

(72) LO, Ron, US

(72) MAHAJAN, Om Prakash, US

(72) VAIOS, Christos I., US

(72) VISSER, Eldred James, US

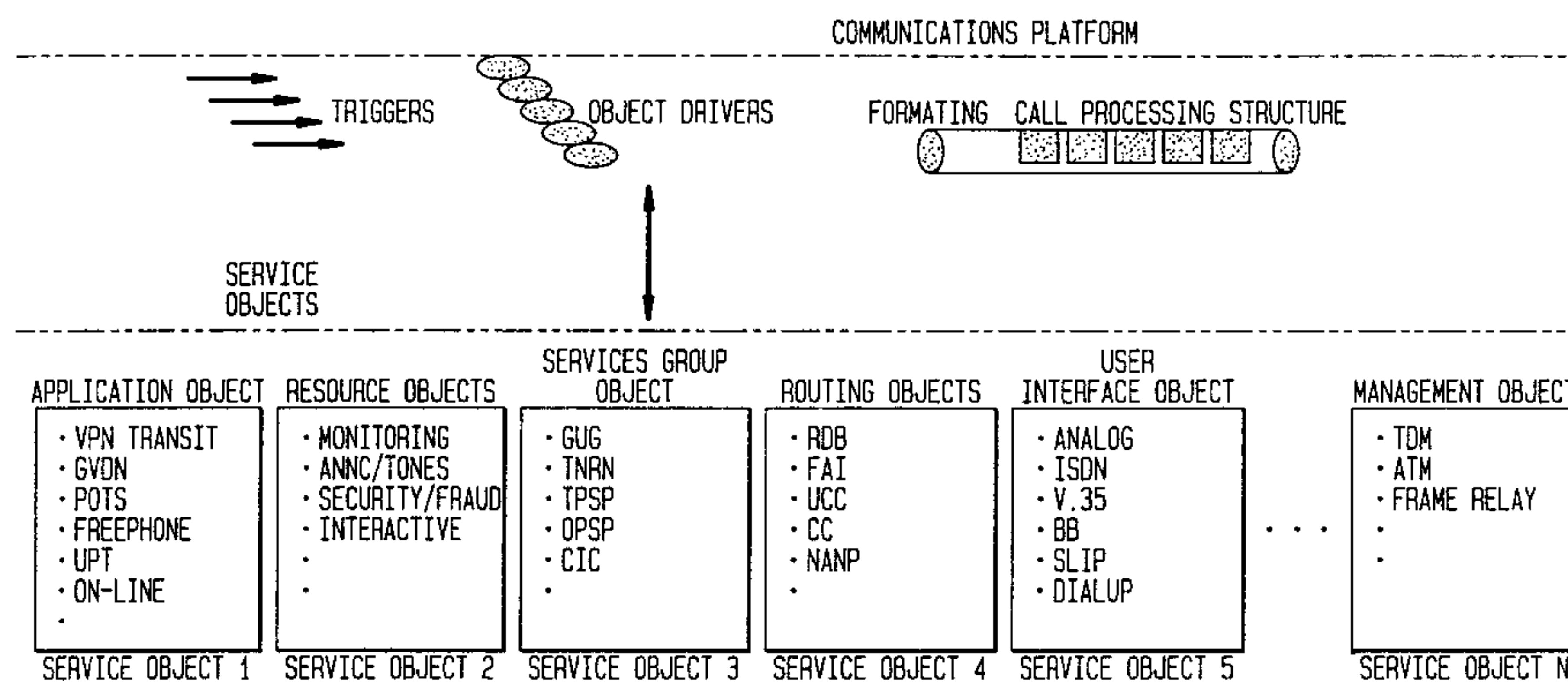
(73) AT&T CORP., US

(51) Int.Cl.⁶ H04Q 3/00

(30) 1997/04/18 (08/844,424) US

(54) **METHODE ET SYSTEME DE MISE EN OEUVRE DE SERVICES
DE TELECOMMUNICATIONS INTELLIGENTS UTILISANT
UNE ARCHITECTURE « ORIENTEE OBJETS » AUTONOME
ET TOLERANTE AUX FAUTES**

(54) **METHOD AND SYSTEM FOR IMPLEMENTING INTELLIGENT
TELECOMMUNICATION SERVICES UTILIZING
SELF-SUSTAINING, FAULT-TOLERANT OBJECT
ORIENTED ARCHITECTURE**



