

# **EXHIBIT A**

Part 1 of 2



US007596784B2

(12) **United States Patent**  
**Abrams et al.**

(10) **Patent No.:** **US 7,596,784 B2**  
(45) **Date of Patent:** **Sep. 29, 2009**

(54) **METHOD SYSTEM AND APPARATUS FOR PROVIDING PAY-PER-USE DISTRIBUTED COMPUTING RESOURCES**

5,765,205 A 6/1998 Breslau et al.  
5,903,762 A \* 5/1999 Sakamoto et al. .... 717/178  
6,065,123 A 5/2000 Chou et al.  
6,771,290 B1 \* 8/2004 Hoyle ..... 715/745  
2002/0178244 A1 \* 11/2002 Brittenham et al. .... 709/223  
2003/0200541 A1 \* 10/2003 Cheng et al.

(75) Inventors: **Peter C. Abrams**, Belmont, CA (US);  
**Rajeev Bharadhwaj**, Los Altos, CA (US);  
**Swami Nathan**, San Jose, CA (US);  
**Robert Rodriguez**, Fremont, CA (US);  
**Craig W. Martyn**, San Francisco, CA (US)

**OTHER PUBLICATIONS**

“Security Roadmap for Ejasent’s Utility Computing Network”, 2001, 18 pgs.

\* cited by examiner

*Primary Examiner*—Chuck O Kendall

(74) *Attorney, Agent, or Firm*—Meyertons Hood Kivlin Kowert & Goetzel, P.C.

(73) Assignee: **Symantec Operating Corporation**,  
Cupertino, CA (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1665 days.

(21) Appl. No.: **09/950,559**

(57) **ABSTRACT**

(22) Filed: **Sep. 10, 2001**

(65) **Prior Publication Data**

US 2002/0166117 A1 Nov. 7, 2002

**Related U.S. Application Data**

(60) Provisional application No. 60/232,052, filed on Sep. 12, 2000.

(51) **Int. Cl.**

**G06F 9/44** (2006.01)  
**G06F 9/445** (2006.01)  
**G06F 15/173** (2006.01)

(52) **U.S. Cl.** ..... **717/172; 717/120; 717/127; 717/178; 709/224; 709/225; 709/226**

(58) **Field of Classification Search** ..... **717/168–178; 709/201–244**

See application file for complete search history.

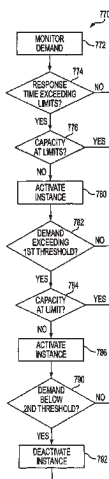
(56) **References Cited**

**U.S. PATENT DOCUMENTS**

5,732,275 A \* 3/1998 Kullick et al. .... 717/170

**25 Claims, 25 Drawing Sheets**

Method, system, apparatus, and computer program and computer program product provide on-demand, scalable computational resources to application providers over a distributed network and system. Resources are made available based on demand for applications. Application providers are charged fees based on the amount of resources utilized to satisfy the needs of the application. In providing compute resources, method and apparatus is capable of rapidly activating a plurality of instances of the applications as demand increases and to halt instances as demand drops. Application providers are charged based on metered amount of computational resources utilized in processing their applications. Application providers access the network to distribute applications onto network to utilize distributed compute resources for processing of the applications. Application providers are further capable of monitoring, updating and replacing distributed applications. Apparatus and system includes plurality of computing resources distributed across a network capable of restoring and snapshotting provisioned applications based on demand.



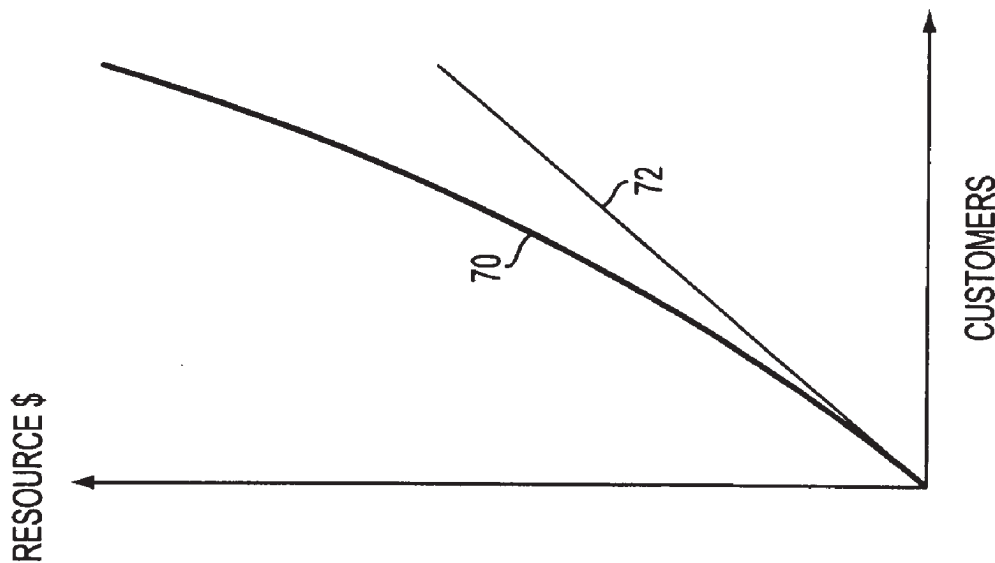


FIG. 2

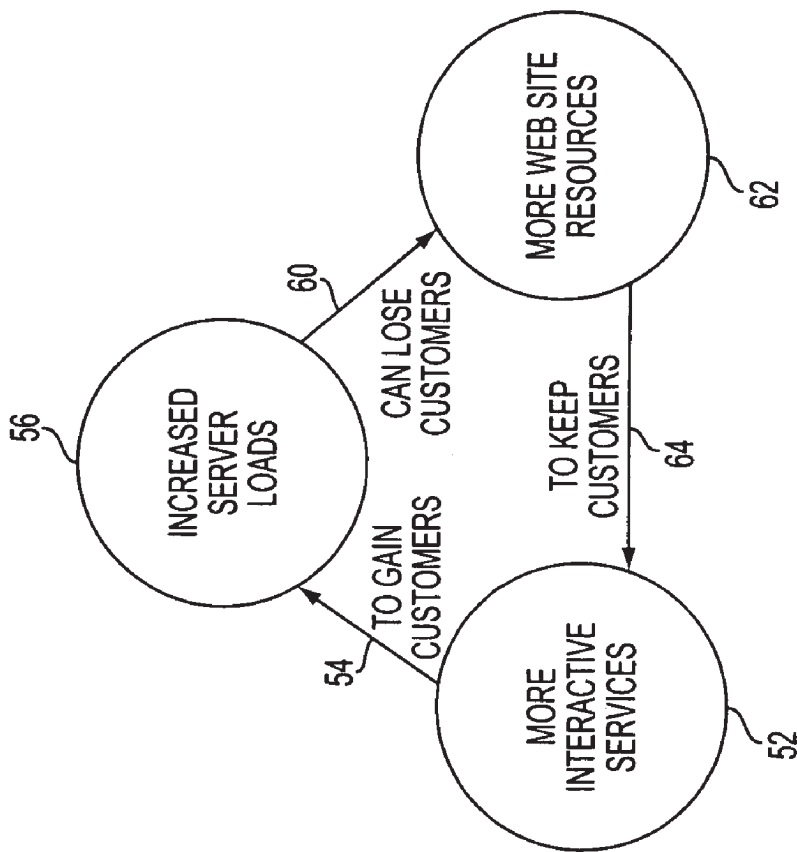
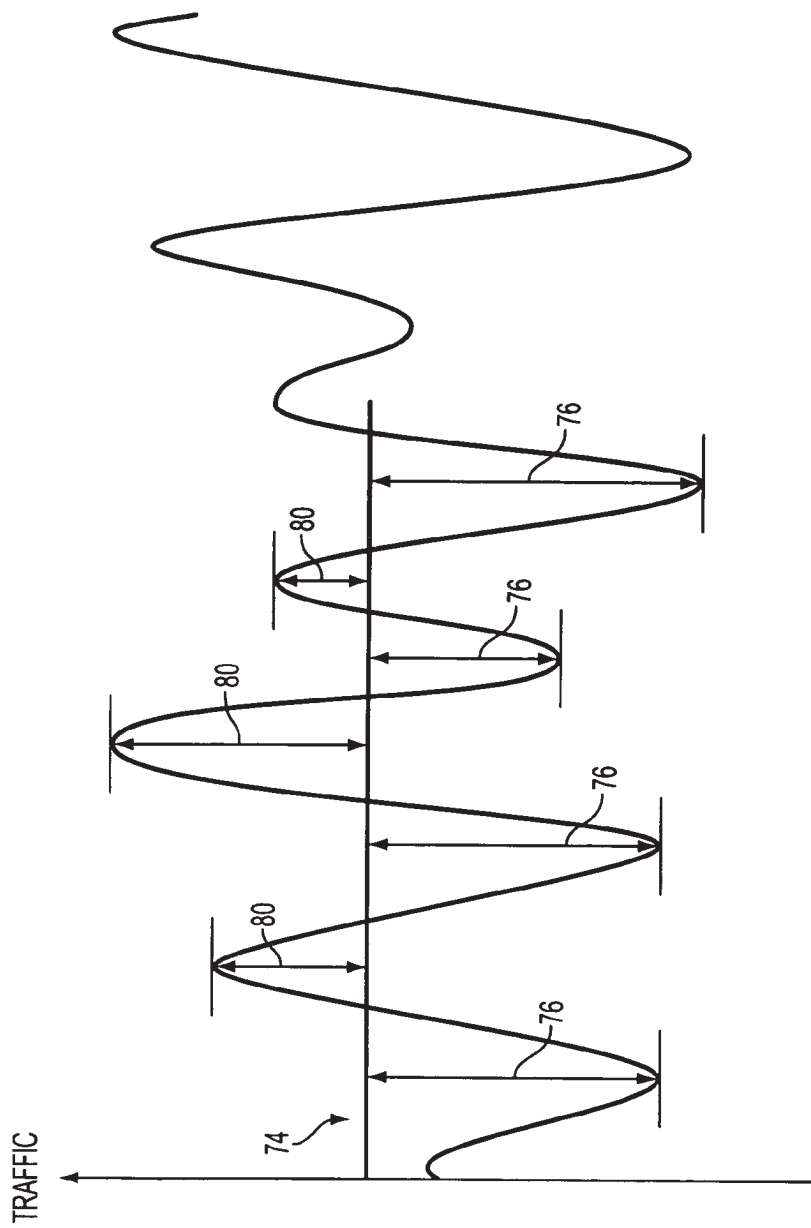
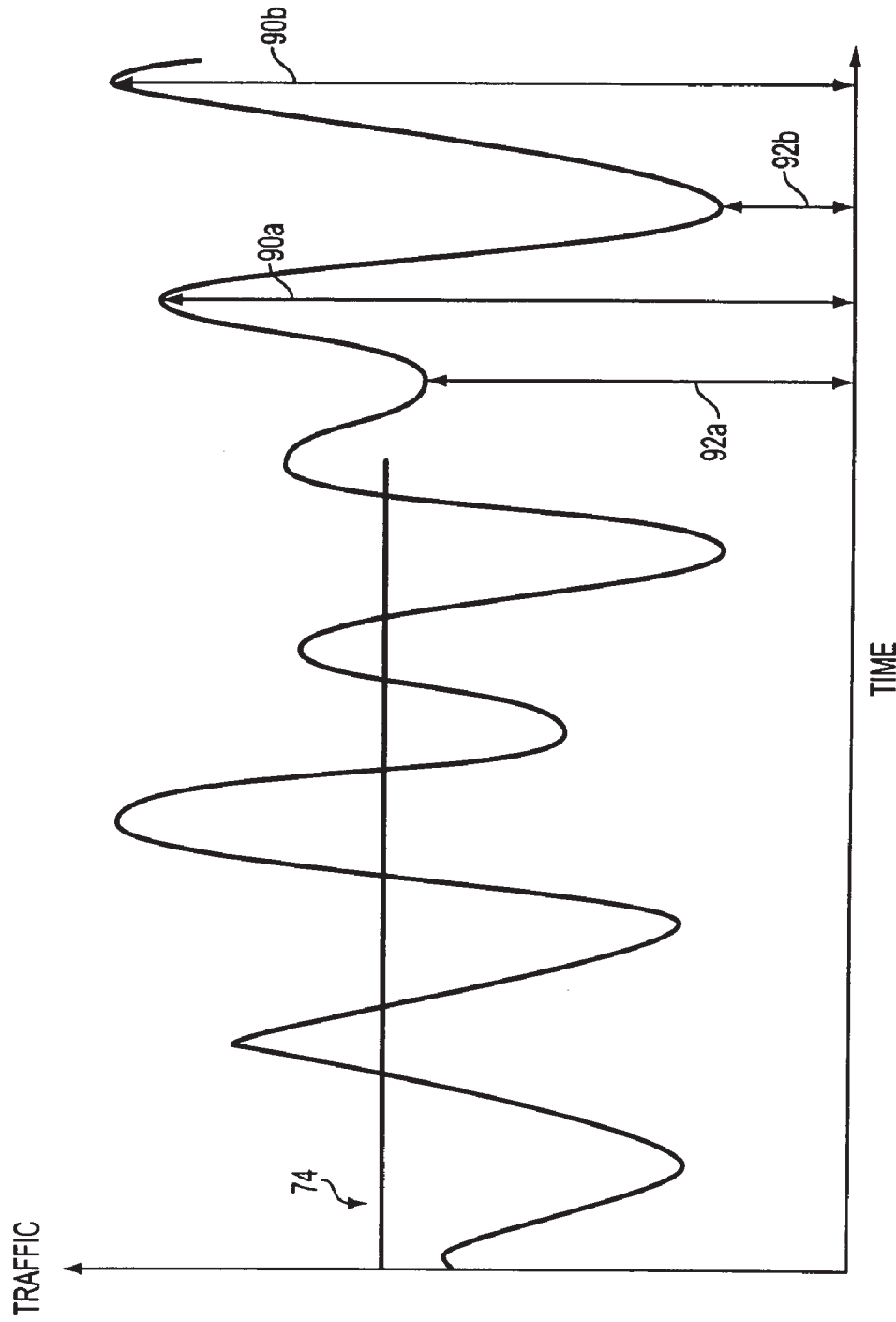


