identified policy 420. At step 520, the appliance 200, in one embodiment, prevents access to the resource until receiving a response 440 from the client 102 regarding the client certificate. At step 525, the appliance 300 receives from the client 102 a response 440 to the certificate request 430. At step 530, the appliance 200 validates the
25 response 440 in accordance with the policy. At step 535, based on validation of the client's

response 400 and the policy 420, the appliance 200 grants or denies a level of access to the resource requested via the client request 405.

In further detail, at step 505, the appliance 200 receives or intercepts the client request 405 from the client 102 or client agent 120. In one embodiment, the appliance 200 intercepts the client's transmissions of the request to a server 106. In other embodiments, the client agent 120 intercepts the client's request from the network stack of the client 102. The client agent 120 then transmits the client request to the appliance 200. In some embodiments, the client 102 via the client agent 120 establishes a transport layer connection with the appliance 200. In one embodiment, the transport layer connection comprises a SSL VPN connection to a network 104. In another embodiment, the transport layer connection comprises a SSL or TLS session. In some embodiments, the appliance 200 hosts the IP address of a server 106 requested by the client 102, such as via a DNS request. For example, the appliance 200 acts a virtual IP for a server 106 accessed by the client 102. In this manner, the client 102 or client agent 120 transmits the client's request to the appliance 200. As such, the appliance 200 receives the client request from the client 102.

At step 510, the appliance 200 identifies or selects a client authentication policy 420 based on the appliance 200 determining a portion of the request is associated with the policy 420. In some embodiments, the appliance 200 determines any of the following matches a corresponding specification or description of a policy 420: URL pattern 450, directory 452, method/function 455, client IP 460, destination IP 465, network ports 470 and/or SSL parameters 475. In other embodiments, the appliance 200 determines a combination of the portions 450, 452, 455, 460, 465, 470 and/or 475 correspond to or match a specification of a policy 420. For example, the appliance 200 may determine if the client request 405 has a URL pattern 450 and a client IP 460 specified by the policy 420. In yet other embodiments,

the appliance 200 determines that one or more of the portions 450, 452, 455, 460, 465, 470 and 475 do not match a specification or description of the policy 420.

Based on identifying the client authentication policy 420 associated with the client request 405, at step 515, the appliance 200 takes the authentication action indicated by the

5 policy 420. In some embodiments, the policy 420 directs the appliance 200 to request a certificate from the client 102. In one embodiment, the policy 420 indicates the certificate is mandatory; while in other embodiments, the policy 420 indicates the certificate is optional.

In some embodiment, the appliance transmits a request 430 to the client 102 for a certificate from the client 102. In some of these embodiments, the appliance 200 uses SSL-based

10 transactions or SSL negotiations to request 430 the certificate from the client 102. As such, in one embodiment, the request 430 comprises a SSL request, transaction or part of a SSL

handshake for a certificate. In another embodiment, the appliance 200 requests a new SSL session with the client 102. In other embodiments, the appliance 200 may use any type and

form of protocol to request a certificate from the client 102. In some embodiments, the

15 appliance 200 requests authentication credentials from a user of the client 102. In yet another embodiment, the appliance 200 does not authenticate the client 102 based on a policy 420 if the client 102 has been previously authenticated, for example, on a previous request or previous triggering of a client authentication policy 420.

At step 520, in some embodiments, the appliance 200 prevents access by the client

20 102 until validating the type of authentication performed by the appliance 200 on the client

102 in accordance with the policy 420. In one embodiment, upon receipt of the client request

405, the appliance 200 buffers, queues, or otherwise holds the client request 405. As such, in some embodiments, the appliance 200 does not transmit the client request 405 beyond the

appliance, for example, onto the virtual private network 104. In one embodiment, the

25 appliance 200 provides a first end-point to one transport layer connection to the client 102

and a second end-point to a transport layer connection to a server 106. In some cases, the appliance 200 provides a plurality of pooled second transport layer connections to one or more servers 106. In another embodiment, the appliance 200 receives the client request 405 on the first transport layer connection and does not transmit the client request 405 via a second transport layer. In one case, the client request 405 comprises a request to open a transport layer connection. In one embodiment, at step 520, the appliance 200 does not yet open the transport layer connection. In another embodiment, the appliance 200 does not associate or link the first transport layer connection of the client 102 with one of a pooled set of more second transport layer connections to a server 106. By preventing the client request 405 from being transmitted beyond the appliance 200 or onto a network before the client 102 has been authenticated according to policy, the appliance 200 provides a more secure gateway access point to the network 104.

At step 525, in response to the appliance's request 430 for a certificate, the client 102 provides a response 440 to the request. In some embodiments, the client 102 or client agent 120 transmits a client certificate to the appliance 200. In other embodiments, the client 102 or client agent 120 transmits a message to the appliance 200 indicating that the client 102 does not have a client certificate. In another embodiment, the client 102 or client agent 120 transmits a message to the appliance 200 indicating there was an error in finding, obtaining or otherwise providing the requested certificate. In some embodiments, the appliance 200 receives the certificate or message about the certificate via a response 440 as part of an SSL handshake or SSL negotiation transaction. In other embodiments, the appliance 200 receives the certificate or message about the certificate via any type and form of protocol. In another embodiment, the appliance 200 sets a time for a predetermined time period for waiting to receive the response 440 from the client 102. As such, in these cases, the appliance 200 may identify a timeout as the response from the client 102.

At step 530, the appliance 200 validates the response 440 from the client 102. In one embodiment, the appliance 200 checks if the certificate is from a valid or expected issuer or otherwise has valid or expected content. In another embodiment, the appliance 200 validates the certificate with any authentication credentials or user profiles. In some embodiments, the appliance 200 uses a third-party service, host or server to validate the certificate. In one embodiment, the appliance 200 acts a certificate authority or issuer for client certificates, and validates the received certificate against a database. In some embodiments, the appliance 200 determines if the certificate is otherwise intact, complete or without error. In yet another embodiment, the appliance 200 determines or validates the completion of a SSL or TLS handshake. In some embodiments, the appliance 200 validates a new SSL or TLS session has been established with the client or otherwise the new session handshake has been completed.