(57) Ensemble de capacités multidimensionnelles permettant l'échange et l'exploitation de données de service par différents programmes logiciels et plates-formes matérielles. Un déclencheur basé sur la demande d'appel est reçu à une plate-forme de commutation. Le déclencheur active un programme de gestion d'objet propre à un service. Le programme de gestion est construit par interaction avec des objets de service. Lorsqu'il est construit, les objets de service utilisent une information de traitement d'appels basée sur le programme de gestion d'objet nouvellement construit. Tout au long du processus, les fonctions de gestion de réseau sont exécutées de manière tolérante aux fautes pour assurer la fiabilité.

(57) A set of multidimensional capabilities allows service-related data to be exchanged among and operate with different hardware platforms and software programs. A trigger based on the call request is received at a switching platform. The trigger activates a service-specific object driver. The object driver is built by interacting with service objects. Once the object driver is built, the service objects are executed with call processing information based on the newly built object driver. During the entire process, network management functions are executed in a fault-tolerant fashion to ensure reliability.

**Method and System for Implementing Intelligent
Telecommunication Services Utilizing
Self-sustaining, Fault-tolerant
Object Oriented Architecture**

ABSTRACT

A set of multidimensional capabilities allows service-related data to be exchanged among and operate with different hardware platforms and software programs.

A trigger based on the call request is received at a switching platform. The trigger activates a service-specific object driver. The object driver is built by interacting with service objects. Once the object driver is built, the service objects are executed with call processing information based on the newly built object driver. During the entire process, network management functions are executed in a fault-tolerant fashion to ensure reliability.

**Method and System for Implementing Intelligent
Telecommunication Services Utilizing
Self-sustaining, Fault-tolerant
Object Oriented Architecture**

FIELD OF THE INVENTION

The present invention relates to implementing intelligent telecommunication services. More specifically, the present invention relates to a method for implementing self-sustaining, fault-tolerant object driver software within a telecommunications network.

BACKGROUND

Current advanced telecommunication services are composed of huge program instructions that are disjointedly distributed across various telecommunication platforms. The ever increasing end-user demand for intelligent services has forced the telecommunication providers to rapidly offer intelligent services that have resulted in complex, costly, and inflexible software architectures. These current software architectures are driven by individual service needs and quite often a number of capabilities are duplicated across multiple services. Furthermore, these current complex software architectures can introduce a subtle, unwarranted interaction between different services. Elaborate testing must be undertaken to ensure that a new service does not adversely affect existing services.

Consequently, these service program instructions suffer several shortcomings. They are difficult to maintain while a call is active, difficult to port onto a new platform and costly to develop. Offering customers bundled services is very difficult due to the lack of correlation of capabilities across various platforms. Furthermore, also due to the lack of correlation, as several services are to be activated for a given call, the service program instruction sets can only be performed consecutively, rather than simultaneously. Thus, they consume an enormous amount of the real-time and memory resources of the actual network elements, i.e., switches, databases, etc. Consequently, these telecommunication services are very complex and inflexible, and customers are forced to buy stand alone service offerings.

This patent application is related to commonly assigned U.S. Patent No. 5,999,603 entitled METHOD AND APPARATUS FOR PROVIDING A MULTI-NETWORK GLOBAL VIRTUAL SERVICES, Serial No. 08/844,425, filed on April 18, 1997 and issued on December 7, 1999.

SUMMARY OF THE INVENTION

The present invention relates to a method for implementing intelligent telecommunication services by utilizing modular, self-sustaining, fault-tolerate object driver software.

The present invention allows software for telecommunication services to be modified easier without the risk of adversely affecting existing services, reused among different physical locations, and developed faster to reduce the time of bringing new products to

market. Thus, the present invention provides improved reliability and flexibility.

The present invention implements intelligent telecommunication services between a calling party making a call request and a called party, within a telecommunication network. The present invention receives a trigger based on the call request from a source. The trigger activates a service-specific object driver. The object driver is built by interacting with service objects. Once the object driver is built, the call is formatted with call processing information based on the newly built object driver. During the entire process, network management functions are executed in a fault-tolerant fashion to ensure reliability. Once the call is formatted, the call can be forwarded to a next switching platform based on the call processing information. The present invention can be performed by a database or by at least one network element.