FIG. 1



TIME  
FIG. 3



TIME  
FIG. 4

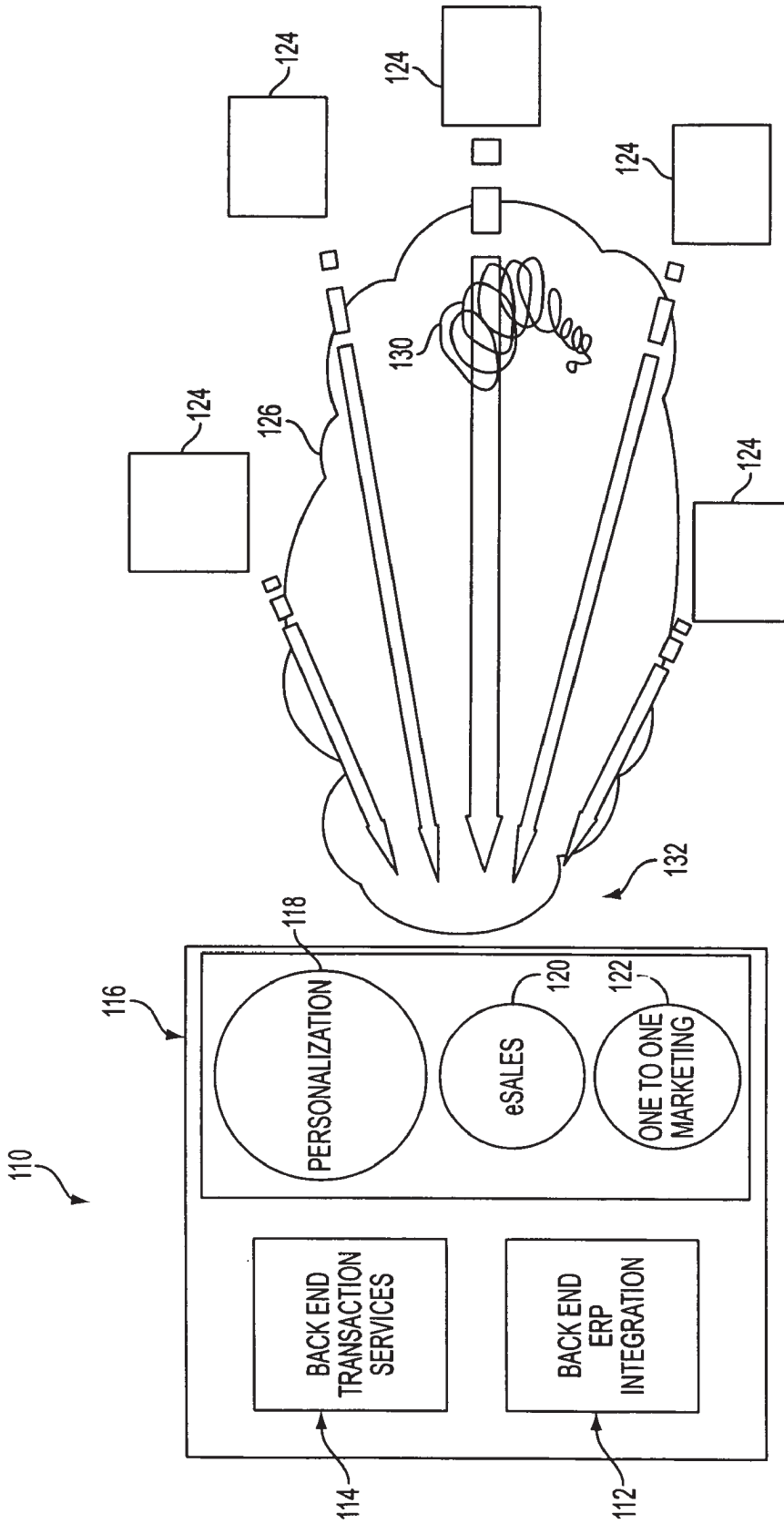


FIG. 5

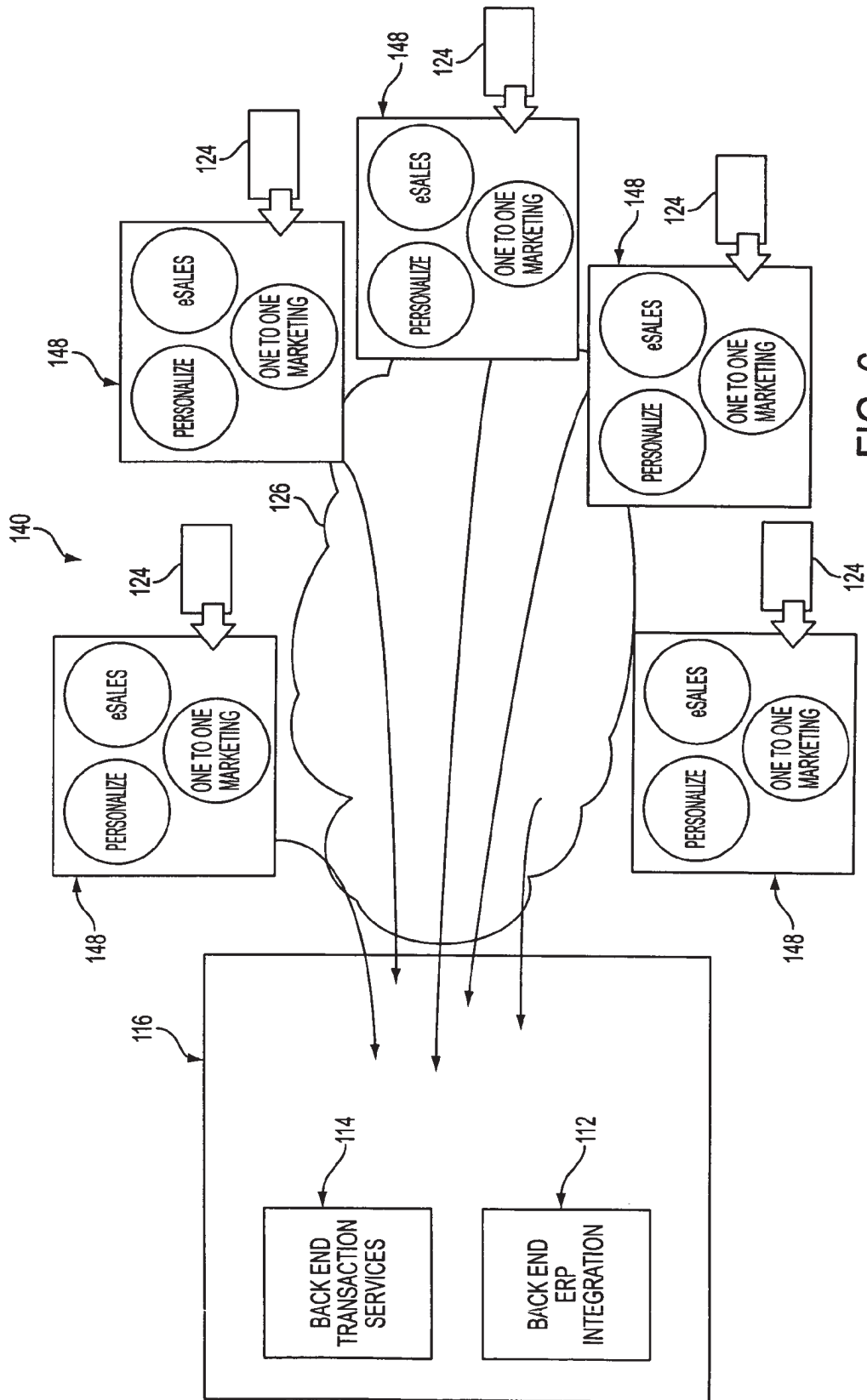


FIG. 6

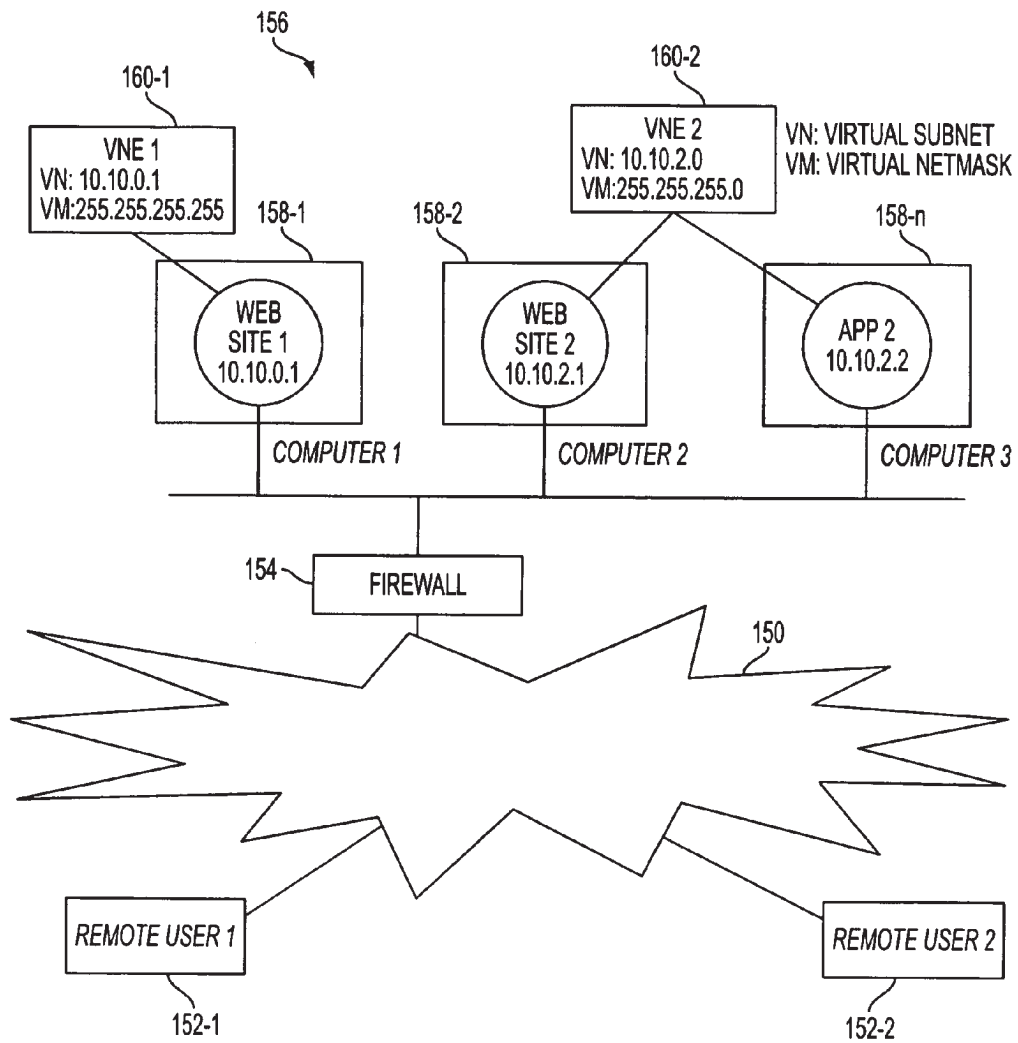


FIG. 7



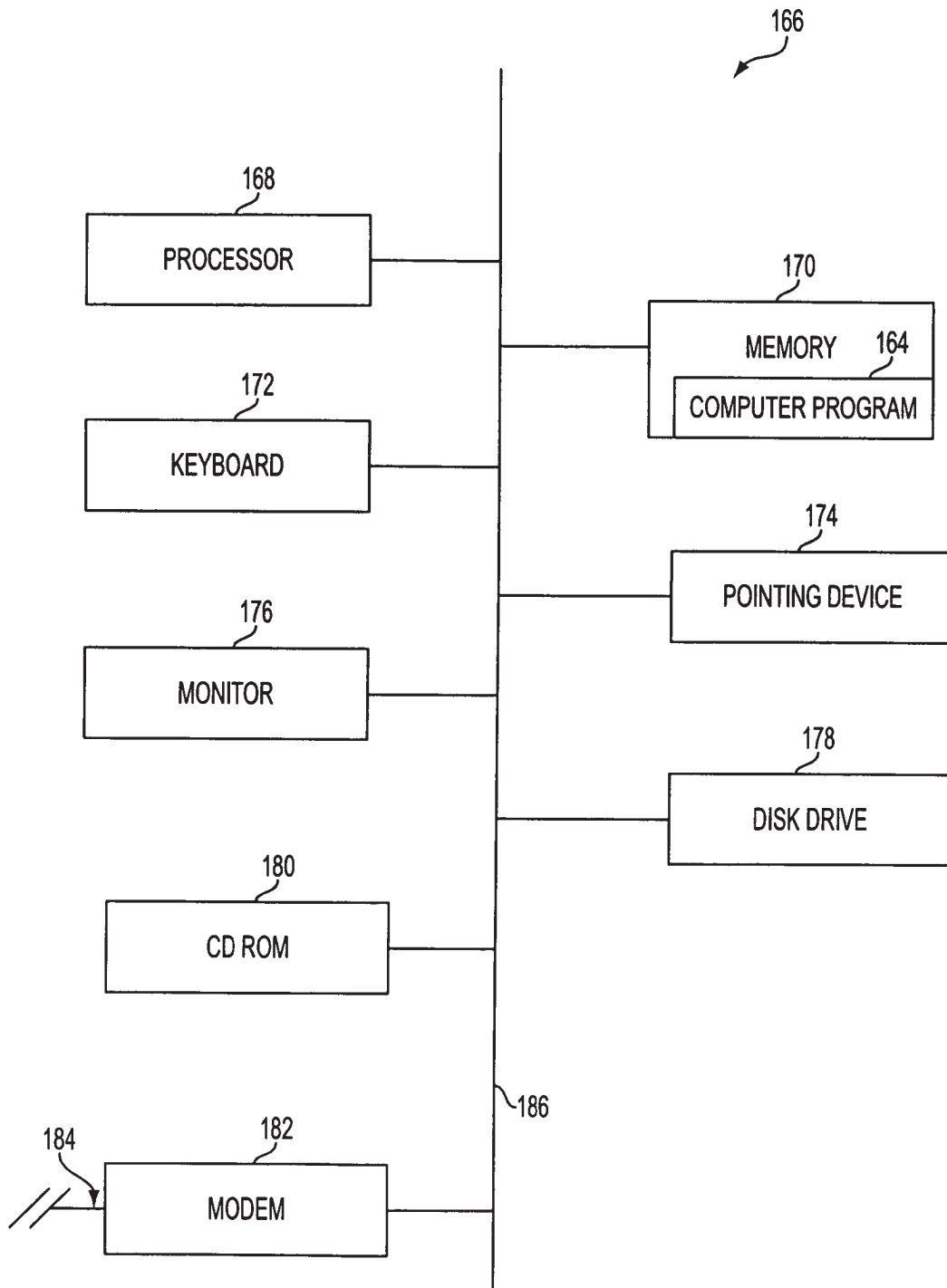


FIG. 8

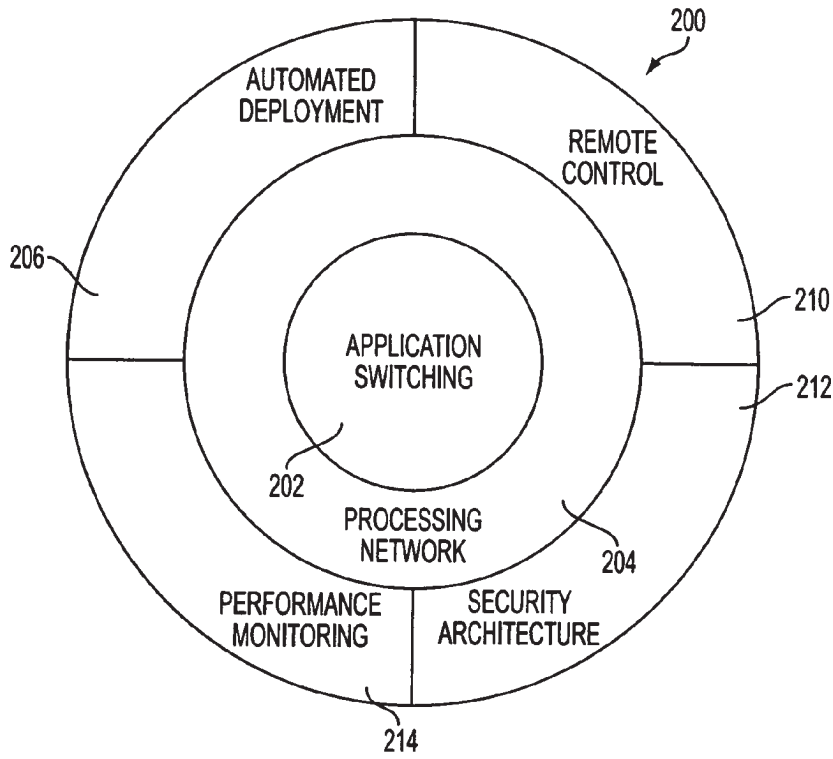


FIG. 9

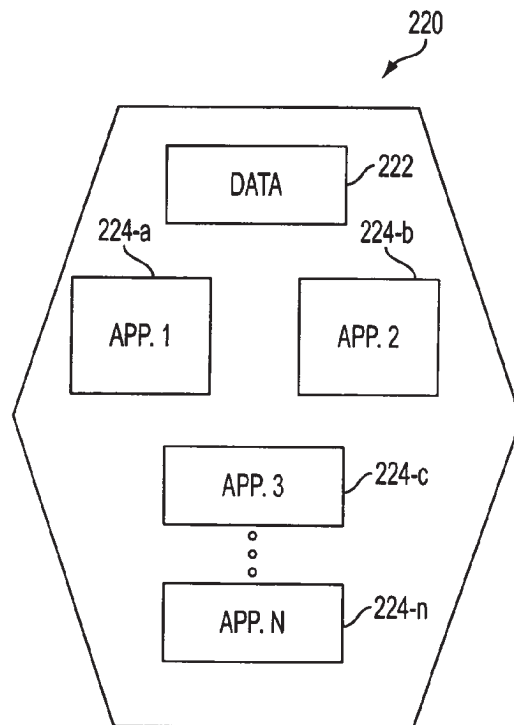


FIG. 10

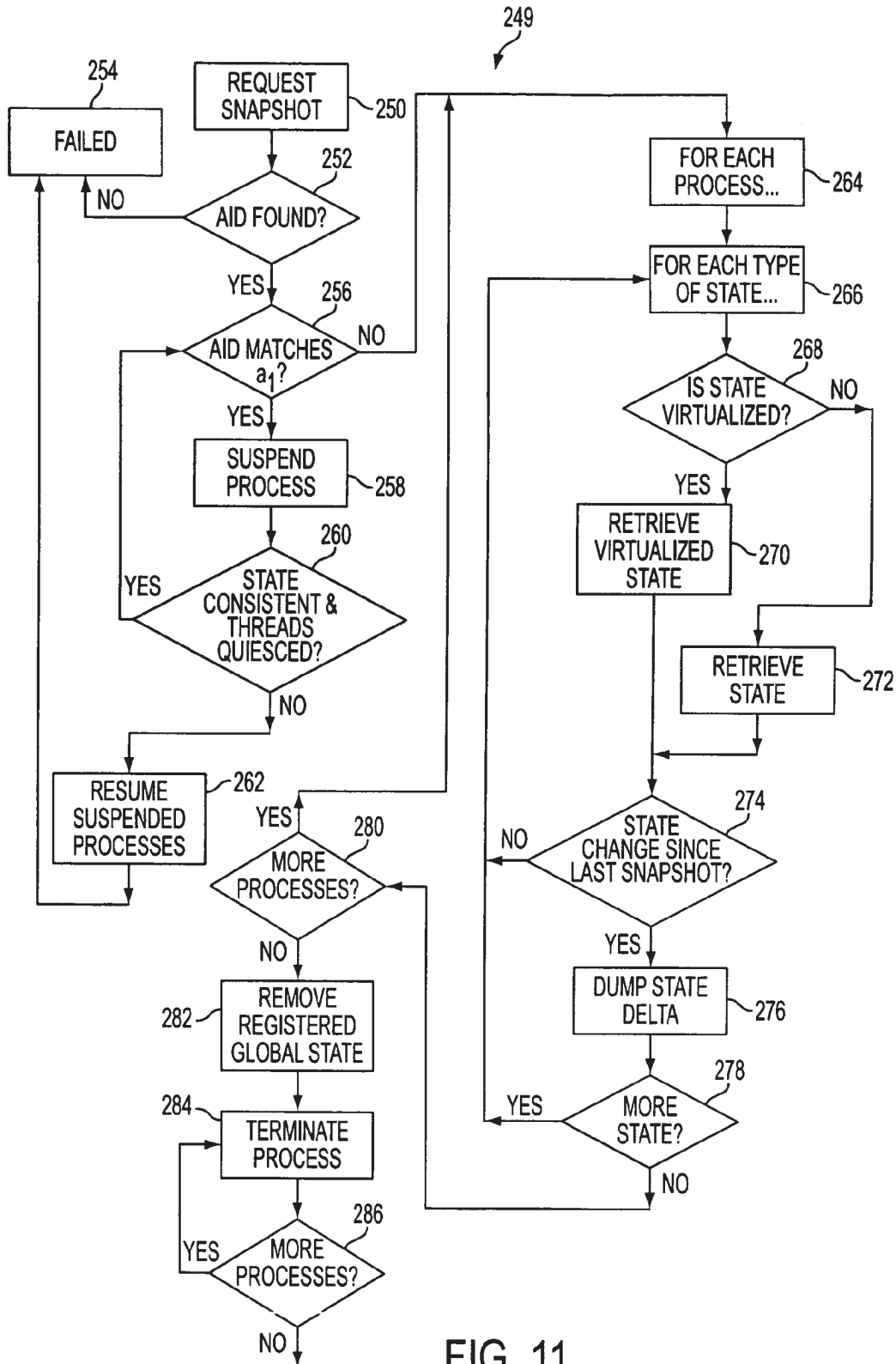


FIG. 11

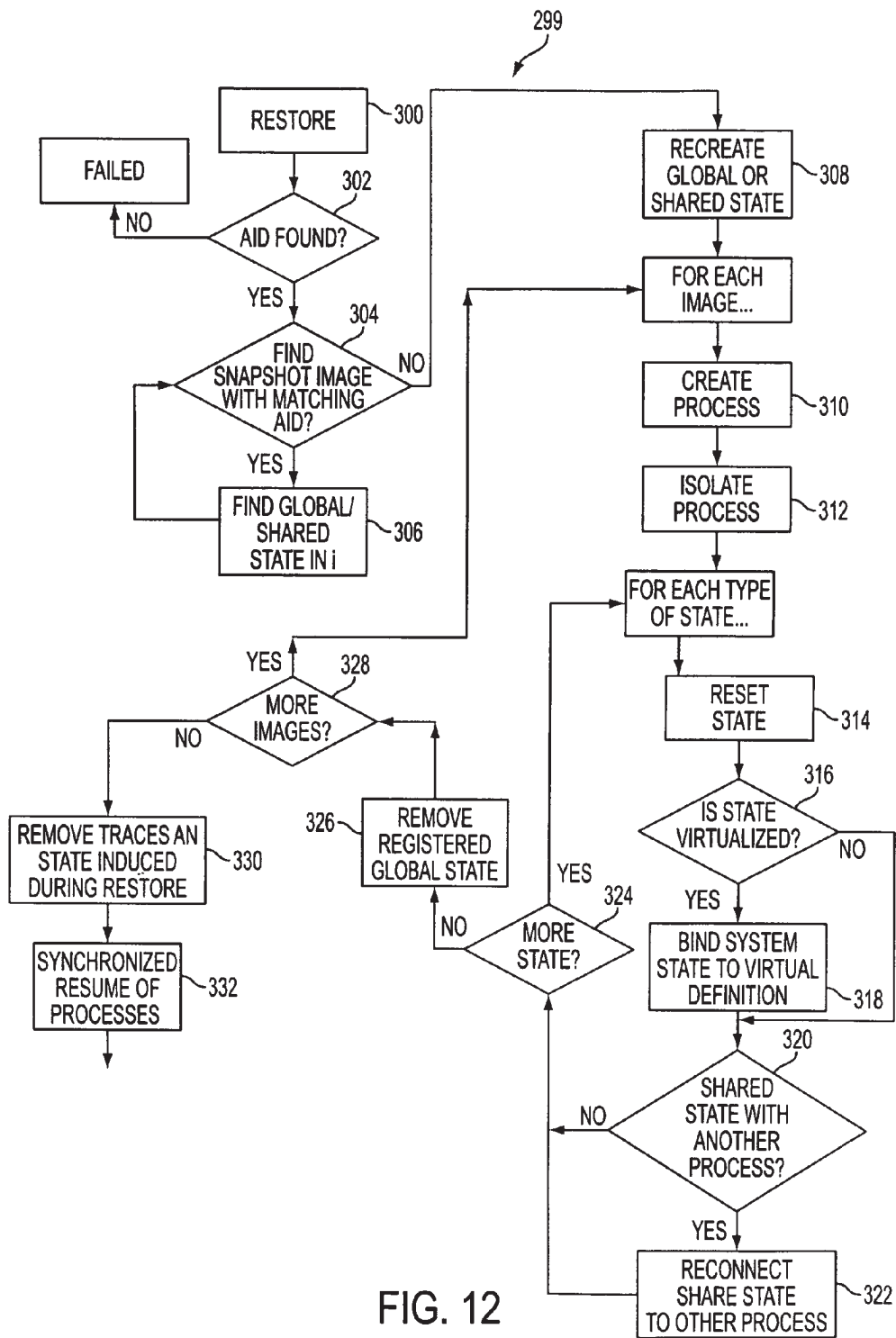


FIG. 12

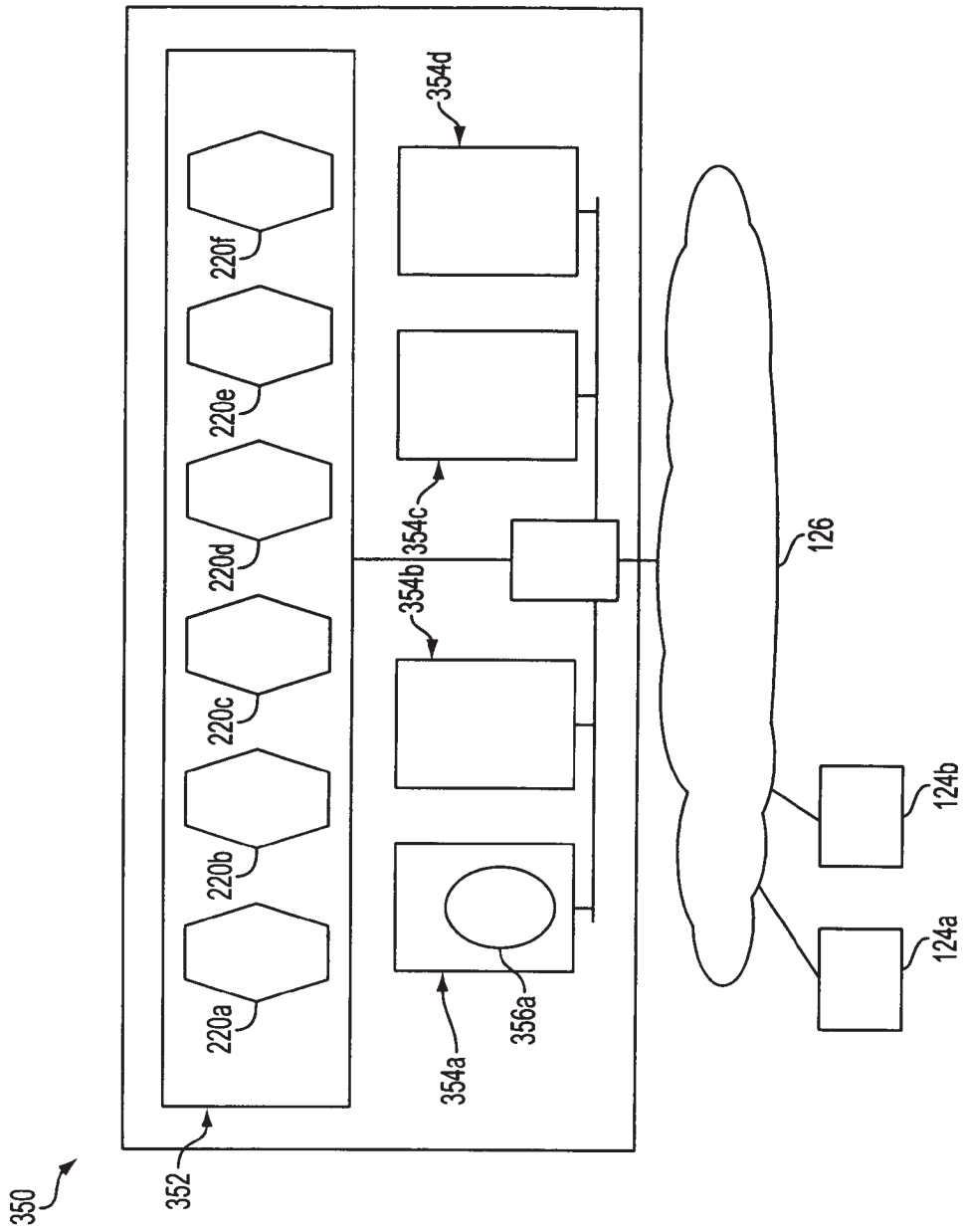


FIG. 13A

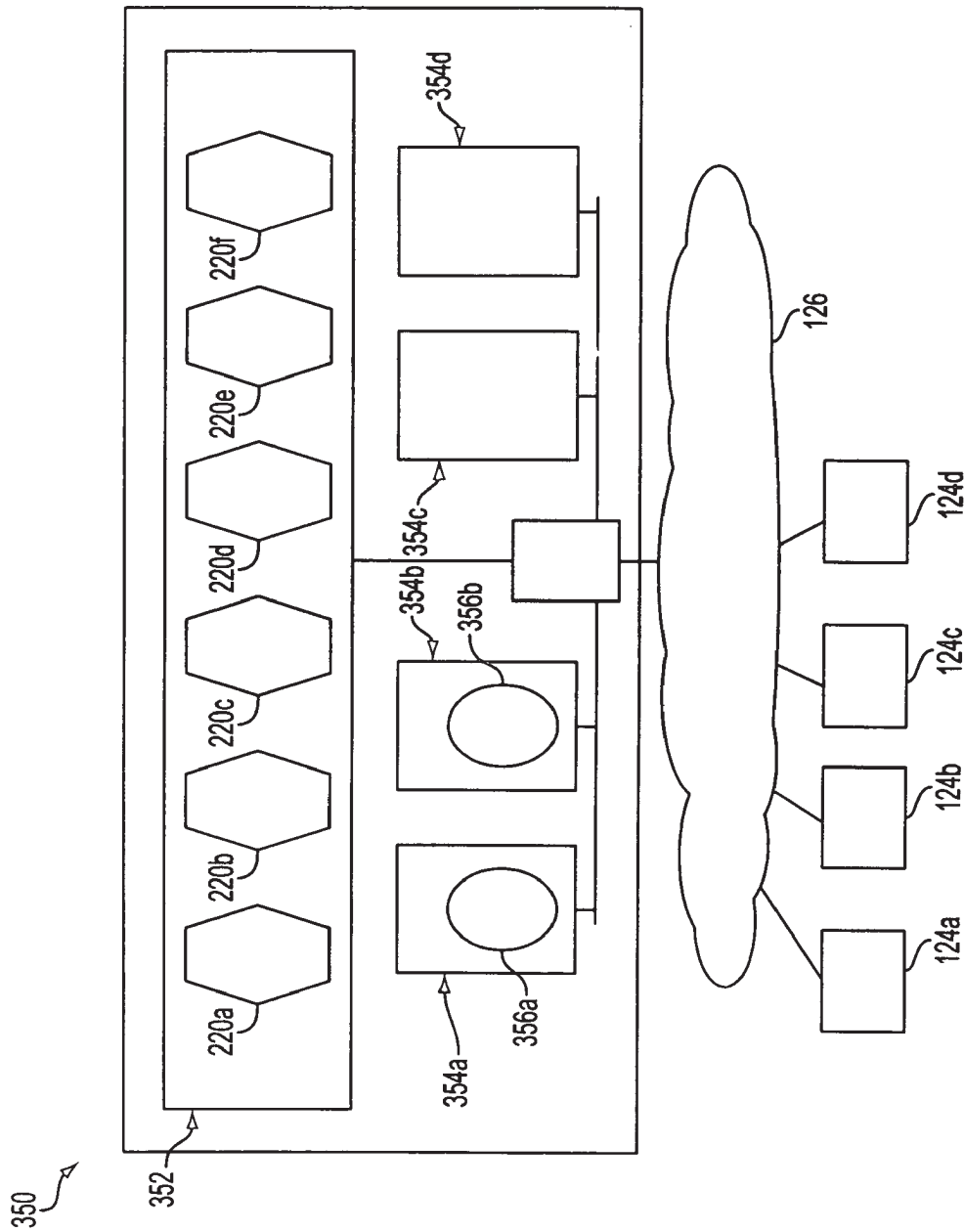


FIG. 13B

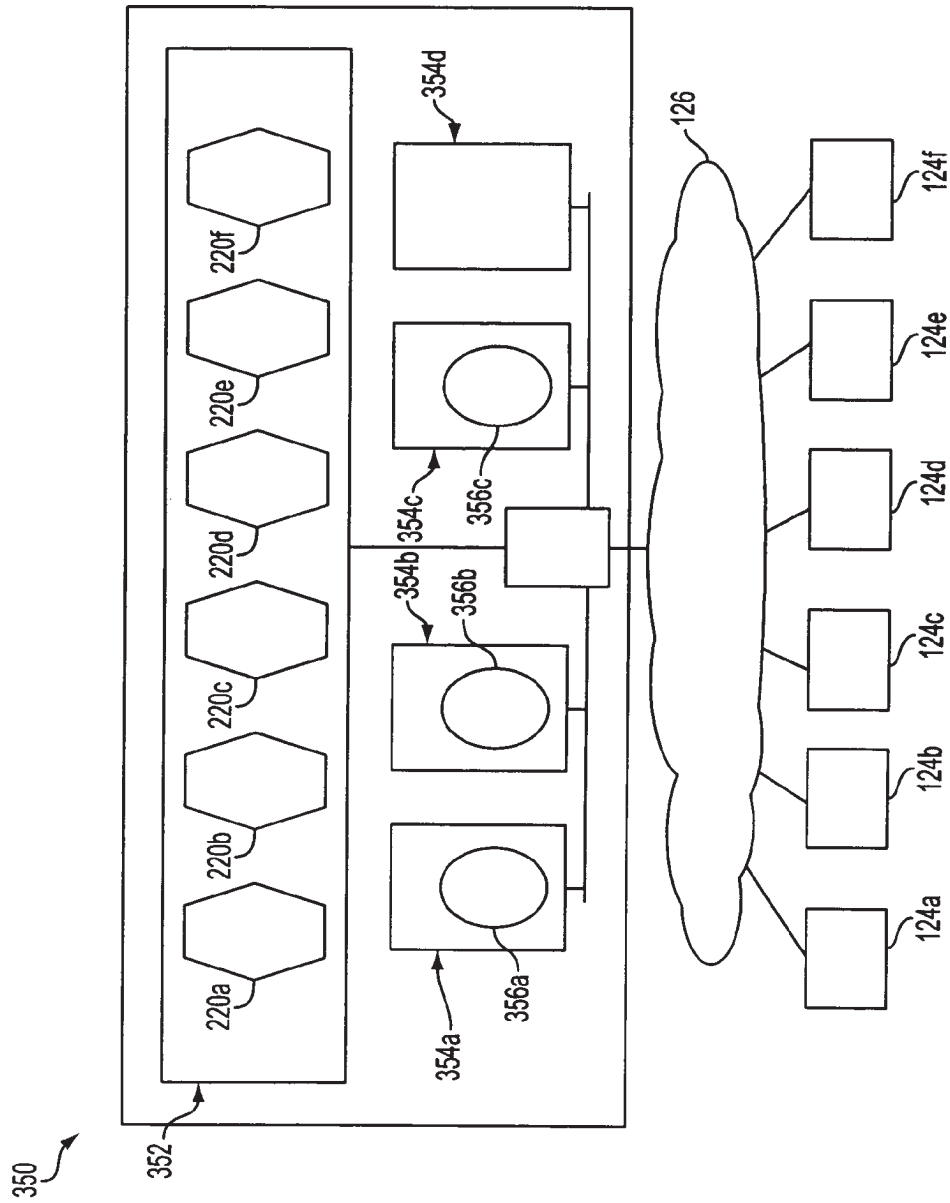


FIG. 13C

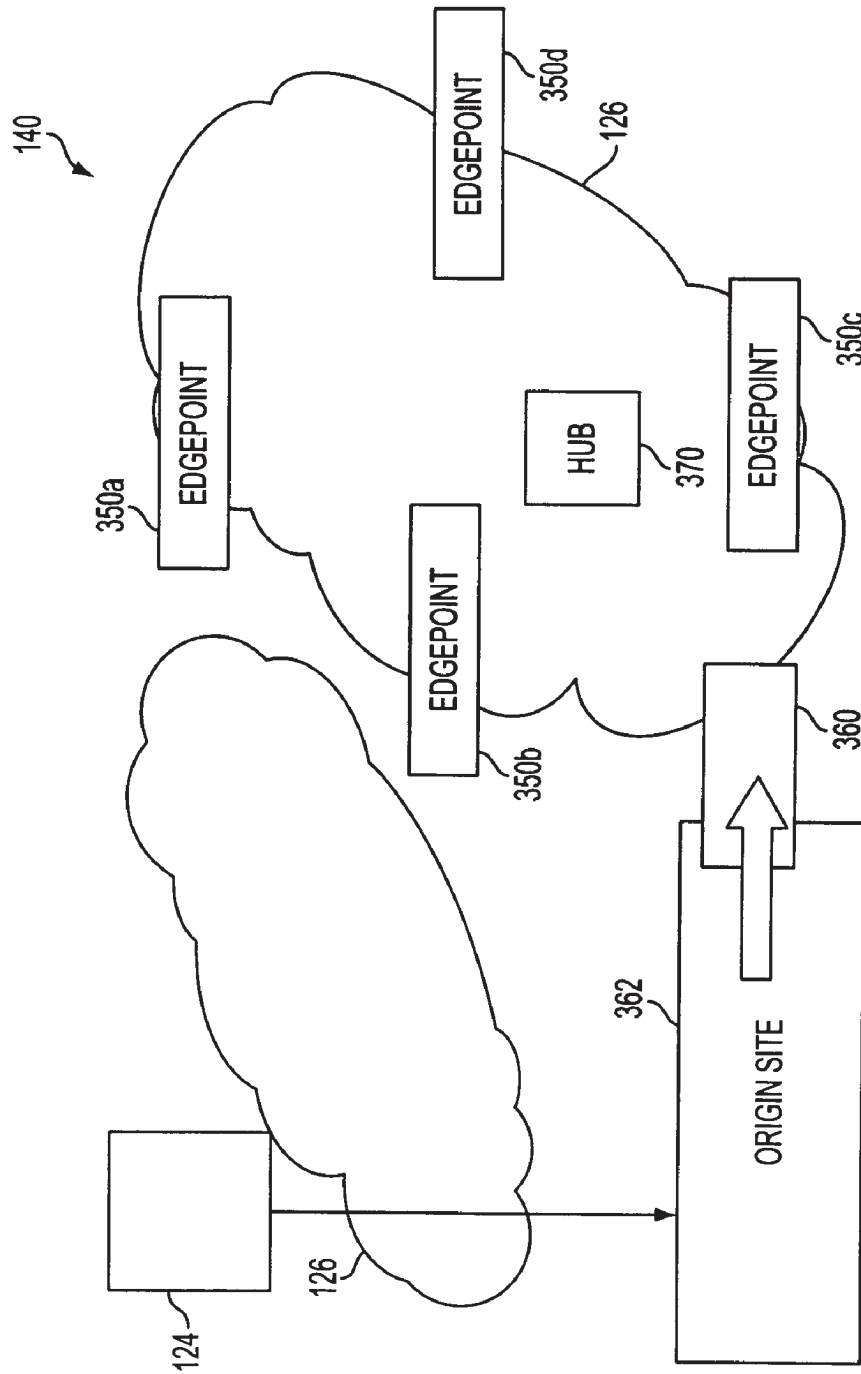


FIG. 14A



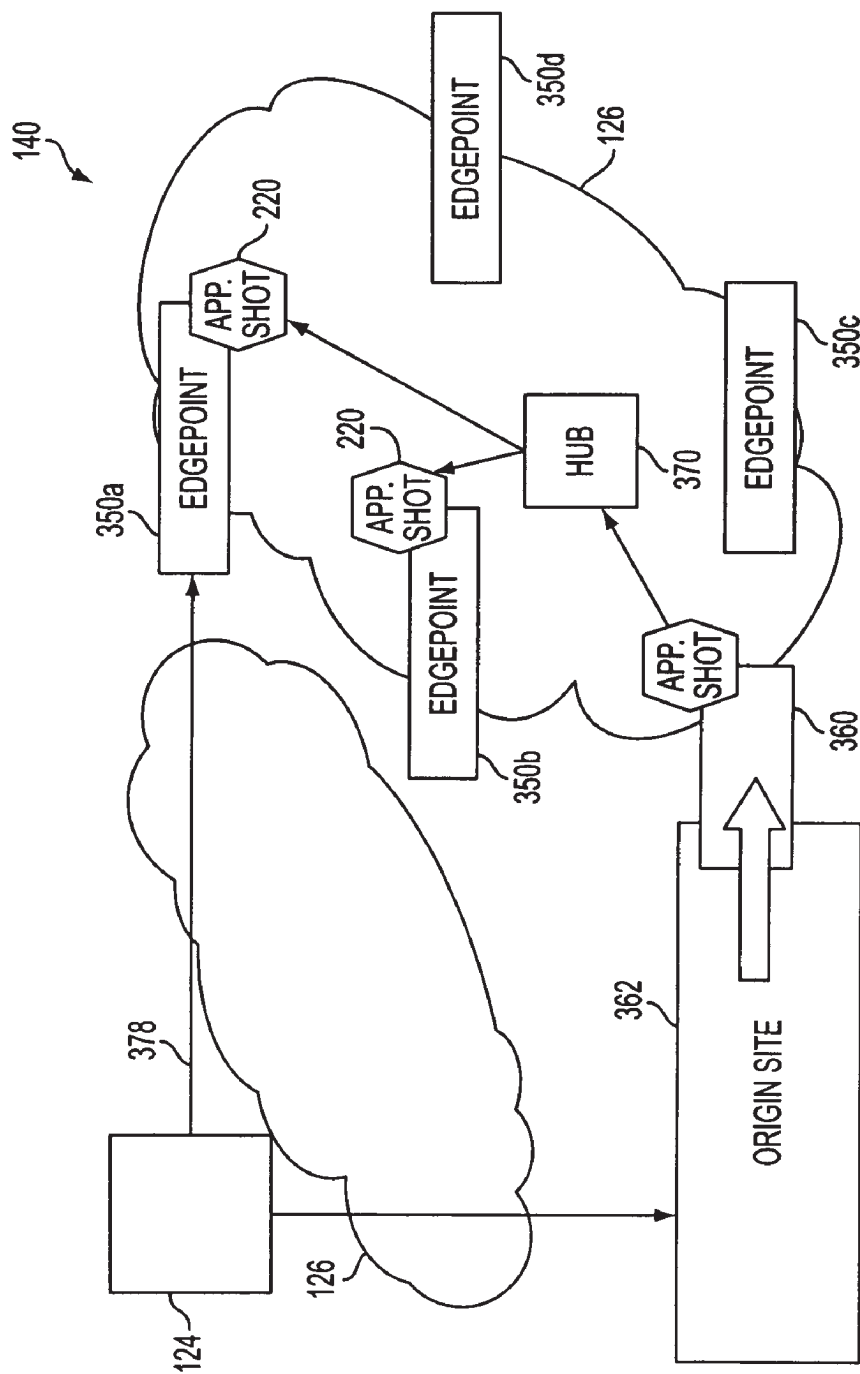


FIG. 14B

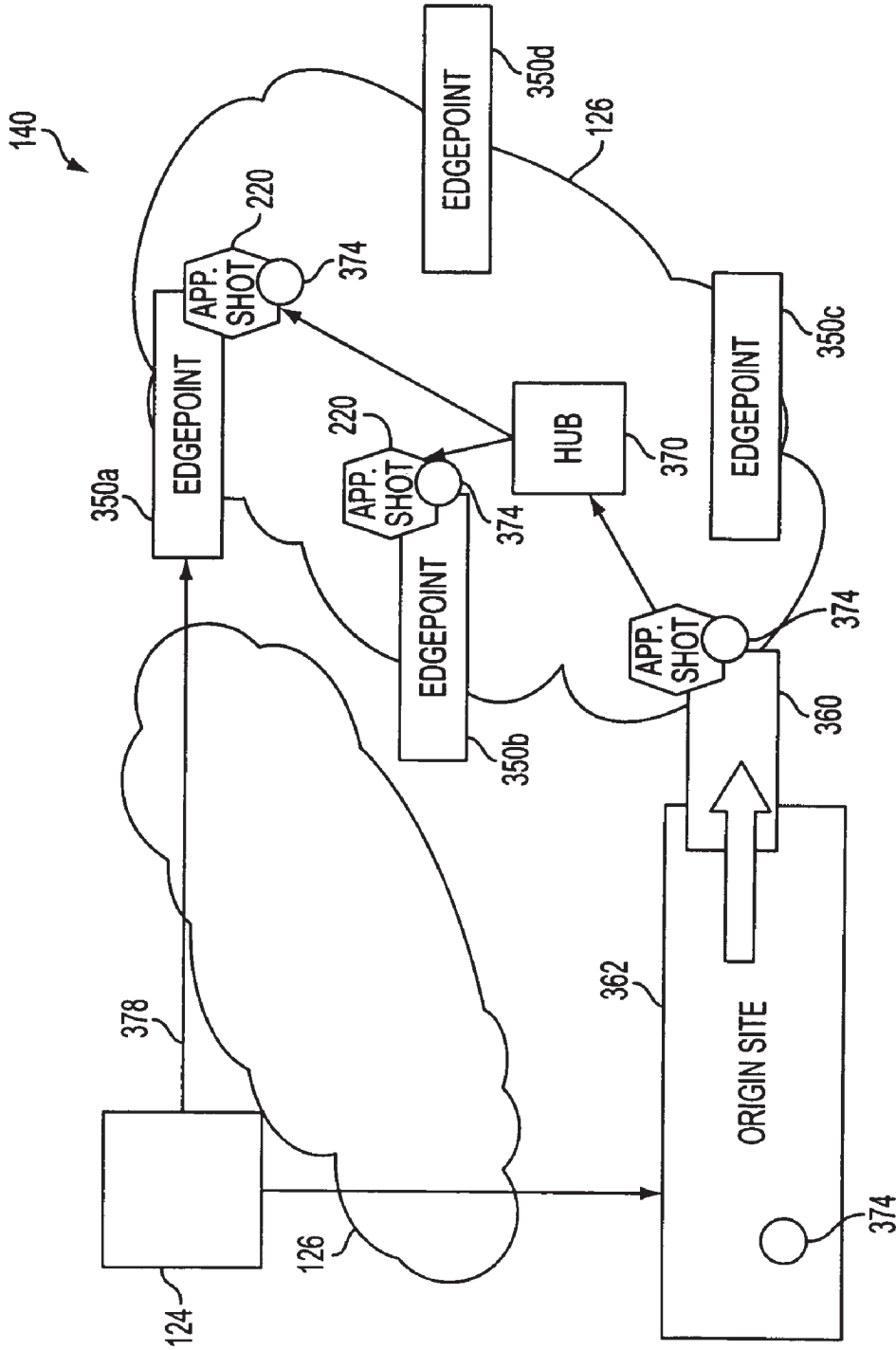


FIG. 14C

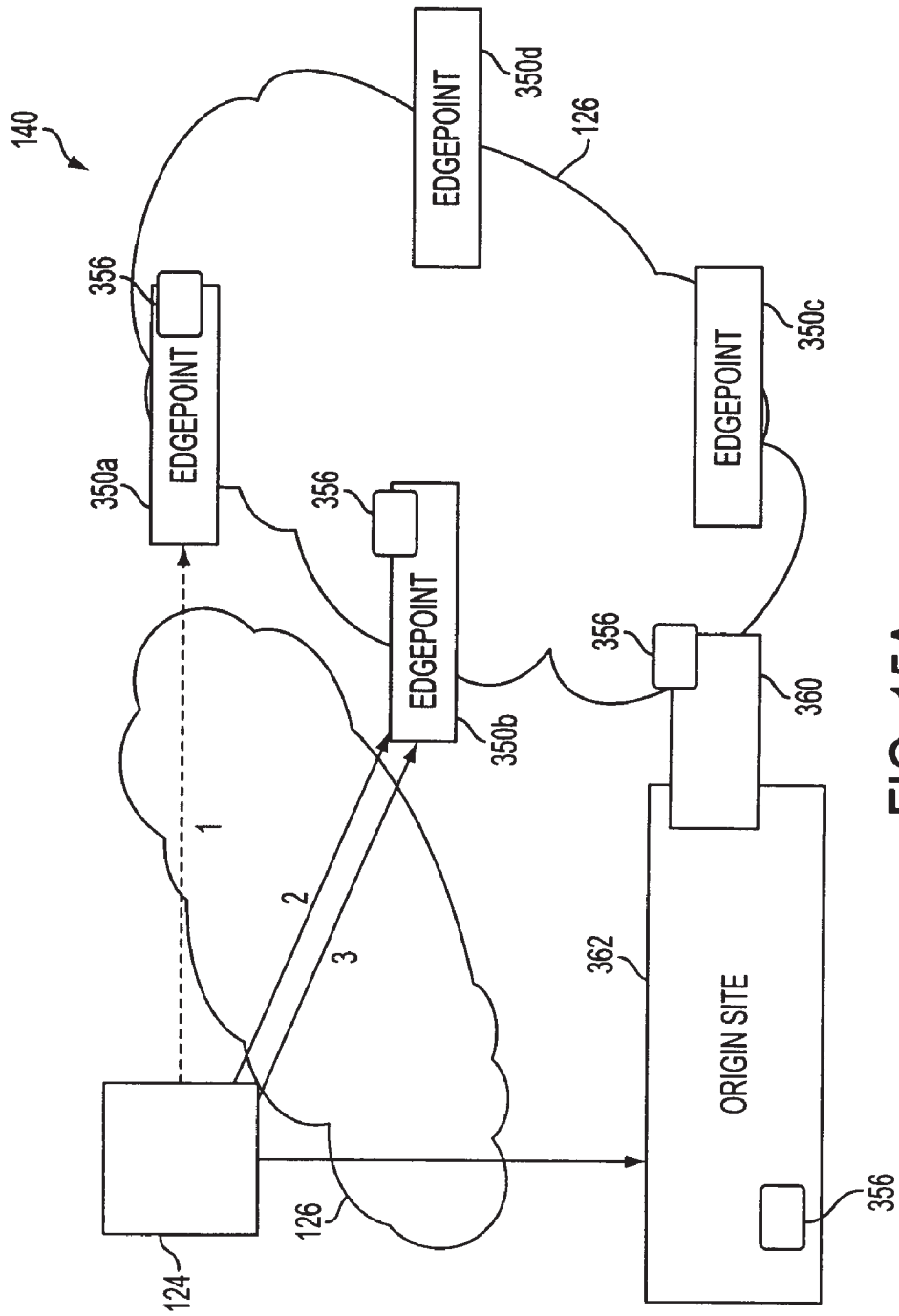


FIG. 15A

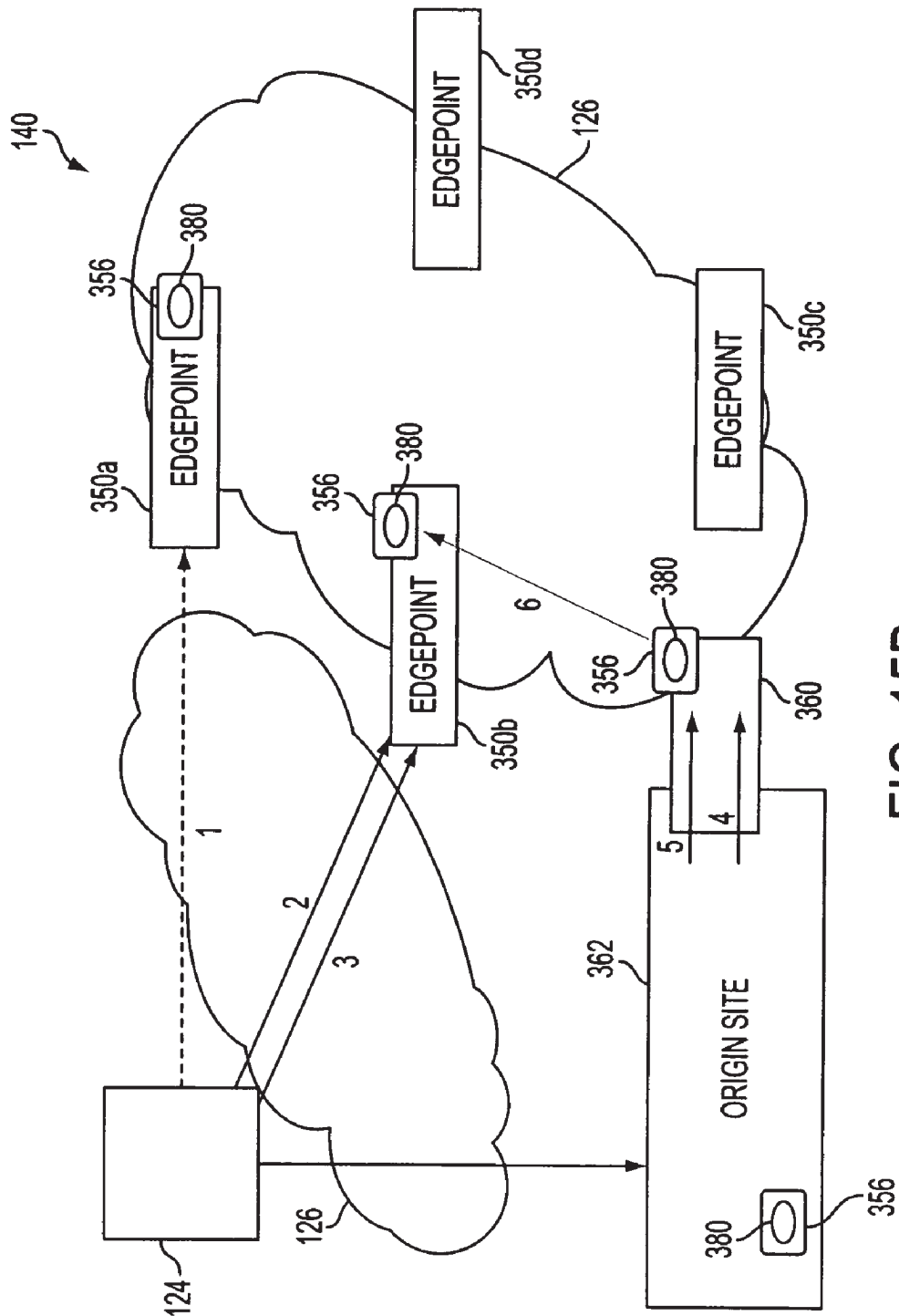


FIG. 15B

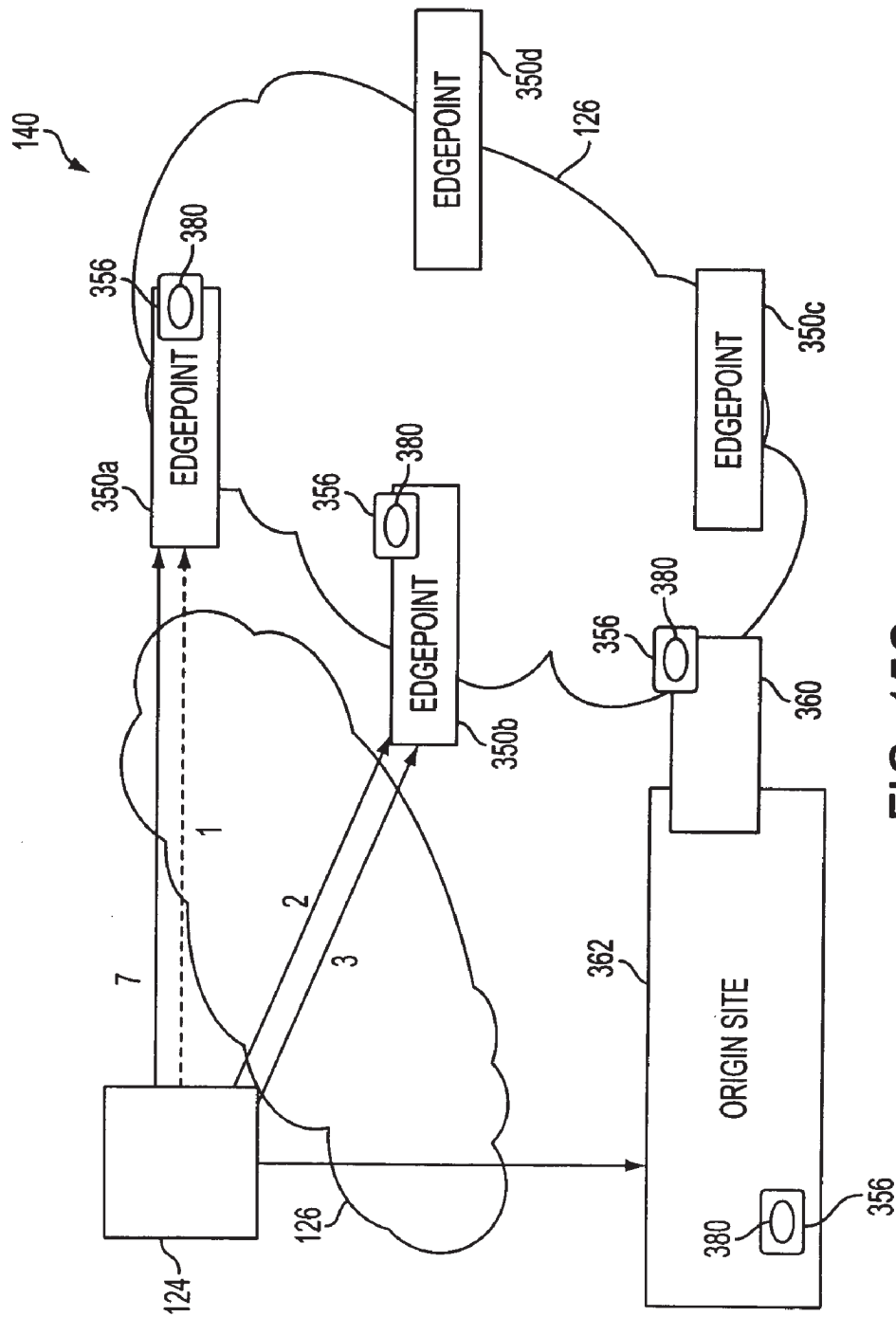


FIG. 15C

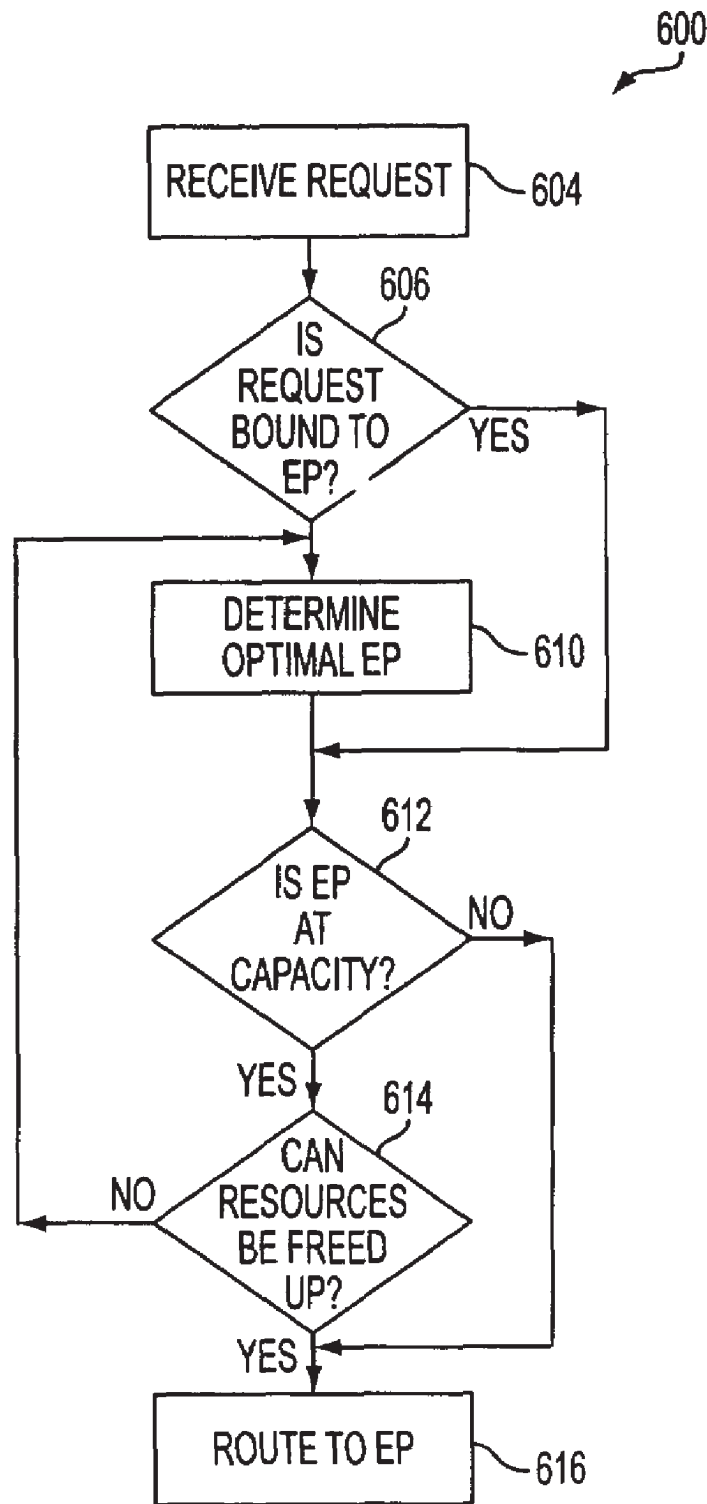


FIG. 16

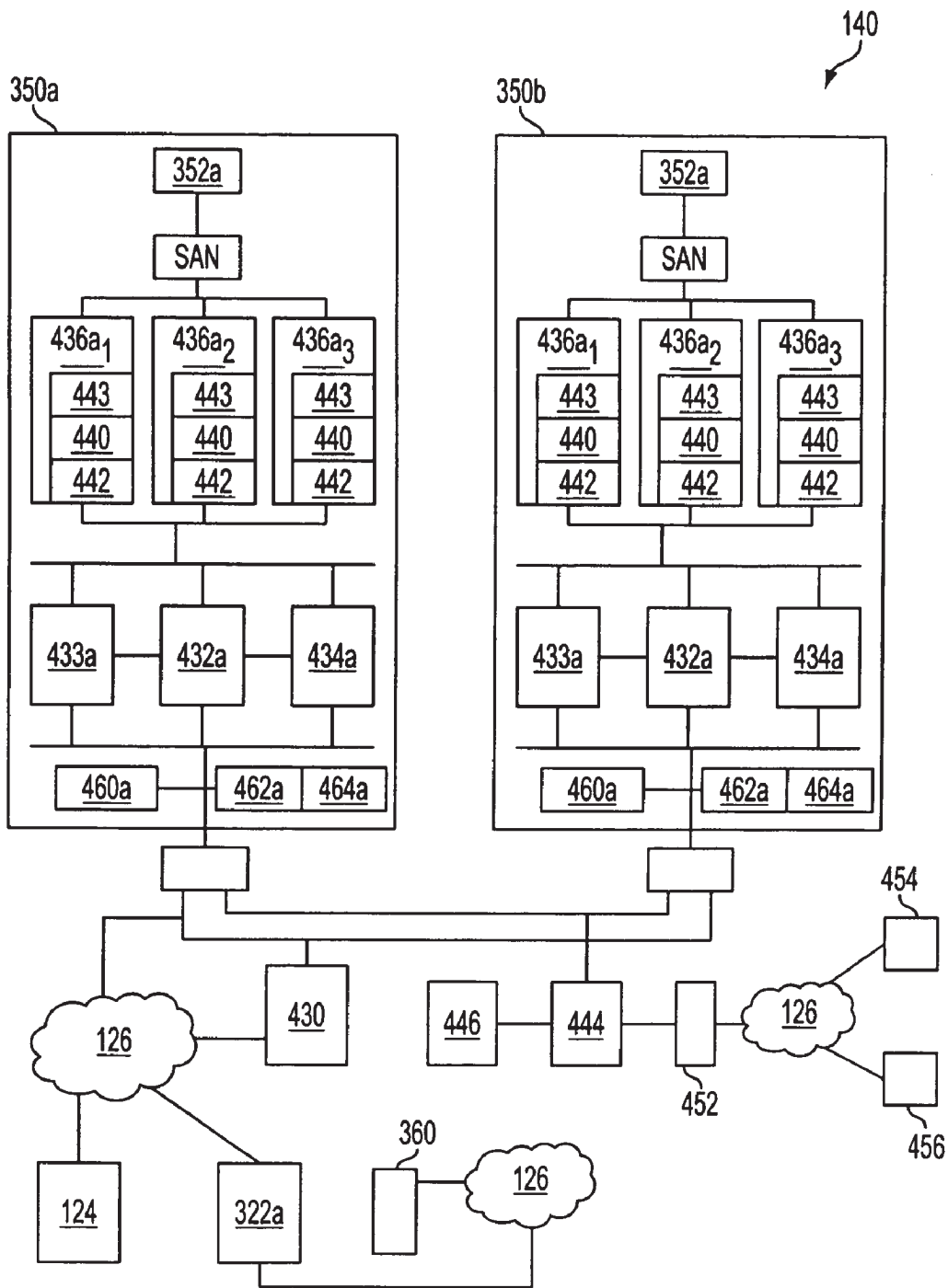


FIG. 17

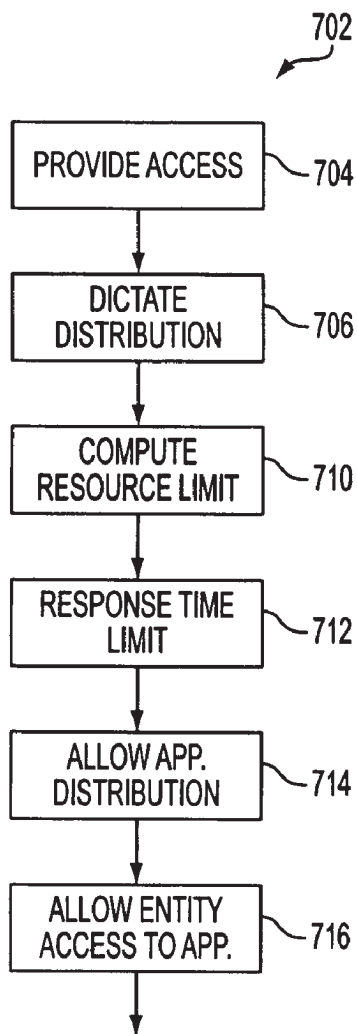


FIG. 18

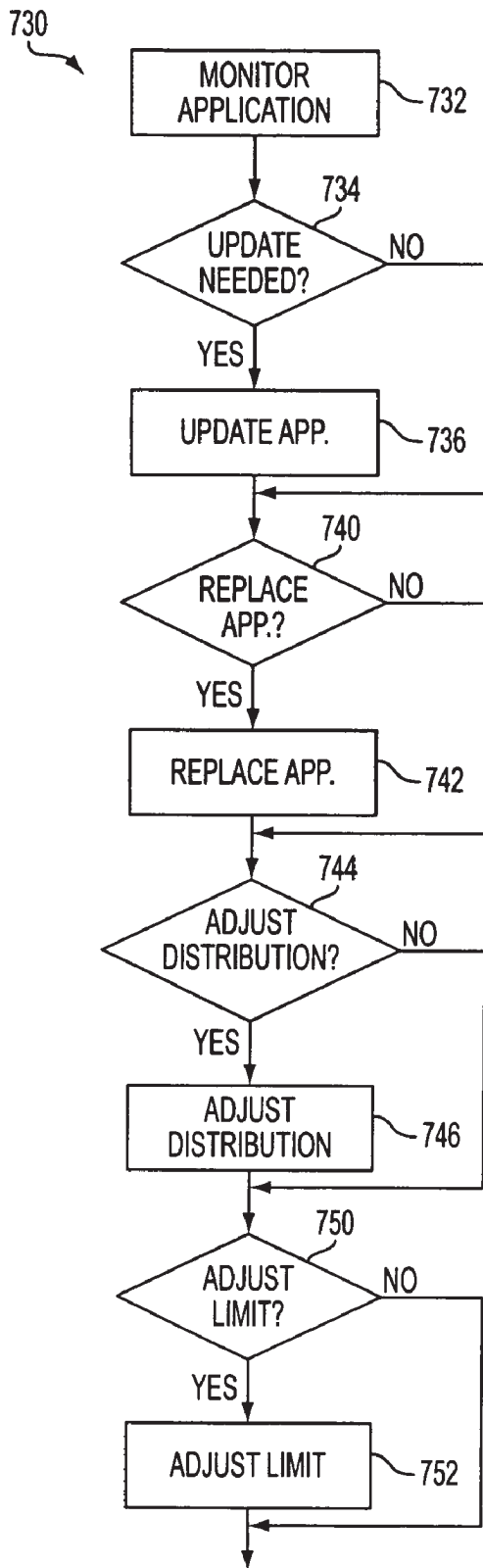


FIG. 19



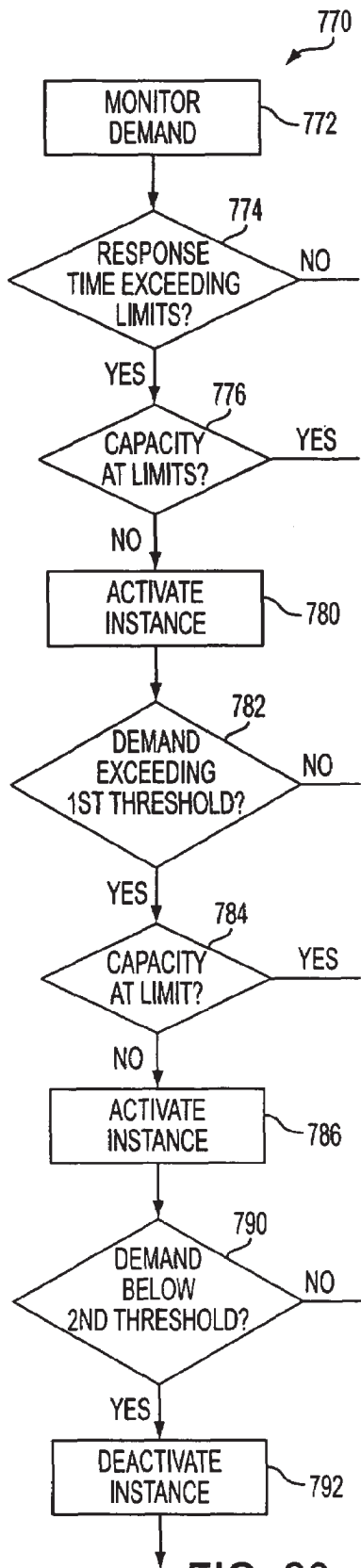


FIG. 20