In some embodiments, the appliance 200 receives as a response authentication credentials of a user of the client 102 and validates the credentials. In one embodiment, the appliance 200 validates the authentication credentials with a database, a server, LDAP directory service or any type and form of authentication service accessible via the network 104. In some embodiments, the appliance 200 determines the authentication credentials are incomplete, invalid or otherwise do not authenticate the user.

.In one embodiment, the appliance 200 determines the certificate is invalid. In other embodiments, the appliance 200 determines the certificate is corrupt, incomplete or has an error. In another embodiment, the appliance 200 determines the certificate is not associated with or belongs to the client 102. In other embodiments, the appliance 200 determines the client 102 did not provide a certificate in response 440 to the request 430. In one embodiment, the appliance 200 determines the response indicates the client 102 does not have a certificate, cannot transmit a certificate or otherwise is not providing a certificate. In some embodiments, the appliance 200 determines the client 102 did not transmit a response

440 to the request 430. In one embodiment, the appliance 200 determines the client 102 did not respond or timed out in responding upon expiration of a timer of a predetermined time out period. In yet another embodiment, the appliance 200 determines there was an error in transmission or receipt of the response 440. For example, in one embodiment, a network
5 disruption may occur and prevent transmission, or completion of transmission or receipt of the response 440.

At step 535, the appliance 200 grants or denies a level of access by the client request 405 based on validation of the request and the client authentication policy 420. In one embodiment, the policy 420 indicates the certificate from the client 102 is mandatory. In this
10 embodiment, if the appliance received a valid certificate from the client 102, the appliance 200 transmits the client request 405 to the destination, such as a server 106. For example, the appliance 200 no longer holds the client request 405 in a buffer or queue, or otherwise releases the client request 405 for transmission on a network 104. In some embodiments, the client request 402 comprises a transport layer open request. In these embodiments, if the
15 certificate is valid, the appliance 200 opens the transport layer connection to the server 106. In one embodiment, the appliance 200 associates the first transport layer connection to the client 102 with an established second transport layer connection to the server 106, such as via a pool of one or more transport layer connections.

In another embodiment, the appliance 200 associates or assigns a level of access to
20 the client 102 based on the certificate. In some embodiments, the appliance 200 associates or assigns a level of access to the client request 405, on a per request basis, based on the certificate. If the certificate is valid, the appliance 200 may assign a level of access for the request based on a group identified by the certificate or by a user associated with the request 405. In some embodiments, the appliance 200, upon validation of the response 450, transmits
25 the client request 405 without assigning a level of access.

If the certificate is not valid, the appliance 200 may provide quarantined or limited access to the client request 405 or the client 102, such as to the network 104 or a server 106.

In some embodiments, the appliance 200 transmits the client request 405 under the limited level of access. In other embodiments, based on the invalid certificate, the appliance 200

5 does not transmit the request 405 or otherwise drops the request 405. In one embodiment, the

appliance 200 does not open a transport layer connection to the server for the client request

405. In another embodiment, the appliance 200 does not associate or link the client's

transport layer connection to the appliance 200 with a transport layer connection to a server,

such as a pooled transport layer connection. In some embodiments, the appliance 200

10 transmits a message to the client 102 indicating the client request 405 or client 102 does not

meet the client authentication policy 420. In other embodiments, the appliance 200 may re-

request the certificate from the client 102.

In some embodiments, the authentication policy 420 may indicate the certificate is optional. In these embodiments, the appliance 200 may transmit the client request 405 on a

15 network 104 or to a server 106 even if it did not receive the certificate. In some cases, the

appliance 200 did receive a certificate, but the certificate was invalid, corrupt or otherwise

incomplete or received with error. In these cases, the appliance 200 may not transmit the

client request 405 or otherwise drop the request 405. In other cases, the appliance 200 inserts

in the client request 405 the client's response to authentication, and transmits the modified

20 client request to the server 405. In one embodiment, the appliance 200 inserts data related to

the authentication certificate, or response to thereof, into a Hypertext Transfer Protocol

(HTTP) header of the client request 405, and transmits the client request 405 to the server

106. In another embodiment, the appliance 200 inserts Secure Socket Layer (SSL)

information into a Hypertext Transfer Protocol (HTTP) header of the client request 405, and

25 transmits the client request 405 to the server. In this manner, the server 106, in some

embodiments, can take corrective action regarding the client's authentication response. In some embodiments, if a valid client certificate was provided, although optional, the appliance may assign the client request 405 or client 102 a higher level of access. In other embodiments, if an invalid or corrupt client certificate was provided, although optional, the
5 appliance may assign the client request 405 to a limited or quarantined level of access.

In view of the structure, functions and operations of the appliance described herein, the appliance provides fine-grained triggering of client authentication policy actions on client requests. Any portion of the client request, from URL patterns to functions or methods of the request to IP related information, may be used to trigger a client authentication request from
10 the appliance to the client. As such, the appliance can be configured with fine-grained client authentication policies on a per request basis or per client basis. Before providing access to a network or server, the appliance can trigger these authentication actions and validate the client. The appliance can control, maintain and protect the level of access to resources accessed via the gateway for a wide variety of client types, users and client access scenarios.
15 A multitude of authentication policies can be configured and triggered to handle different levels of access to protected resources by these different clients via the appliance. This can be useful in protecting against attacks, probing or zombie clients.