The network management functions can identify and recover an object driver not received. The network management functions can monitor several service objects simultaneously and more than once. The network management functions can also include generic network management of call recovery.

The trigger actuating a service-specific object driver can be initiated by the source or by the network.

The object driver can be built from many service objects simultaneously.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates the configuration and operation of a contemporary telecommunications network.

Fig. 2 illustrates a services functional model according to an embodiment of the present invention.

Fig. 3 shows a flowchart of a process according to an embodiment of the present invention.

Fig. 4 shows a network configuration for a VPN transit service according to an embodiment of the present invention.

Fig. 5 illustrates the functional components of the present invention for the VPN transit service of Fig. 4.

DETAILED DESCRIPTION

The following terms, commonly known to those of ordinary skill in the art, are discussed throughout the detailed description and are provided here in glossary form for convenience.

BB:	Broadband
CdPN:	Called Party Number
CgPN:	Calling Party Number
CXR ID:	Carrier ID
FAI:	Foreign Administration ID
GUG:	GVNS User Group
GVNS:	Global Virtual Network Service
ISDN:	Integrated Services Digital Network
OPSP:	Originating Participating Service Provider
POTS:	Plain Old Telephone Service
RDB:	Routing Data Block
SI:	Service ID
TNRN:	Terminating Network Routing Number
TPSP:	Terminating Participating Service Provider
UCC:	Unassigned Country Code
VPN:	Virtual Private Network
WZ1:	World Zone 1

Fig. 1 illustrates the configuration and operation of a contemporary telecommunication network. Network 10 comprises a plurality of switching platforms 20 and links 30. Each of the switching platforms 20 also is connected to at least one adjunct platform 50. An adjunct platform can be connected to a single switching platform, or can be shared by two or more switching platforms or databases. Note that the depiction of a network comprising only five switching platforms is for convenience of illustration; an operating network can have a much larger number of switching platforms and associated connecting links. Similarly, although Fig. 1 shows a specific arrangement of adjunct platforms and switching platforms, any number of arrangements are possible.

Various switching platforms are shown illustratively connected to communications devices 40. It should be understood that the individual communications devices shown connected to the switching platforms in the figure are used for simplicity of illustration, and that an actual implementation of such a network would ordinarily have a number of communications devices connected at such switching platforms. Note, as well, that the communications devices shown in Fig. 1 can also represent another network or network devices, such as a LEC or PBX, which is connected to network 10.

Each communications device 40 can generate information for use by, or receive information from, other communications devices in the network. The term "information" as used herein is intended to include text, graphics, voice, and/or video. Communications device 40 also transmit call request data that can identify the source, the destination and other data related to services available to manage the call.

The present invention illustrated in Fig. 2, introduces a set of multidimensional capabilities which

allow service-related data to be exchanged among and operate with different hardware platforms and software programs. Fig. 3 shows a flowchart of a process according to an embodiment of the present invention; Table 1 summarizes the functional components of the present invention and non-exhaustively lists the elements associated with each functional component.

Triggers	Object Drivers	Service Objects	Call Processing Structures
• CdPN • CgPN • CXR ID • *288	• Service ID (SI) • Service Group ID • Data Rates	• Application • Resources • Services Group • Routing • User Interface • Network Transport • Management	• Application-specific protocols • Messages • Parameters/IEs • Service-specific protocols • Messages • Parameters/IEs

Table 1

The process begins at step 100 when a call request is received by a switching platform 20. Several elements of the call request can be used by a switching platform 20 as triggers to prompt the switching platform to take specific actions. Triggers are instructions or application service elements that direct a switching platform to suspend basic call processing control and invoke the service logic control. For example, the source's number or Calling Party Number (CgPN) can be used as a trigger. The first column of Table 1 lists other possible parameters that can be used as triggers.

Triggers can prompt specific actions by the switching platform 20 during call setup or while a call is already established and the source or the network wants to perform a new action (e.g., a sequence call).

A combination of one or more triggers, either network specific and/or user specific, can be used to prompt a specific action by the switching platform 20.

For example, the Called Party Number (CdPN) and the Calling Party Number (CgPN) have the unique characteristics of a public or customized dialing plan

sequence and can meet the criteria for a trigger. The CdPN may be different from the user dialed number (DN), therefore accommodating instructions from database dips.