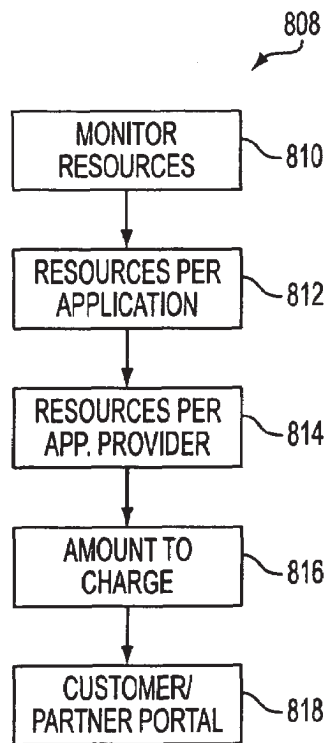


FIG. 21

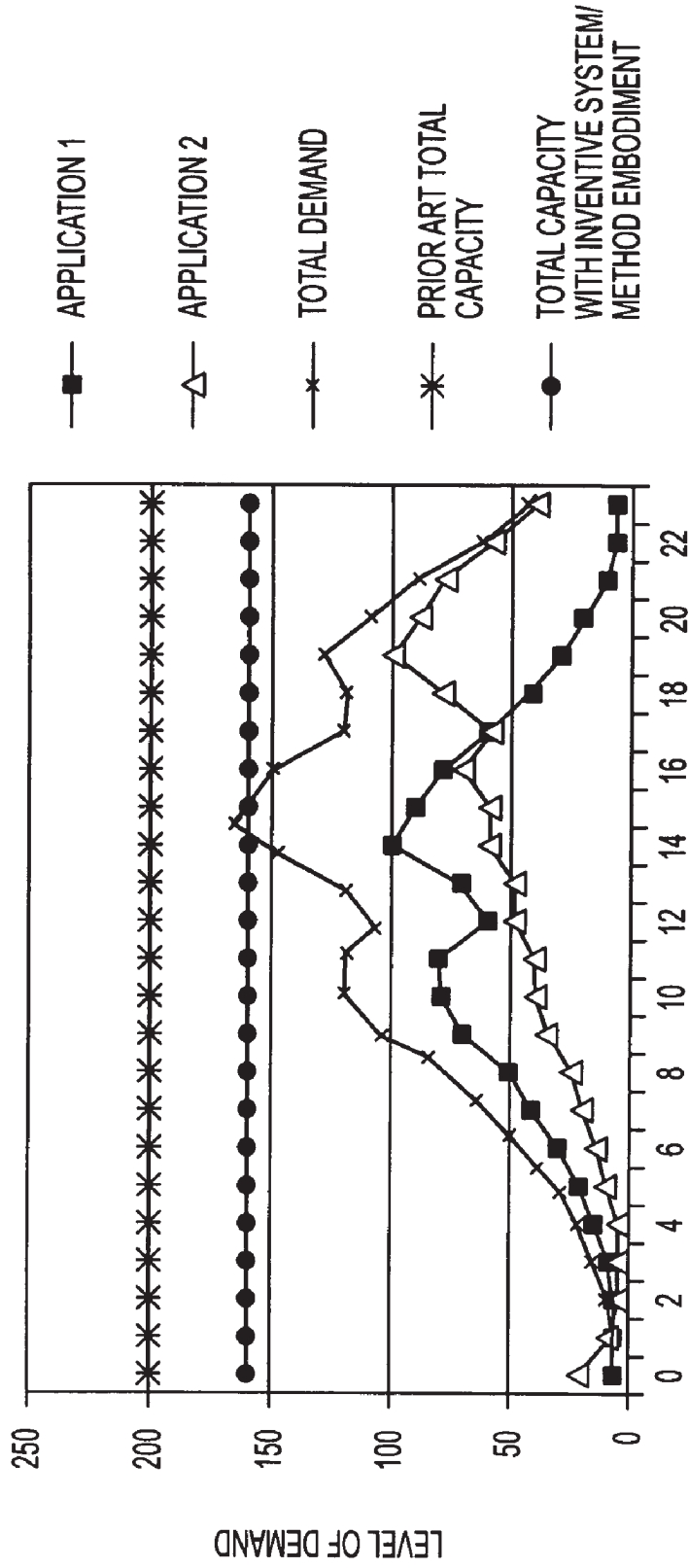


FIG. 22

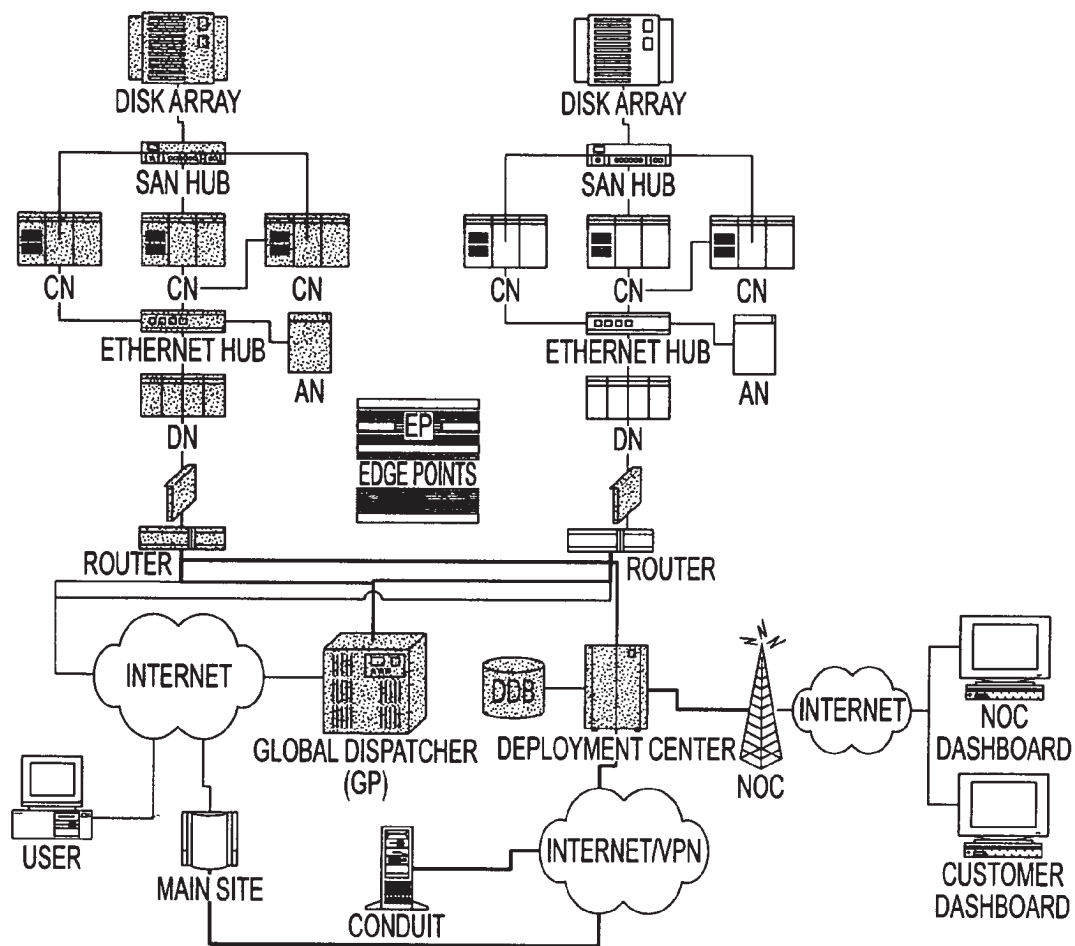


FIG. 23

## METHOD SYSTEM AND APPARATUS FOR PROVIDING PAY-PER-USE DISTRIBUTED COMPUTING RESOURCES

### RELATED APPLICATIONS

The present application claims priority to and incorporates the following applications in their entirety by reference:

A Method and Apparatus for Providing Pay-Per-Use, Distributed Computing Capacity, U.S. Provisional Application Serial No. 60/232,052, filed on Sep. 12, 2000;

Snapshot Virtual Templating, U.S. patent application Ser. No. 09/684,373, filed on Oct. 5, 2000;

Dynamic Symbolic Link Resolution, U.S. patent application Ser. No. 09/680,560, filed on Oct. 5, 2000;

Snapshot Restore of Application Chains and Applications, U.S. patent application Ser. No. 09/680,847, filed on Oct. 5, 2000;

Virtual Resource-ID Mapping, patent application Ser. No. 09/680,563, filed on Oct. 5, 2000; and

Virtual Port Multiplexing, patent application Ser. No. 09/684,457, filed on Oct. 5, 2000.