We Claim:

1. A method for an appliance to authenticate access of a client to a protected resource on a server via the appliance, the method comprising the steps of:

(a) receiving, by an appliance, a client request to access a protected resource of a server;

(b) determining, by the appliance, a portion of the client request matches a corresponding specification of a client authentication policy of the appliance;

(c) queuing, by the appliance, the client request upon determining the portion of the client request matches the client authentication policy;

(d) determining, by the appliance, the client authentication policy identifies an action to request an authentication certificate from the client in order to access the server via the appliance, and

(e) transmitting, by the appliance in response to the client authentication policy, a request to the client for an authentication certificate.

2. The method of claim 1, comprising determining, by the appliance one or more of the following portions of the client request matches a corresponding specification of the client authentication policy: a Uniform Resource Locator (URL) pattern, an identifier of one of a method or function, a directory, a client network identifier, a server network identifier, a network port, and a Secure Socket Layer (SSL) parameter.

3. The method of claim 1, comprising determining, by the appliance, the client has been previously authenticated to access the protected resource, and allowing access, by the appliance, to the protected resource.

4. The method of claim 1, specifying, by the client authentication policy, that the authentication certificate is mandatory

5. The method of claim 4, comprising not transmitting, by the appliance, the client request to the server upon one of receiving an invalid authentication certificate or not receiving an authentication certificate.

6. The method of claim 1, specifying, by the authentication policy, that the authentication certificate is optional

7. The method of claim 6, comprising transmitting, by the appliance, the client request
5 to the server upon one of not receiving an authentication certificate from the client or receiving an invalid authentication certificate from the client.

8. The method of claim 6, comprising inserting, by the appliance, into the client request a portion of the client's response to the request for the authentication certificate, and
10 transmitting the client request to the server.

9. The method of claim 1, comprising inserting, by the appliance in response to the client authentication policy, data related to the authentication certificate into a Hypertext Transfer Protocol (HTTP) header of the client request, and transmitting the client request to
15 the server.

10. The method of claim 1, comprising inserting, by the appliance in response to the client authentication policy, Secure Socket Layer (SSL) information into a Hypertext Transfer Protocol (HTTP) header of the client request, and transmitting the client request to
20 the server.

11. The method of claim 1, comprising identifying, by the client authentication policy, a pattern for associating a portion of the client request with the client authentication policy

12. The method of claim 1, wherein step (c) comprises preventing, by the appliance, establishment of a transport layer connection with the server.

13. An appliance for providing finer control for authenticating access of a client to a protected resource on a server, the appliance comprising:

30 means for receiving a client request to access a protected resource of a server;
means for determining a portion of the client request matches a corresponding specification of a client authentication policy of the appliance;
means for queuing the client request upon determining the portion of the client request matches the client authentication policy;

means for determining the client authentication policy identifies an action to request an authentication certificate from the client in order to access the server via the appliance, and

means for transmitting, in response to the client authentication policy, a request to the client for an authentication certificate.

5

14. The appliance of claim 13, comprising means for determining one or more of the following portions of the client request matches a corresponding specification of the client authentication policy: a Uniform Resource Locator (URL) pattern, an identifier of one of a method or function, a directory, a client network identifier, a server network identifier, a network port, and a Secure Socket Layer (SSL) parameter.

10

15. The appliance of claim 13, comprising means for determining the client has been previously authenticated to access the protected resource, and allowing access, by the appliance, to the protected resource.

15

16. The appliance of claim 13, comprising means for specifying, by the client authentication policy, that the authentication certificate is mandatory

17. The appliance of claim 16, comprising means for not transmitting, by the appliance, the client request to the server upon one of receiving an invalid authentication certificate or not receiving an authentication certificate.

20

18. The appliance of claim 13, comprising means for specifying, by the authentication policy, that the authentication certificate is optional

25

19. The appliance of claim 18, comprising means for transmitting the client request to the server upon one of not receiving an authentication certificate or receiving an invalid authentication certificate from the client.

30

20. The appliance of claim 18, comprising means for inserting, into the client request a portion of the client's response to the request for the authentication certificate, and transmitting the client request to the server.

21. The appliance of claim 13, comprising means for inserting, in response to the client authentication policy, data related to the authentication certificate into a Hypertext Transfer Protocol (HTTP) header of the client request, and transmitting the client request to the server
- 5 22. The appliance of claim 13, comprising means for inserting, by the appliance in response to the client authentication policy, Secure Socket Layer (SSL) information into a Hypertext Transfer Protocol (HTTP) header of the client request, and transmitting the client request to the server.
- 10 23. The appliance of claim 13, comprising means for identifying, by the client authentication policy, a pattern for associating a portion of the client request with the client authentication policy
24. The appliance of claim 13, comprising means for preventing establishment of a
15 transport layer connection with the server.