Similarly, the CgPN can be different than the Automatic Number Identification (ANI) but the same as the billing number (BN), therefore the call can be billed properly.

Similarly, when a call is already active, a trigger like *288 (*ATT) could be perceived as a network trigger, which means the user wants to go back to the network functions for additional capabilities or communication, e.g., third party calling.

Upon recognition of the type of action needed based upon the trigger, the process continues to step 110. At step 110 the switching platform 20 assigns a specific object driver for the call. Object drivers are service-specific application elements that can dynamically invoke specific service objects as will be discussed below. All object drivers have the unique capability to control all call functions in a multitasking fashion when an application is executed. For example, an object driver can communicate with the user or network at any point during the processing of a call without necessarily interrupting the service logic, e.g., call transfer, call sequence or third party call control. The second column of Table 1 lists some possible object drivers.

If the trigger is initiated by the user, the assigned object driver will look for specific capabilities to which the user has subscribed and try to correlate the necessary information before continuing to step 120, the next step in the process. If the switching platform 20 cannot recognize the user or the call, it can assign a "pseudo object driver" and all the information is correlated and the user's request is

categorized. Then, for network purposes, the "pseudo object driver" can be overwritten for easier processing of the call (e.g., bill the call to the right system).

Once the appropriate object driver has been assigned in step 110, the process proceeds to step 120.

At step 120, service objects are activated based on the contents of the object driver assigned in step 110. Service objects are sets of service-specific instructions that define the call processing segments of a call. A service object can exist for each type of network function; see, for example, the third column of Table 1. Table 2 non-exhaustively illustrates several service objects organized by function: Application, Resource, Service Group, Routing, User Interface, Network Transport Type, and Management. Table 2 also illustrates several members of each illustrated service object.

Appl.	Resource	Servi ce Group	Routi ng	User Interfa ce	Network Transpo rt	Mngt .
VPN Transit	Monitori ng	GUG	RDB	Analog/ POTS	TDM	
GVNS	Annc/Ton es	TNRN	FAI	ISDN	ATM	
POTS	Security / Fraud	TPSP	UCC	V.35	Frame Relay	
Freeph one	Interact ive	OPSP	CC	BB		
UPT		CIC	NANP	SLIP		
On-line				Dialup		

Table 2

Service objects can interact/exchange information with each other to create the correct call processing segments, consistent with user-subscribed capabilities and the available network resources. Some of these instructions could be used as unique "pointers" to drive a chain of call processing segments. These pointers (e.g., a country code) could derive both user and network instructions for a particular call. The collection of these service objects can then assist in forming the appropriate communication agents or call processing structures, as will be described below in connection with step 130.

The Management service object ensures proper handling of the other service object drivers; in other

words, the Management service object provides the needed reports, responds to failures by communicating with the user and with the network call maintenance systems (OA&M). The Management service object detects the occurrence of any possible errors when building each object driver. The management service object can check multiple times during the processing of the other service objects, even though all of the service objects are being processed simultaneously. If an error occurs during processing of one of the other service objects, the Management service object can recover from the error by reinitiating the processing, rather than allowing the call to fail. This recovery can occur at all call levels including when the switching platform is prompted by a trigger, when the object driver is being built, when the service object drivers are being executed or even when the call processing information is being formatted.

Once all service objects are determined in step 120, the process proceeds to step 130. At step 130 switching platform 20 must identify and execute call processing structures. Call processing structures are stored program instructions responsible for initiating a "call session" that manages the association of interconnections in various media, based on specific service object instructions. Once the call processing structures have been identified, a communication software module resident at switching platform 20 provides the necessary network-specific instructions (internal and external) on how to execute the call processing structures. In other words, the communication module builds and then processes the necessary protocol structure that incorporates the instructions from the service object's call processing segments. This includes

instructions to add or delete information based on the required interworking functions between networks/elements (e.g., switch to database, switch to adjunct/peripheral or switch to switch) and to form the correct signaling messages and information elements (e.g., Call Setup Message). The fourth column of Table 2 shows possible types of protocol structures for the call processing structures.