### FIELD OF INVENTION

In general the invention pertains to computer application processing, more particularly to distributed computing for computer application processing, and most particularly to system and method for providing computer application processing with dynamic capacity control and pay-per-use usage charging on an on-demand basis.

### BACKGROUND

There is a trend of emerging computing infrastructure aimed at on-demand services, particularly for Internet or other distributed networked computing services. There are basically three categories of on-demand services that are currently available. The first is content delivery, the second is storage, and the third is bandwidth. These services are provided as needed or on-demand, based on a user's needs at any given time. For example, if a first data provider needs greater storage space, an on-demand storage provider simply allocates a greater amount of storage memory to that user, and the first data provider is charged based on the amount of memory space used. If the first data provider no longer needs that amount of memory and deletes information, the on-demand storage provider is then able to re-allocate that memory space to an alternative data provider and the first data provider is charged less because of the reduced storage use.

One of the problems that companies with substantial IT investments face is that it is very difficult for them to predict how much demand they will have for their applications (capacity planning). Therefore, it is extremely difficult for them to determine how large a server farm to deploy which will allow greater user access to their services.

Another problem faced by application or website providers is the continued need for resource capacity to provide adequate service to their users. This is also referred to as the scalability problem. FIG. 1 shows a simplified block diagram representation of the diseconomy of scale resulting in the server infrastructure. What is seen is that application providers are in what is sometimes referred to as a high growth spiral. In the high growth spiral the application provider starts by building a service 52 to gain application users or customers 54. The increase in users results in an increase in the application providers server loads 56. This increased server load

causes an increase in response time and often results in the application provider's sites failing or going down, which may result in a loss 60 of users. The application provider must then invest in more resources and infrastructure 62 to reduce response time, improve reliability and keep their users happy 64. This increased response time, and reliability then attracts more users 54, which returns the application provider back to a point where the increased load demands stress or tax the application provider's servers 56, resulting again in a slower response time and a decrease in reliability. Thus, application providers are constantly going around in this high growth spiral.

FIG. 2 shows a graphical representation of the cost per user to increase resource capacity. One of the problems faced by application providers is that the cost of server infrastructure may typically increase faster than the current number of users so that costs are non-linear. This means that as the application provider's server farm gets more complex the cost delta 70 to add enough capacity to service one additional user increases. Thus, the cost 70 of continuing to grow increases dramatically in relation to the cost per user. With most every other business, as the business grows, economies of scale come into effect and the costs per user served actually decreases 72. This is one of the real problems faced by application providers.