1/9

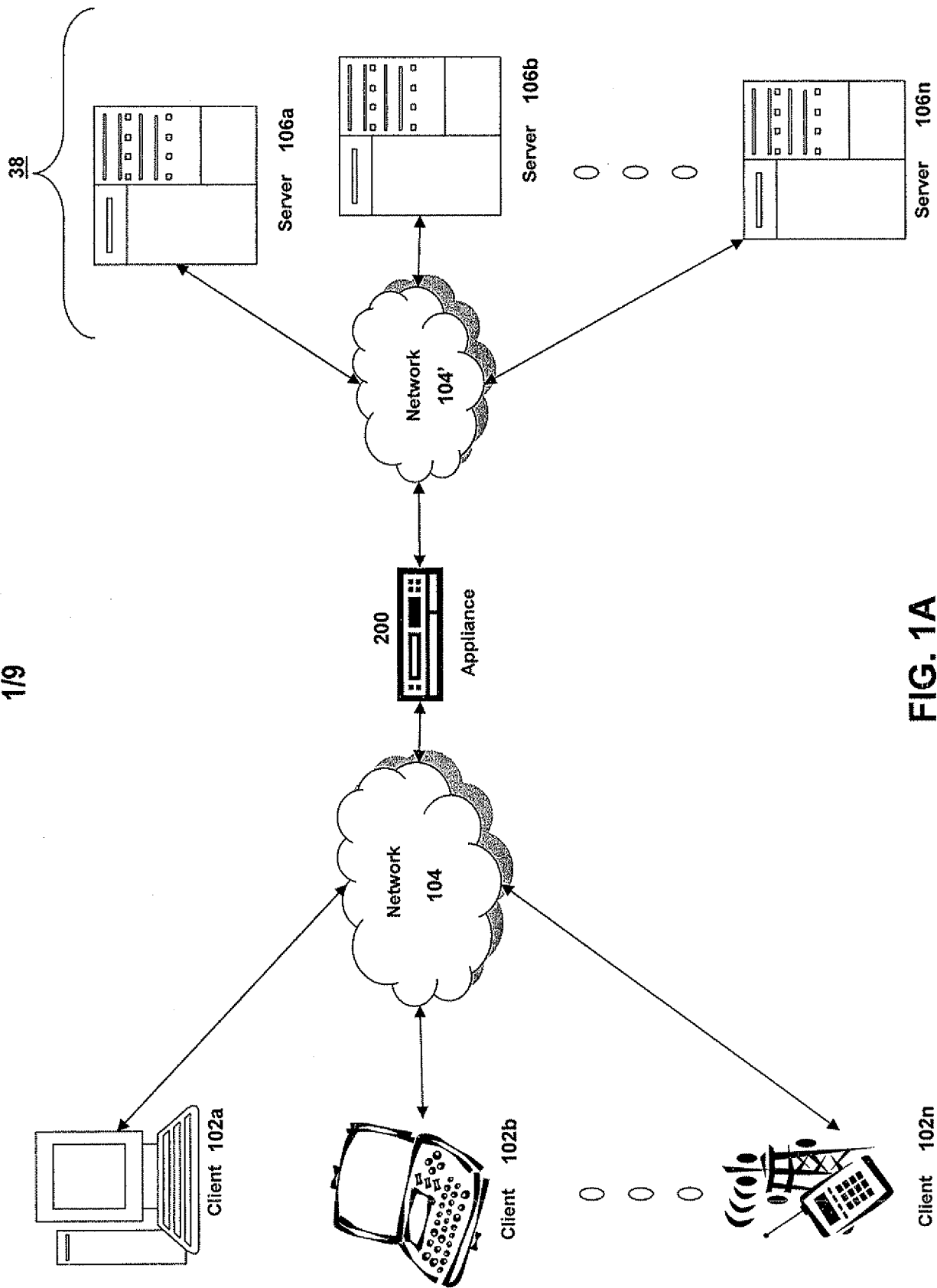


FIG. 1A

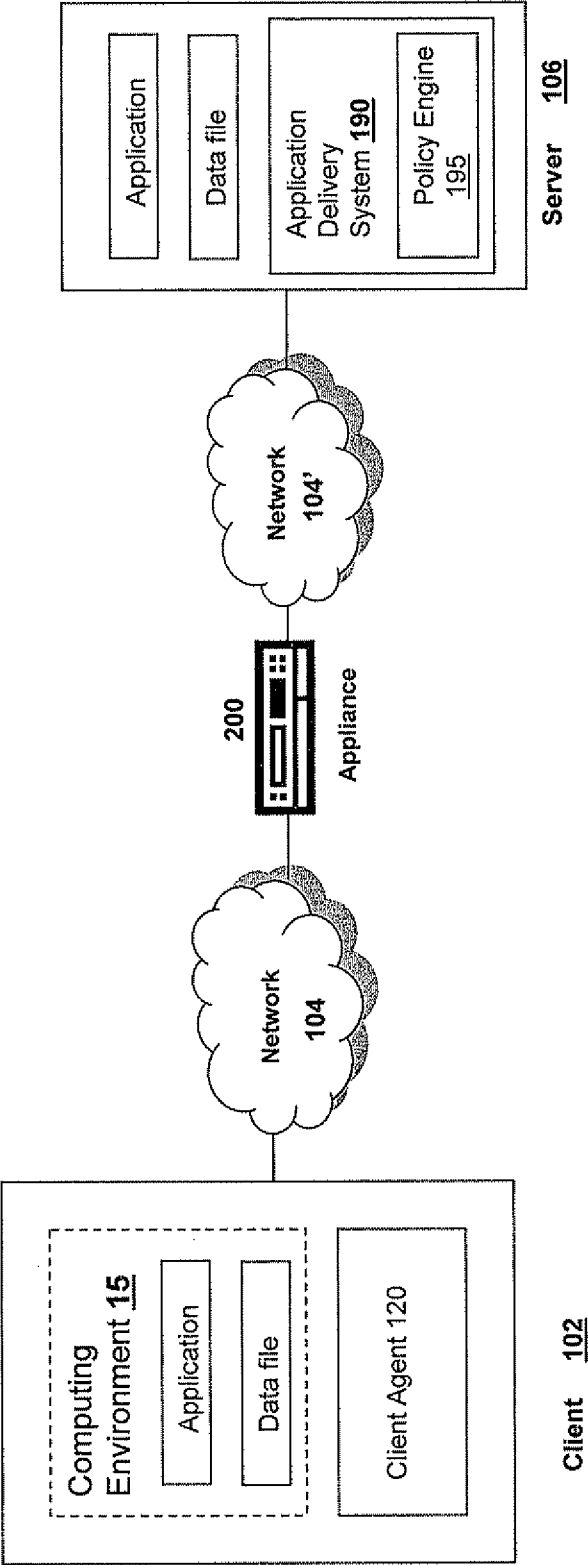