Once the call processing structures are identified and executed, the switching platform 20 has to format all the data according to the required protocol. The switching platform 20 builds the appropriate primitives for the external communications protocol and forwards the information to the next element or network.

The software associated with the functional components of the present invention can physically reside across various elements in a telecommunications network, i.e., switches, databases, and peripherals/adjuncts. Alternatively, the functional components of the present invention can be integrated into one network element, e.g., a database.

For illustrative purposes, the following example briefly describes how the present invention can implement a specific telecommunication service: a VPN transit service call from U.K. to Japan, via AT&T's domestic network. Fig. 4 shows a network configuration through which a call utilizing the VPN transit service can flow according to an embodiment of the present invention. Switching platform 200, a gateway switch located in the U.K., is connected to switching platform 210, a AT&T gateway switch. Switching platform 210 is connected to switching platform 220, a gateway switch located in Japan, and to an adjunct platform 230. The connections between the switching platforms can be

through public facilities or dedicated facilities. Although Fig. 4 shows only a few network elements for convenience of illustration, any number of additional network elements can be present.

In this example, the AT&T gateway switching platform 210 receives a transit call via a CCITT #7 ISUP signaling message. Because this call is intended to be a VPN transit call, the incoming signaling message includes the network destination code (CXR ID) which identifies the terminating service provider. In this example, the CXR ID would refer to a terminating service provider in Japan.

Fig. 5 illustrates the functional components of the present invention as the VPN transit service call of Fig. 4 is implemented. The switching platform 210 recognizes the CXR ID as a trigger 300 and then assigns an object driver 310; in this example the service ID object driver 310 is assigned the specific value of 22.

Switching platform 210 accesses adjunct platform 230 to build the service ID object driver 310 from the appropriate service objects 320. In this example, a service ID object driver 310 having a value of 22 is associated with a specific application service object 320, in this case VPN transit. The service ID object driver 310 is also associated with other service objects 320. For example, the service ID object driver 310 can be associated with user interface service objects, such as POTS; resources service objects, such as call monitoring and customized announcements; and service group service objects, such as GUG. Other service objects can also be used to process the call from a network perspective including routing, network transport, and perhaps billing recording among others.

Switching platform 210 executes the service objects 320. It can use specific pointers, such as "routing codes" within a routing table to speed the execution process, as illustrated in Table 3. A routing code be used to obtain the complete information needed to construct the call processing structure 330.

As illustrated in Table 3, the switching platform 210 can use a routing code to generate the call processing structure 330 (e.g., outgoing trunk information, network routing information, overflow indicator.)

Routing Code	Network Routing Info	Outgoing Trunk Info	Overflow Indicator	OPSP
XXX	CdPN		XXX- XXX*	1- 456-28X

Table 3

In the case where an entry is found in the routing table, the switching platform 210 populates the CdPN parameter and constructs the Forward GVNS parameter in the call processing structure 330. The switching platform 210 then routes the call to switching platform 220 via public facilities or dedicated facilities. If no match is found for the routing code in the routing table, the call is terminated.

It should, of course, be understood that while the present invention has been described in reference to a particular network configuration, different network configurations are possible. For example, the functional components of the present invention can physically reside across various elements in a communication network. Similarly, the functional components of the present invention can be integrated into a single network element.

CLAIMS

1. A method for implementing intelligent telecommunication services between a source making a call request and a destination, within a telecommunication network, comprising the steps of:

(a) receiving, from the call request, a trigger parameter for suspending basic call processing and invoking service logic control;

(b) assigning a service-specific object driver based on the trigger parameter, the object driver dynamically invoking a plurality of predefined service objects each having service-specific instructions;

(c) building the object driver by invoking the plurality of service objects;

(d) formatting call processing information based on the object driver; and

(e) executing network management functions during said steps (a), (b), (c) and (d) to enhance reliability.

2. The method of claim 1, further comprising:

(f) forwarding the call request to a next switching platform based on call processing information.

3. The method of claim 1, wherein said executing step (e) includes recovering the object driver identified as not received by a management service object of said plurality of service objects.