Bottlenecks exist in various system resources, such as memory, disk I/O, processors and bandwidth. To scale infrastructure to handle higher levels of load requires increased levels of these resources, which in turn require space, power, management and monitoring systems, as well as people to maintain and operate the systems. As user load increases, so does complexity, leading to costs increasing at a faster rate than volume.

Another problem with providing application processing or services is the amount of capacity that will be needed at start-up, as well as the capacity needs in the future to maintain response time and reliability. These are both start-up costs. It is relatively impossible to predict in advance, with any degree of accuracy, just how successful a site or service is going to be prior to launching and activating the site.

FIG. 3 shows a graphical representation of user capacity demands of an application provider. When an application provider installs a certain number of servers, whatever that number is, the provider has basically created a fixed capacity 74, while demand itself may be unpredictable. Because of the unpredictability of usage demands on servers, that fixed capacity 74 will be either too high 76, and the application provider did not have as many users as anticipated resulting in wasted capacity 76 and wasted capital. Or the fixed capacity 74 was too low 80, and the application provider obtained more users than predicted, resulting in insufficient capacity 80. Thus, if the fixed capacity 74 is too high, the application provider has invested too much capital 76. If the fixed capacity 74 is too low 80, the application provider has users who are dissatisfied because the user does not get the service they need or it takes too long to get responses. This unpredictability is an extremely difficult problem faced by companies providing application processing and services and is particularly severe for those providing services over the Internet simply because of the dynamics of the Internet. The demand is completely unpredictable, and is substantially impossible to plan.

One problem faced by on-line application providers or other users of distributed computing networks is that the network is actually very slow for interactive services as a result of large traverses across the network, because communication signals run into the inherent latency of the network. For example, if an Internet user is in New York, but that New York user want to access a website that is serviced in Los

3

Angeles, the New York user must be routed or hopped all the way across the U.S. Sometimes users will be routed all the way around the world, to get to a specific site. These long distance routings run into large amounts of latency delay. This inherent latency of distributed networks is amplified by the significant increase in the number of interactive services deployed by application and website providers having very active pages or sites. Further, there is a general trend towards customized pages per user. These are sites which are custom created by the server or application for a particular user. These customized sites reduce caching effects to substantially zero. Thus, a customized page, created for that specific user, is generated at the server origin site and routed all the way back across the net to the user adding further inherent delays in the response time. This adds up to a very slow service for more complex interactive services.

In prior art systems, application providers wishing to provide applications have to buy or lease a server, then they must buy or develop the applications that are going to be loaded and run on that server, load the server, and activate the server to provide access to that application. The server is a fully dedicated resource, so that 100% of the time an application is dedicated to a specific server.

Prior art application processing systems require an application provider to route a user to a single central site to allow access to the applications. Every user attempting to access the application is directed to the single central site. Thus, resulting in a bottle neck at the central site. In the prior art single server or single central site, the application provider, however, does maintain access to and control over the application. In some systems where the application provider outsources their server capacity, the application provider must select from a preselected limited number of applications. Further, the application provider no longer has direct control over the application. Any changes desired require the application provider to submit a request to the server provider. Then the server provider must schedule a time at low demands to take the server down to make the changes. This process results in large lag times between the decision to make changes and the implementation of those changes.

### SUMMARY

The novel method, apparatus, computer readable medium and computer program product of the present invention provides on-demand, scalable computational resources to application providers over a distributed network and system. The resources are made available upon receiving requests for a first application. Once a request is received, routing of the request is determined and the request is routed to access the first application. The application provider is then charged based on the amount of resources utilized to satisfy the request. In determining routing the method and apparatus determines if a first instance of a first application is active, and if the first instance is at a capacity. A first set of compute resources is provided to satisfy the first request and the amount charged to the first application provider is increased based on the first set of compute resources. In one embodiment, the method and apparatus activates a second instance of the first application on a second set of the available compute resources if the first instance is at capacity and the amount charged to the first application provider is increased based on the second set of compute resources. As a result, resources needed are dynamically available on demand, and freed when not needed. The application provider is only charged for services that are actually used.

4

In one embodiment, a third set of compute resources are freed up if the compute resources are not available. A second instance of the first application is restored on a fourth set of compute resources such that the fourth set of compute resources includes at least a portion of the freed up third set of compute resources, and the amount charged to the first application provider is increased based on the fourth set of compute resources. In freeing up resources, a first instance of a second application is snapshotted, wherein the second application is provided by a second application provider, and an amount charged to the second application provider is reduced based on the freed up third set of compute resources.

The method and apparatus provides application providers with access to the network, where the network includes the distributed compute resources configured to provide the application processing and allows the application providers to distribute applications onto the network to utilize the distributed compute resources for processing of the applications. The application providers are further capable of monitoring, updating and replacing the distributed applications. The method and apparatus increases the amount of compute resources utilized in providing processing for an application as demand for the application increases. As the amount of compute resources is increased the amount charged to the application provider is increased based on the amount of compute resources utilized. As demand for the application falls, the amount of resources is reduced and the amount charged the application provider is reduced.

In one embodiment, the apparatus for providing the on-demand compute resources includes a first resource manager, at least one snapd (snapshot or snapshot daemon) module configured to snapshot an active application, at least one restored (restore daemon) module configured to restore a snapshotted application, and a first set of compute resources configured to provide application processing. The resource manager couples with and provide at least some control to the snapd module, restored module and the first set of compute resources. The resource manager is further configured to monitor the amount of the first set of compute resources utilized in providing application processing. In one embodiment, the apparatus includes at least one perfd (performance or performance daemon) module coupled with the first resource manager and the first set of compute resources, and is configured to monitor the first set of computational resources and provide the resource manager with compute resource utilization. In one embodiment, a deploy module couples with the first resource manager and the first set of compute resources, and is configured to receive at least one application from at least one of the application providers, and provision the first set of compute resources to be utilized in processing the at least one application. A conduit couples with the deploy module, and is configured to provide the application providers with access to the deploy module to distribute applications or updates for application processing. A local dispatcher couples with the first resource manager and the first set of compute resources, and is configured to receive directions from the resource manager and to provide routing of requests for the at least one application to the first set of compute resources. In one embodiment, the resource manager, snapd module, restored module, perfd module, local dispatch module and deploy module are cooperated into a single edgepoint. In one embodiment, the apparatus includes a plurality of edgepoints distributed to provide the on-demand, distributed compute resources.

In one embodiment, the apparatus includes a plurality of sets of compute resources and a plurality of resource managers, such that the sets of compute resources are utilized for

5

application processing. Further, a global dispatcher coupled with the plurality of resource managers, wherein the global dispatcher is configured to receive requests for at least one application and to route the requests to an optimal resource manager. In one embodiment, the apparatus includes one or more compute modules which comprise at least a snapd module, a restored module and at least a third set of compute resources.

In one embodiment the novel network providing on-demand compute resources includes a first means for application processing configured to provide application processing, a first application distributed onto the network and configured to be processed by the first means for application processing, a first means for managing application processing coupled with the first means for application processing, and configured to activate at least a first instance of the first application on a first set of the first means for application processing based on a first amount of demand for the first application. The network further includes a means for monitoring coupled with the first means for application processing, and configured to monitor at least the first set of the first means for application processing utilized to provide the entity with access to the first instances of the first application, and a means for determining an amount to charge coupled with the first means for application processing, and configured to determine an amount to be charged based on the first set of the first means for application processing utilized in providing the entity with access to the first instance of the first application. The means for managing application processing is further configured to activate a second instance of the first application on a second set of the first means for application processing based on a second amount of demand for the first application. The means for monitoring is further configured to monitor the second set of the first means for application processing utilized to satisfy the second amount of demand for the first application, and the means for determining an amount to charge is configured to determine an amount to be charged based on the second set of the first means for application processing utilized in providing access to the second instance of the first application. The means for managing application processing is further capable of deactivating one of the first and second instances of the first application based on a third amount of demand for the first application. In one embodiment, the method and apparatus includes a plurality of means for application processing, and a means for dispatching coupled with the plurality of means for application processing. The means for dispatching is configured to route at least one entity to an optimal means for application processing allowing the at least one entity access to at least one application. In one embodiment means for application processing is an edgepoint. In one embodiment, the means for dispatching is a global dispatcher. In one embodiment, the means for application processing is a compute module.

In one embodiment, the system, method, and business operating model provide a computer application processing capacity as a pay-per-use utility on demand.

#### BRIEF DESCRIPTION OF THE FIGURES

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

FIG. 1 shows a simplified block diagram representation of the diseconomy of scale resulting from the server infrastructure;

6

FIG. 2 shows a graphical representation of the cost per user to increase resource capacity;

FIG. 3 shows a graphical representation of user capacity demands of an application provider;

FIG. 4 shows a graphical representation of the on-demand response of the present on-demand system;

FIG. 5 depicts a simplified block diagram of a business operating over the Internet, sometimes referred to as an e-business;

FIG. 6 depicts a simplified schematic block diagram of one embodiment of the novel distributed on-demand application processing system which substantially eliminates the bottleneck and tornado effects seen in the prior art;

FIG. 7 illustrates in high level block diagram form one implementation of one embodiment of the overall structure of the present invention as used in connection with a computer network such as the internet;

FIG. 8 depicts a block diagram of one embodiment of a computer for implementing the on-demand method and apparatus of the present invention in a computer readable medium;

FIG. 9 shows a simplified block diagram of one embodiment of an overall system architecture for the distributed, on-demand application processing service and system of the present invention;

FIG. 10 shows a simplified block diagram of one embodiment of the application switching architecture;

FIG. 11 depicts a simplified flow diagram of one implementation of a sequence of steps executed by the present invention to perform a snapshot of a process or application instance;

FIG. 12 illustrates a simplified flow diagram of one implementation of the sequence of steps executed to restore a snapshotted application;

FIGS. 13A-C shows a simplified block diagram of one embodiment of an edgepoint of the present invention;

FIGS. 14A-C show simplified block diagrams of embodiments of the present on-demand application processing system in cooperation with the preexisting internet infrastructure;

FIGS. 15A-C show a simplified block diagram of one implementation of one embodiment of the novel on-demand apparatus and the optimal user and entity routing provided by the present invention;

FIG. 16 shows a simplified flow diagram of one implementation of one embodiment of the method and system providing on-demand compute resources;

FIG. 17 shows a simplified block diagram of one implementation of one embodiment of a novel on-demand apparatus including a plurality of edgepoints;

FIG. 18 depicts a simplified flow diagram of a process for an application provider to access and distribute applications onto the distributed, application processing system of the present invention;

FIG. 19 depicts a simplified flow diagram of one embodiment of a process for an application provider to monitor and update applications distributed onto the system;

FIG. 20 depicts a flow diagram of one embodiment of a process for monitoring demand and determining an amount to bill an application provider;

FIG. 21 depicts a simplified flow diagram of one implementation of one embodiment of a process for determining an amount of resources utilized for an application and the amount to be charged to the application provider based on the amount of resources utilized;

FIG. 22 depicts typical exemplary demand situation for two different applications (or customers) across a twenty-four hour time period; and

FIG. 23 is a diagrammatic illustration showing an embodiment of a system according to the invention.

#### DETAILED DESCRIPTION

Among other aspects and innovations, the invention provides structure, system, method, method of operation, computer program product, and business model and method for providing distributed on-demand application processing.

There is a missing category in the available internet infrastructure based on-demand services. On-demand services fail to provide on-demand application processing, delivered as an on demand infrastructure service.

In one embodiment, the present invention provides for on-demand application processing, delivered as an on demand internet (or other networked) infrastructure service. Application processing may for example include one or more of, but is not limited to, deploying, instantiating, running and operating an application. One of the major benefits of providing this type of on-demand service is improvement in operational and other economics. The novel on-demand application processing method and system of the present invention improves: operational economics such as the elimination of costly server infrastructure expansion, simplifying and reducing capacity planning and an economic cost based on use; and user satisfaction by providing a maximum and assured application, such as an internet site, responsiveness under substantially any user load and for users located substantially anywhere. The present inventive on-demand application processing method and system changes the economic focus of server infrastructure.

The novel on-demand application processing method and apparatus of the present invention solves an application provider's capacity planning problem. For example, an application provider is an entity that provides a service via a computer network such as, Charles Schwab, WebVan-like entities, Walmart.com, and Intuit, which provide various types of applications accessed by individuals or entities over the internet. One of the problems that such companies face is that it is very difficult for them to predict how much demand they will have for their services and applications. Therefore it is extremely difficult for them to determine how large a server farm to deploy to allow greater user access to their services.

The present on-demand application processing method and apparatus solves this problem by providing on-demand processing capacity. Thus, the on-demand system provides the application provider with additional access to further processing capabilities without the need or expense of the application provider trying to predict how much processing capability will be needed. Further, one of the advantages of the present on-demand application processing method and system is that the application provider's cost is based on the amount of processing capability actually used. Thus, instead of having a huge up front capital investment to provide the expected processing capabilities and thus take all the risk to create these services, the present on-demand application processing method and system provides the application processing capacity based on demand, avoiding the need to predict processing needs, and eliminating the up-front capital investment.

Another major benefit provided by the novel on-demand application processing method and system is application user or customer satisfaction. An application user's satisfaction is achieved and maintained because the on-demand application processing substantially improves the response time of applications by increasing the processing capacity as the demand increases, is capable of spreading the load across a plurality

servers, and enhancing consistency. The on-demand application processing system is further able to put a cap or limit on how much response time is built into the server side of application processing.

The present on-demand application processing method and system solves the growth spiral and the exponential cost per user increase in providing applications and services over the internet by supplying resource capacity based on the demand for the applications. The present on-demand method and system will increase resource capacity to an application provider as user access grows, and will also reduce resource capacity as user access decreases. Thus, the application provider simply pays for the amount of resources needed and used.

The present invention provides an ideal solution to the fixed capacity problem shown in FIG. 3, through a flexible on-demand variable capacity providing substantially unlimited server infrastructure capacity which responds within seconds because demand patterns change within seconds. FIG. 4 shows a graphical representation of the on-demand response of the present on-demand system. The present on-demand application processing system solves the unpredictable capacity problem by providing on-demand server or application processor capabilities with a response time of seconds or less. If the capacity demand increases, the on-demand capacity of the present invention adjusts to supply further capacity 90a and 90b. If the capacity demand decreases, the on-demand capacity of the present invention adjusted to supply less capacity 92a and 92b, thus reducing the overall cost.

The present invention provides an ideal solution, by providing substantially instant variable capacity. As an example, the present invention provides an infrastructure or virtual infrastructure, which comes on-line or is activated for those peak times (i.e., those 10 minutes) when a site gets a rush of Web traffic, and then the virtual infrastructure reduces or goes away when the Web traffic is reduced. Further the present invention provides substantially unlimited processing resources, thus providing as much processing as is needed. The present invention further provides unlimited processing resources with a global reach, because application providers now have users all over the world. The present invention further provides this substantially unlimited capacity to application providers at a pricing scheme which charges the application providers for the amount of capacity utilized, obviating the need for capital expenditures. The present on-demand application processing method and system is flexible and capable of running substantially any application, thus the application providers are not limited or locked into a particular application. The present invention provides the application providers with the ability to have the freedom to choose their own application sets. Further, the present invention allows the application sets to be completely under the application provider's control. As an example, once an application provider deploys an application set, the application provider maintains control over that application set, the data in and related to the application set, and other such control features. Thus preventing an application provider from being at the mercy of someone else owning their application set. Instead, the application provider maintains complete control over the services provided through the distributed application set.

FIG. 5 depicts a simplified block diagram of a business operating over the Internet, sometimes referred to as an e-business. Generally, an e-business has a set of servers 110 that run several or all of their different applications. The servers 110 have back end ERPs 112, back end transaction systems or services 114, and front end systems 116 including, but not limited to, personalization service 118, an e-selling

system 120, and a one-to-one marketing system 122, which is found in a central site. Users 124 gain access through the internet 126 to the central server or central site 110. As the number of users 124 accessing the server 110 increases, a tornado effect 130 results, and a bottleneck 132 is created, adversely affecting the response time and reliability of the site.