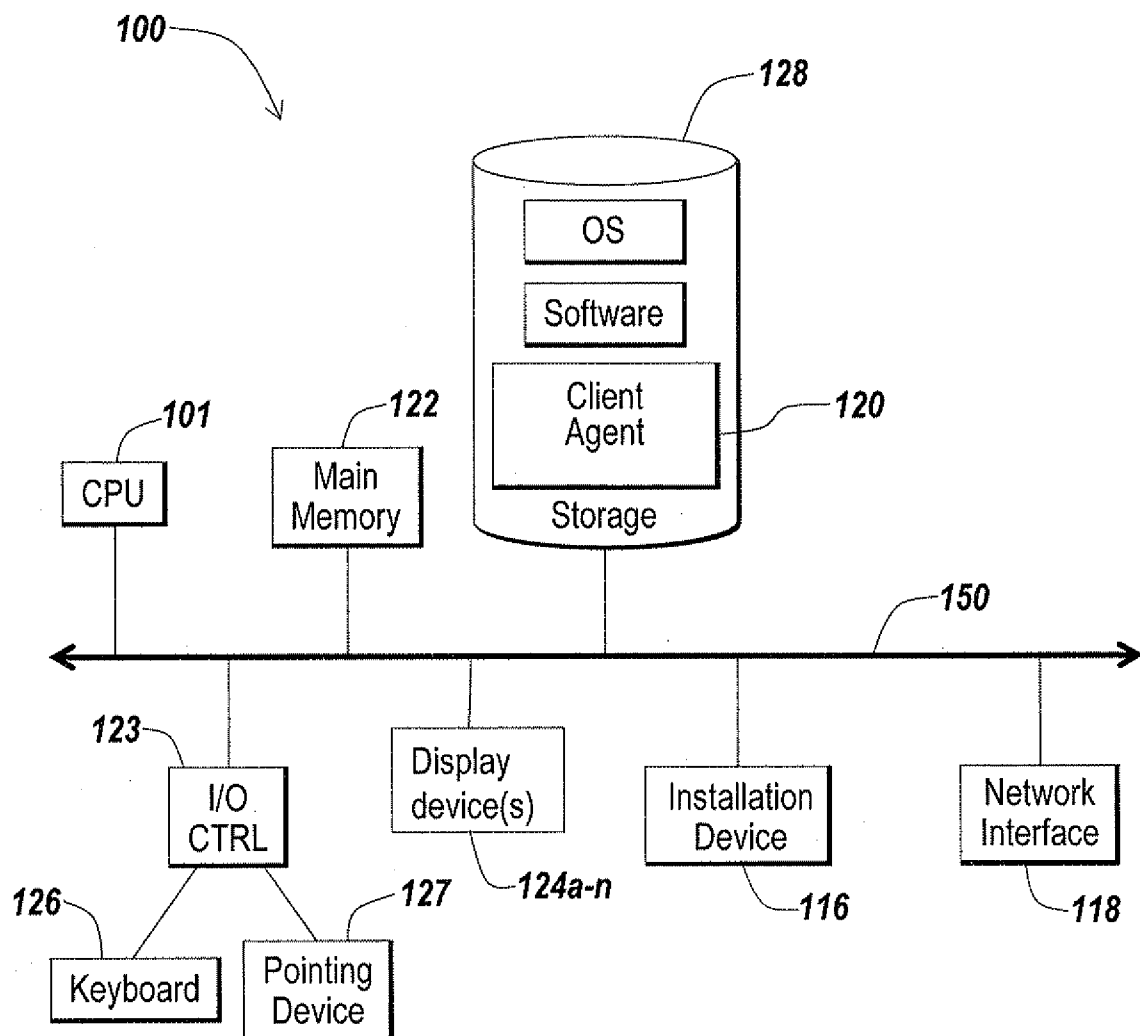
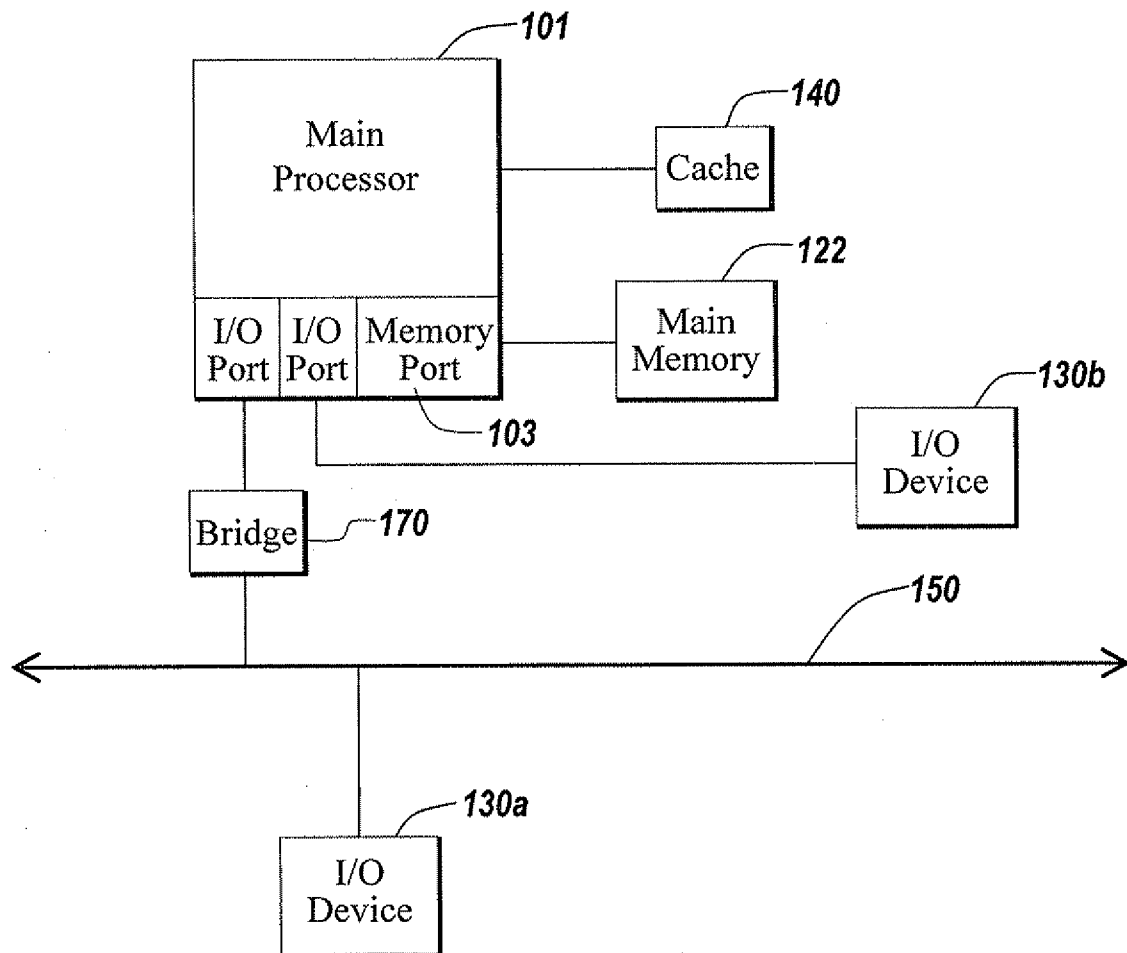