4. The method of claim 1, wherein said executing step (e) monitors the plurality of service objects simultaneously.
5. The method of claim 3, wherein said executing step (e) monitors at least one of the plurality of service objects more than once.
6. The method of claim 3, wherein said executing step (e) includes generic network management of call recovery.
7. The method of claim 1, wherein the trigger parameter received in said step (a) is initiated by the source.
8. The method of claim 1, wherein the trigger parameter received in said step (a) is initiated by the network.
9. A system for implementing intelligent telecommunication services between a source making a call request and a destination, within a telecommunication network, comprising:
 - a switching platform connected within the network between the source and the destination, said switching platform receives, from the call request, a trigger parameter for suspending basic call processing and invoking service logic control, said switching platform assigns a service-specific object driver based on the trigger parameter, the object driver dynamically invoking a plurality of predefined service objects each having service-specific instructions, said switching platform formats call processing information based on

- 18 -

the object driver, said switching platform executes network management functions to enhance reliability; and

an adjunct platform connected to said switching platform, said adjunct platform builds the object driver by invoking the plurality of predefined service objects.

10. The system of claim 9, wherein said switching platform forwards the call request to a next switching platform based on call processing information.

11. The system of claim 9, wherein the trigger parameter received by said switching platform is initiated by the source.

12. The system of claim 9, wherein the trigger parameter received by said switching platform is initiated by the network.