FIG. 6 depicts a simplified schematic block diagram of one embodiment of the novel distributed on-demand application processing system 140 of the present invention which substantially eliminates the bottleneck and tornado effects seen in the prior art. In one embodiment, the present on-demand system 140 pushes or distributes application processes, such as the front end systems, including, but not limited to, personalization 118, eSales 120, and one-to-one marketing 122 (see FIG. 5), out into the Internet 126, and out into the infrastructure of the Internet. In one embodiment, distributed compute resources, such as processors, computers and/or servers 148, of the on-demand system 140 are geographically distributed, and in one embodiment located and installed globally all around the world. By geographically distributing the servers 148, the application processing can also be distributed, allowing traffic from users 124 to be distributed and routed to the servers 148 distributed across the Internet 126. In one embodiment, final applications or transactions, such as final purchases, and any other conventional transaction, are routed back across the internet 126 to the transactions system 114 and the ERP system 112. In one embodiment, the transaction system 114 and the ERP system 112 are not moved out or distributed across the distributed servers 148. As an example, a user 124 does his/her shopping and configuring, and the like as well as determining what he/she would like to buy, through a distributed server 148 which is geographically closer to the user in the on-demand system 140. In one embodiment, once the user 124 selects or hits the "buy" button of the interactive website to complete the sales transaction, that transaction is forwarded to the backend systems 112 and 114 maintained on the application provider's central servers to complete the transaction. Thus, significantly reducing the amount of traffic into the application provider's central servers, eliminating the bottle neck effect, improving performance, enhancing response time, and thus improving and maintaining user satisfaction.

In one embodiment, the entire central site including the back end ERP 112 and transactions service 114 are distributed out onto the distributed on-demand system 140. Thus, even the final transactions, such as the final purchase, are performed on the distributed servers 148.

FIG. 7 illustrates in high level block diagram form one implementation of one embodiment of the overall structure of the present invention as used in connection with a computer network 150 such as the Internet. In one embodiment, computer network 150 is a direct link between one or more remote entities, such as the users 152-1 and 152-2, a separate application, a server, a process, computational device and substantially any other entity capable of issuing requests for application processing. In one embodiment, computer network 150 is a network providing an indirect link, such as an intranet or global network (i.e., the Internet). Remote users 152-1 and 152-2 utilize the computer network 150 to gain access to a plurality of computers or servers 158-1, 158-2, through 158-n. In one embodiment, the computers 158 are protected by a firewall 154. In one embodiment, computers 158 are edgepoints (described more fully below), groups of edgepoints, global dispatchers or other components of a private network 156. In one embodiment, computers 158 are used to run various applications, such as hosting web sites for access by

remote users 152. In one embodiment, the present invention is implemented on computer network 156 in the form of virtual environments 160-1 and 160-2. While only two virtual environments are illustrated, it is to be understood that any number of virtual environments may be utilized in connection with the present invention

In one embodiment, the method and system of the present invention is implemented in a computer readable medium, such as a computer program 164 and executed on a computer 166 as illustrated in the high level block diagram of FIG. 8. As shown, computer 166 incorporates a processor 168 utilizing, in one embodiment, a central processing unit (CPU) and supporting integrated circuitry. A memory 170 which is any type or combination of memory including fast semiconductor memory (for example, RAM, NVRAM or ROM), slower magnetic memory (for example, hard disk storage), optical memory and substantially any conventional memory known in the art, to facilitate storage of the computer program 164 and the operating system software. In one embodiment, also included in computer 166 are interface devices including, but not limited to, keyboard 172, pointing device 174, and monitor 176, which allow a user to interact with computer 166. Mass storage devices such as disk drive 178 and CD ROM 180 may also be included in computer 166 to provide storage of information. Computer 166 may communicate with other computers and/or networks via modem 182 and telephone line 184 to allow for remote operation, or to utilize files stored at different locations. Other media may also be used in place of modem 182 and telephone line 184, such as a direct connection, high speed data line or a wireless connection, and the like. In one embodiment, the components described above may be operatively connected by a communications bus 186. In one embodiment, the components may be operatively connected by wireless communication.

FIG. 9 shows a simplified block diagram of one embodiment of an overall system architecture 200 for the distributed on-demand application processing service and system 140. The on-demand system 140 includes an application switching architecture or technology 202 configured to provide application switching, an edge processing network 204, which, in one embodiment, is hundreds of machines, edgepoints or servers in hundreds of data centers distributed throughout the world and/or the internet, automated deployment 206, remote control 210, security architecture 212, and performance monitoring 214, all coupled to cooperate and provide application processing, and deployment.

Some of the advantages provided by the on-demand method and system 140 include: protection during peak loads, in one embodiment, with guaranteed application response time SLA; global reach with application provider control of distributed web presence; freedom to grow aggressively including elastic web-processing infrastructure on demand; no capital investment with costs based on the amount of capacity used; supporting substantially any application on substantially any platform to preserve application provider's current application investment; and higher reliability because the system provides superior response time and automatically routes around failures.

FIG. 10 shows a simplified block diagram of one embodiment of the application switching architecture 202. In one embodiment, the application switching architecture 202 includes an application snapshot or appshot 220. An appshot 220 is a set of all data and/or state necessary to halt (and then restore and restart) at least one application at a given point in time, such that, the application may be restored at a later time on substantially any machine. For example, an appshot 220 can be an already running application halted at a point in time



without the application knowing it was halted. In one embodiment, an appshot 220 is the encapsulation of an application stack of at least one running application including the different processes, states, and interprocess communication. For example, a set of interdependent and/or interacting applications halted together may be included in an appshot 220. In one embodiment, the appshot 220 includes, data 222, and a set of interactive applications, 224a-224n.

One example of an appshot 220 is a configuration engine, which allows users to shop online and decide exactly what they want to purchase. A snapshotted application and/or process, and the method for performing a snapshot is more fully described in co-pending U.S. patent application Ser. No. 09/680,847, filed on Oct. 5, 2000, incorporated in its entirety herein by reference.

In one embodiment, an appshot 220 encapsulates a multi-tier applications stack, including data 222. The present on-demand application processing method and system 140 performs this appshot encapsulation or snapshotting which saves the states of a running set of processes. The encapsulation of an appshot 220 allows the on-demand system 140 to replicate an application and provide a plurality of instances of the same application to be operated at substantially the same time utilizing a plurality of subsets of the on-demand computational resources. The replication allows the on-demand system 140, among other things, to move the appshot 220 to another set of compute resources such as another server, computer or machine, to duplicate the appshot 220 to other servers, and to replace or upgrade an appshot 220. Further, the encapsulated appshot 220 allows the on-demand system 140 to put an application when operating as an instance of an application into a form which allows the system to remove the instance of the application from an idle server when the application instance associated with an appshot 220 is not being used, and to store the appshot 220 in a memory with accompanying application states. As an example, an appshot 220 is an already running application halted at a point in time. Thus the on-demand system is capable of freeing up resources to allow other applications to utilize the idle resources.

In one embodiment, the on-demand application system 140 is capable of relocating or replicating an appshot 220 to other or alternate sets of computational resources such as other compute modules and/or other edgepoints 350 (see FIG. 14A) distributed throughout the worldwide on-demand system 140 providing at least a portion of the distributed on demand computational resources. In one embodiment, an edgepoint 350 is a computing facility with intelligent routing and load balancing architecture or capabilities. The edgepoint is capable of operating as a server. An edgepoint includes at least one and usually a plurality of servers or processors, such as a Sun Server 420 available from Sun Microsystems, Windows NT server from Microsoft, and a Linux server available from Linux, and substantially any other conventional server known in the art. The edgepoints are deployed, in one embodiment, throughout the world making up at least a portion of the on-demand system 140. Application providers will generate an appshot 220 of the application, applications or site which they want to distribute throughout the on-demand system 140. The appshot 220 can then be distributed to specific edgepoints, or distributed globally to every edgepoint 350 of the on-demand system 140. Thus, when an entity 124, such as a user, a separate application, a server, a process, computational device and substantially any other entity capable of issuing requests for application processing, wants to access the application or site, the edgepoint 350 activates or restores the appshot 220 to an activate instance of the appli-

cation or applications encapsulated within the appshot 220. The configuration and structure of the appshot 220 also allows the edgepoint 350 to re-encapsulate or snapshot the application or applications back into an appshot 220 and store the appshot 220 in a memory when the application or applications are not in use. As discussed above, the appshot 220 is capable of being restored or reactivated when needed. In one embodiment, the application can be restored from an appshot 220, usually in less than 5 seconds, and more usually less than 3 seconds, depending on the available edgepoint resources. Thus, in one embodiment, the on-demand system 140 provides capacity on demand by restoring an appshot 220 when needed to provide one or more instances of an application. The system monitors the resources utilized to provide processing for the active application instances. The application provider is then charged according to the amount of computational resources utilized in providing users with access to their distributed applications.

FIG. 11 depicts a simplified flow diagram of one implementation of a sequence of steps executed by the present invention to perform a snapshot of a process or application instance. In step 250, a snapshot of an active application is requested. The processes that are snapshotted together in the form of an application chain share the same application ID (AID). As such, the AID is looked up (decision step 252) in memory containing a list of the AID's present. If the AID is not found control returns at step 254. However, if the AID is found, control continues to decision step 256 where a search is performed for a process belonging to the application having the matched AID. If a process is found, control continues to step 258, where the process is suspended. If the state is consistent and the threads are quiesced (decision step 260), control loops to step 256 and the remaining processes belonging to the application are located and suspended. However, if a process is located that does not have a consistent state or a thread is not quiesced, suspended processes are resumed and the snapd module 262 returns a notice that the snapshot was not completed.

In one embodiment, a snapd module (snapshot daemon module) comprises a daemon listening on a port that does the snap-shotting of all processes that have the same snapshot id (snapid). The state of the applications after a snapshot is taken is stored in one or more files. The state that is typically saved includes process state information (for example, in a snapshot file per process), and memory information (for example, in a file per anonymous and shared memory segments). In one embodiment, the snapshot file stores all process state information as apeudo ELF file. A different ELF\_NOTE section is created for each process state record (such as file descriptor). Another file called snaplist.snapid identifies all the processes in that snapid along with any parent/child relationship. In one embodiment, the process state information is collected during execution in preload libraries or when the snapshotting is done from the kernel.

Once the related processes are suspended, the states of the suspended processes are checked to see if they are virtualized (step 268). A virtualized state is any process state that reflects a virtualized resource. If the state is virtualized, it is retrieved at step 270 otherwise the non-virtualized state is retrieved at step 272. If the state has changed since the last snapshot (step 274), the new state is recorded. Control then loops to step 266 and executes through the above sequence of steps until the states of the processes are checked. Once completed, control proceeds to step 282, registered global states, such as semaphores, are removed. A registered global state is a state that is not specifically associated with any one process (i.e., private state). A global state is usually exported (accessible) to all

processes and its state can be modified (shared) by all processes. Control proceeds to step 284, where the process is terminated. If there are remaining processes (step 286), these are also terminated. This sequence of steps is concluded to create a snapshot of an application instance which is stored, in one embodiment, as a file and made available for reactivation or transmission to another compute modules, and/or other edgepoints.

FIG. 12 illustrates a simplified flow diagram of one implementation of the sequence of steps executed to restore a snapshot application. The snapshot application is accessed via a shared storage mechanism through a restore call at step 300. The AID for the snapshot application is looked up and (decision step 302) if not found a notification is issued that the snapshot application has not been restored. However, if the AID is found, control continues to decision step 304 where, if the snapshot application matching the AID is located, the global/shared state for each process associated with the snapshot are found. Control then continues to step 308, where remaining global or shared state for the processes are recreated. Then a process is created that inherits the global/shared state restored in step 308, and the created processes are isolated to prevent inter-process state changes. At step 314, for each type of state within the processes, the process-private resources are recreated to their state at the time the application was snapshot. If the state is virtualized (decision step 316), the system state is bound to a virtual definition. In one embodiment, as part of the restore a step is performed to create a virtual mapping. This is done by taking the system resource that was created in step 314 and binding it to the virtual definition that was saved during the snapshot in step 266. This allows the application to see a consistent view of resources, since it may not be guaranteed that at restore time the same system resource will be available. If the state is shared with another process, such as via a pipe (decision state 320), the shared state is reconnected with the other process at step 322. If there are more states (decision step 324) steps 314 through 322 are repeated. Once steps 314 through 322 have been executed for all states, control continues to step 326, where the process is placed in synchronized wait. If there are remaining images in the snapshot application (decision step 328), steps 310 through 326 are repeated. Once all images have been processed, control continues to step 330, where traces and states induced during restore of the process are removed, and a synchronized operation of the processes occurs at step 332. Once steps 300 through 332 have executed without error, the restored application can continue to run without interruption. Thus, the present invention avoids the overhead and delay of shutting down an application, storing data to a separate file, moving both the application and data file elsewhere, and restarting the program or application.

FIGS. 13A-C depict one implementation of one embodiment of the system and method or process for providing on-demand compute resources provided by the present invention. FIG. 13A shows a simplified block diagram of one embodiment of an edgepoint 350 of the present invention. The edgepoint 350 includes a memory 352, which is any type or combination of memory including fast semiconductor memory (e.g., RAM or ROM), slower magnetic memory (e.g., hard disk storage), optical memory substantially and any conventional memory known in the art. Within memory 352 is stored appshots 220a-f. Edgepoint 350 further includes compute resources 354. Compute resources include but are limited to at least one of a microprocessor, memory, control logic, or combination thereof. In operation, the edgepoint 350 is accessed by at least one entity, such as users 124a-b, over a network, such as the internet 126. When a user 124a is routed

to the edgepoint 350 and requests one or more applications, the edgepoint 350 determines which appshot 220a-f provides the desired application. The edgepoint pulls or duplicates the appshot 220b from a halted or snapshot state, and unencapsulates or restores the appshot 220b onto a first set of compute resources, such as a first server 354a. The server 354a activates an instance of the application 356a to allow the user 124a to utilize the application 356a. In one embodiment, the edgepoint 350 restores an application from an appshot 220 immediately upon request.

Referring to FIG. 13B, once a server 354a is running the application instance 356a, the application 356a is fully active and operating, so that additional users 124b can be routed to and gain access to the active application 356a. Because the application 356a is already active, additional users 124b get an immediate response with substantially no delay. However, as more users request access to the application 356a, the response time begins to suffer, and the effectiveness of this application begins to deteriorate because the application 356a becomes overloaded. As additional users 124c attempt to access the instance of the application 356a response time degrades. In one embodiment, a predetermined response time threshold or limit is set, or a predefined number of users is set which limits the number of users allowed to access one instance of an application. Thus, when a new user 124c attempts to access the application 356a, and this new user 124c exceeds the predefined threshold, the edgepoint 350 activates the appshot 220b to initiate a second instance of the application 356b. Thus, this demonstrates the ability of the present invention to provide capacity on the run or on-demand, and provide an optimal response time for the applications 356a-f.

Referring to FIG. 13C, as the numbers of users accessing the second instance of the application 356b continues to increase, the threshold will once again be reached. Once additional users 124e attempting to access the second instance of the application 356b exceeds the limit, the edgepoint 350 will again activate the appshot 220b to activate a third instance of the application 356c. This will continue until the servers 354a-f are occupied to capacity running at least one of the applications 356a-f. At which point, the edgepoint 350 will signal the system 140 and the on-demand application system 140 will then direct or route additional users to other edgepoints 350 throughout the distributed on-demand system 140. Thus, the system 140 ensures an effective response time and reliability, and thus improves user satisfaction.

The present invention also provides for the freeing up of system resources to be utilized by alternative applications. As the number of users 124 decrease below a threshold, one of the application instances, such as the third instance 356c, can be terminated or snapshot to free up a set of resources. The freed resources allows the edgepoint 350 to activate and run an alternative appshot 220a-f. Thus, the on-demand system 140 not only provides resources but reduces resources when not needed, resulting in a reduced cost to the application provider. Further, the present inventive on-demand method and system 140 provides for the ability to share resources among application providers because applications can be initiated as well as removed from compute resources allowing a substantially unlimited number of applications to utilize the same resources.

In one embodiment the edgepoint 350 is not limited to activating a single application 356 from a single appshot 220. A single edgepoint 350 can activate a plurality of different applications 356 on a variety of different sets of compute resources, such as servers 354, based on the applications requested by the users 124.