FIG. 1B

3/9

*Fig. 1C*

4/9

*Fig. 1D*

5/9

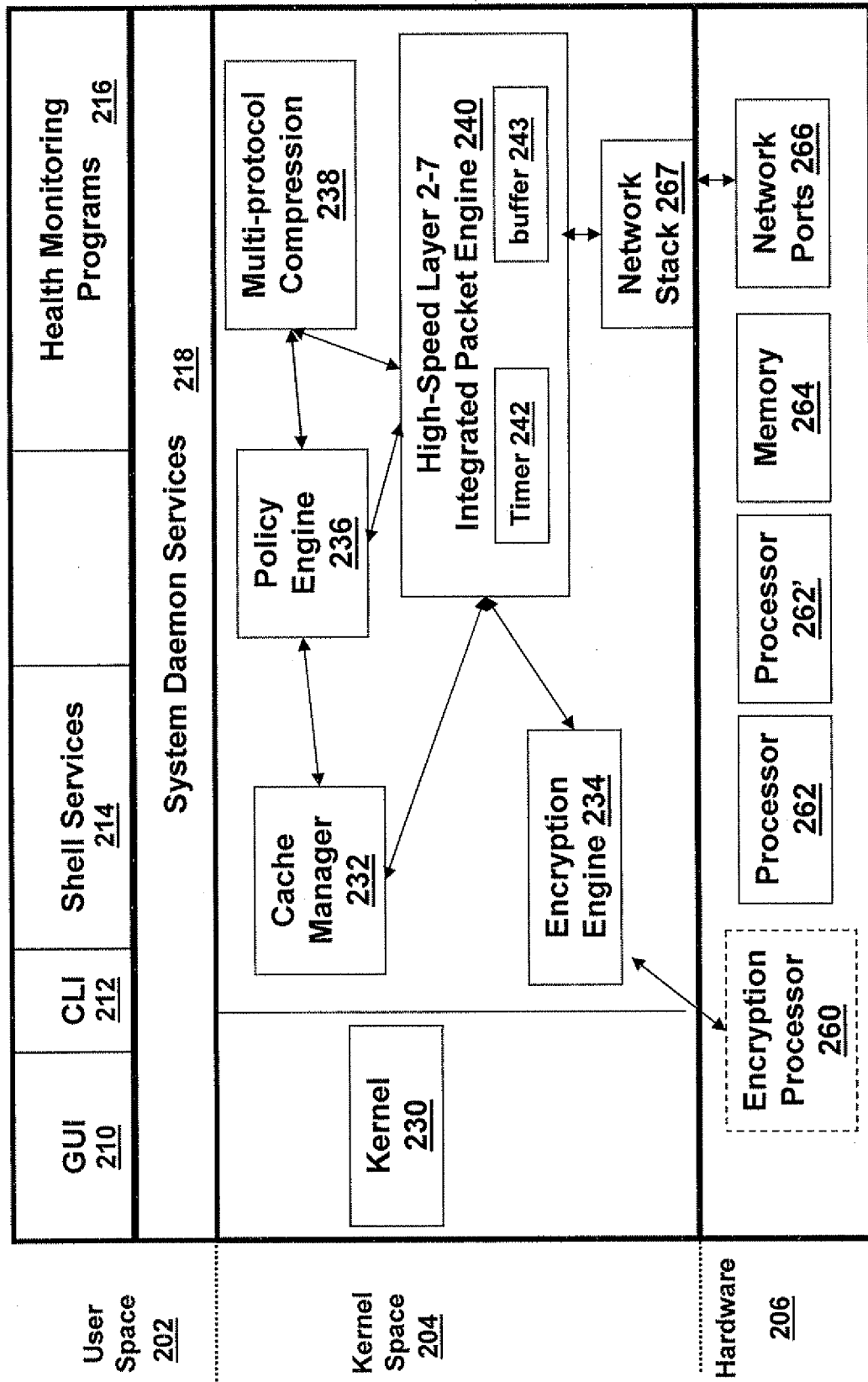


FIG. 2A

6/9

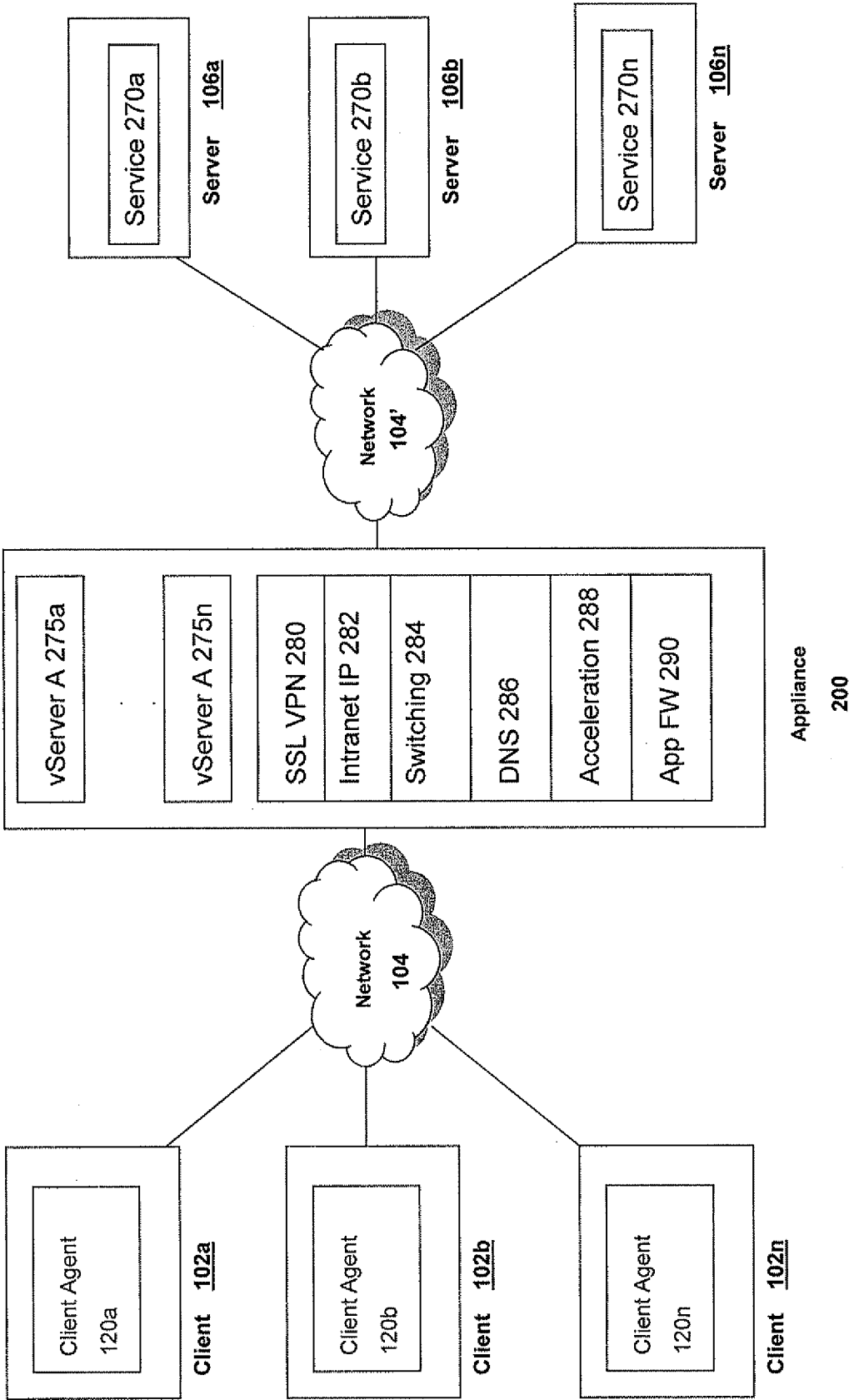
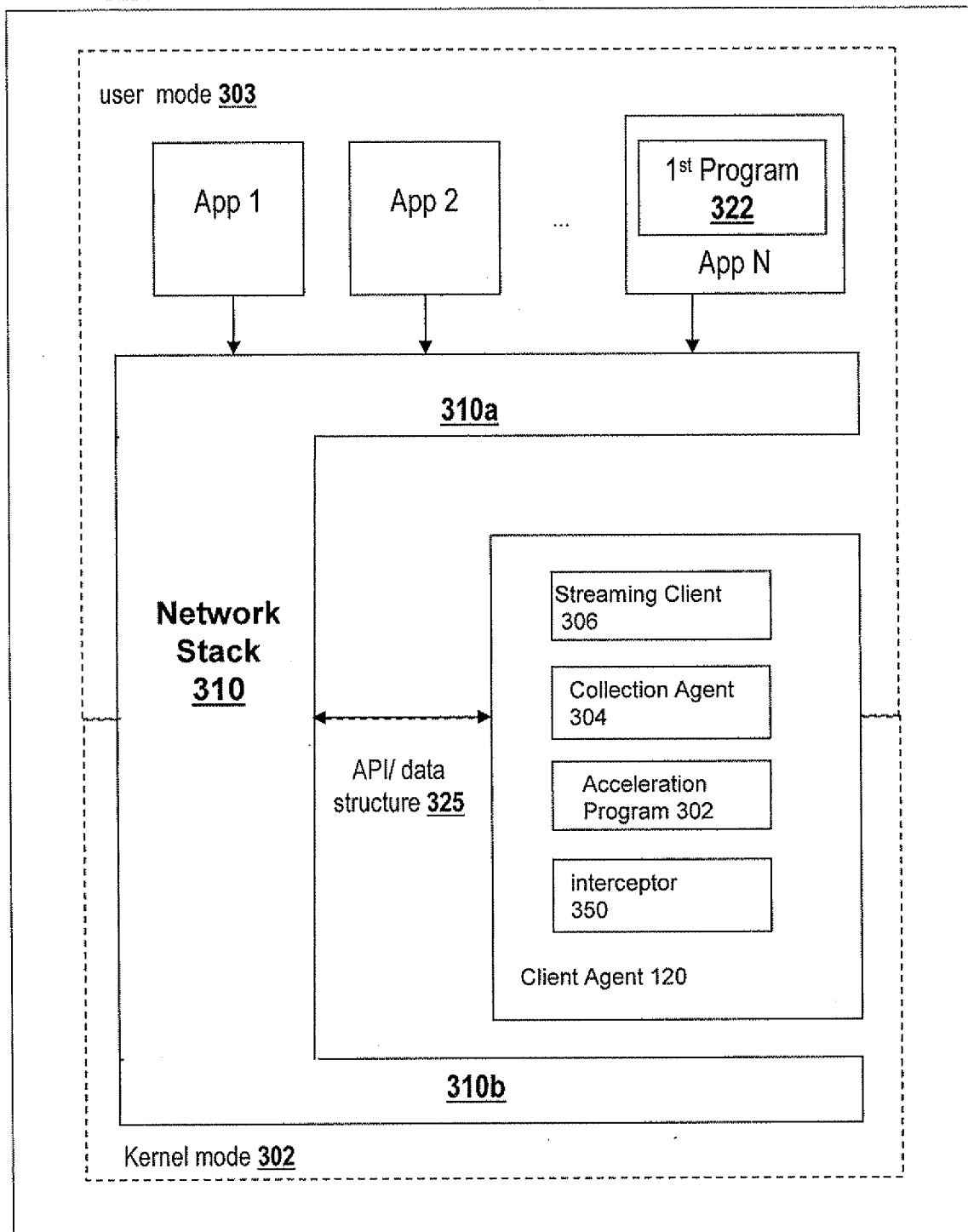


FIG. 2B

7/9

Client 102

100

Fig. 3

8/9

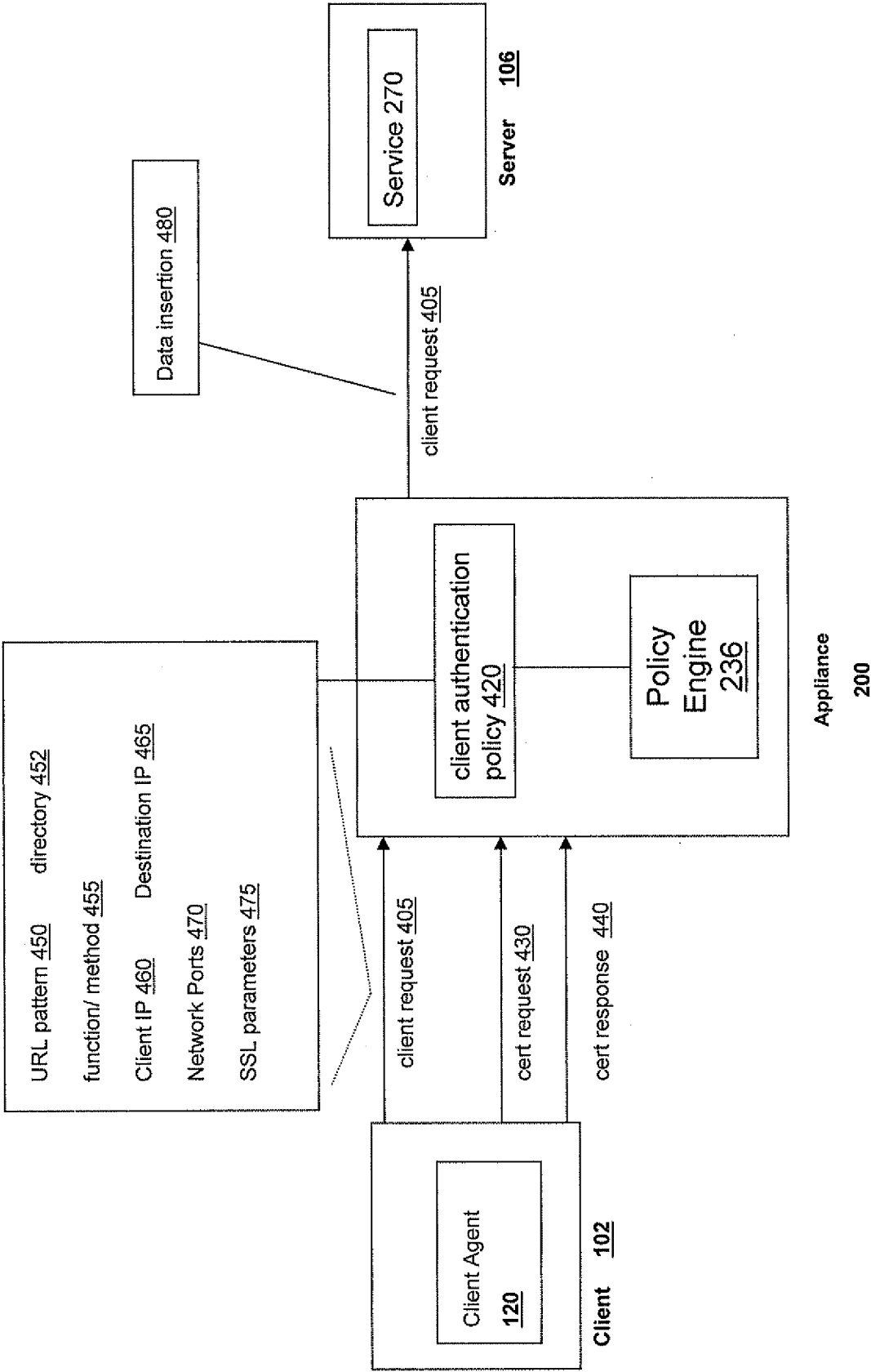


FIG. 4

9/9