FIG. 1

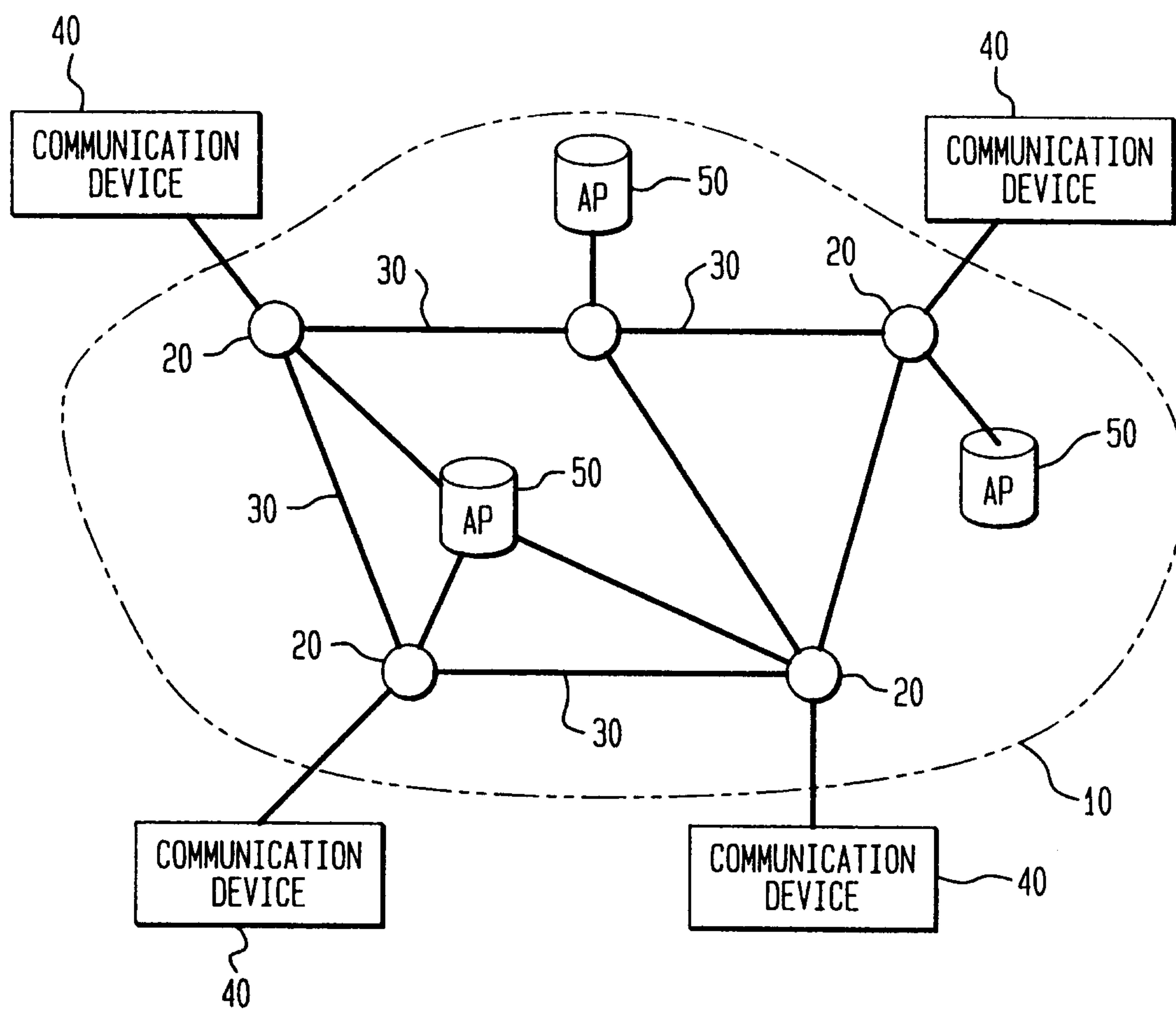


FIG. 2

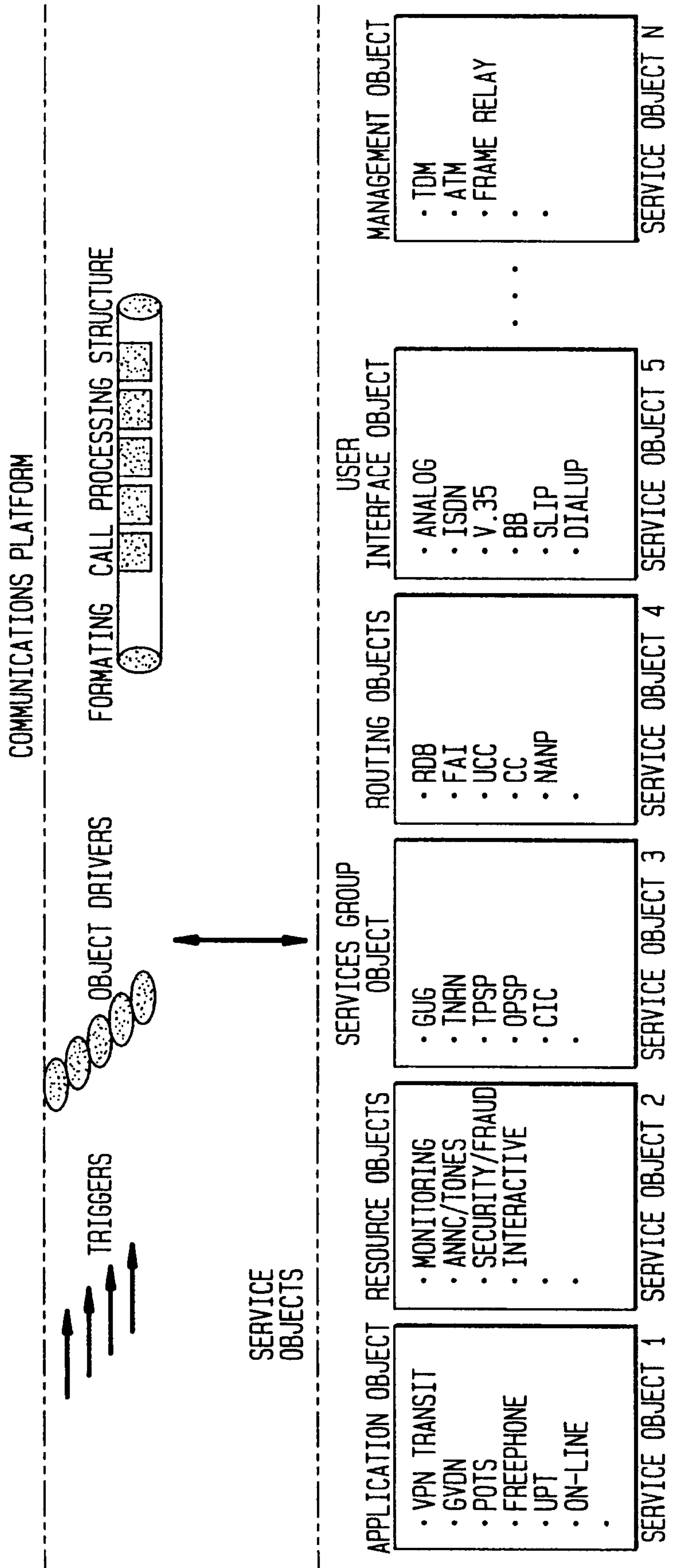


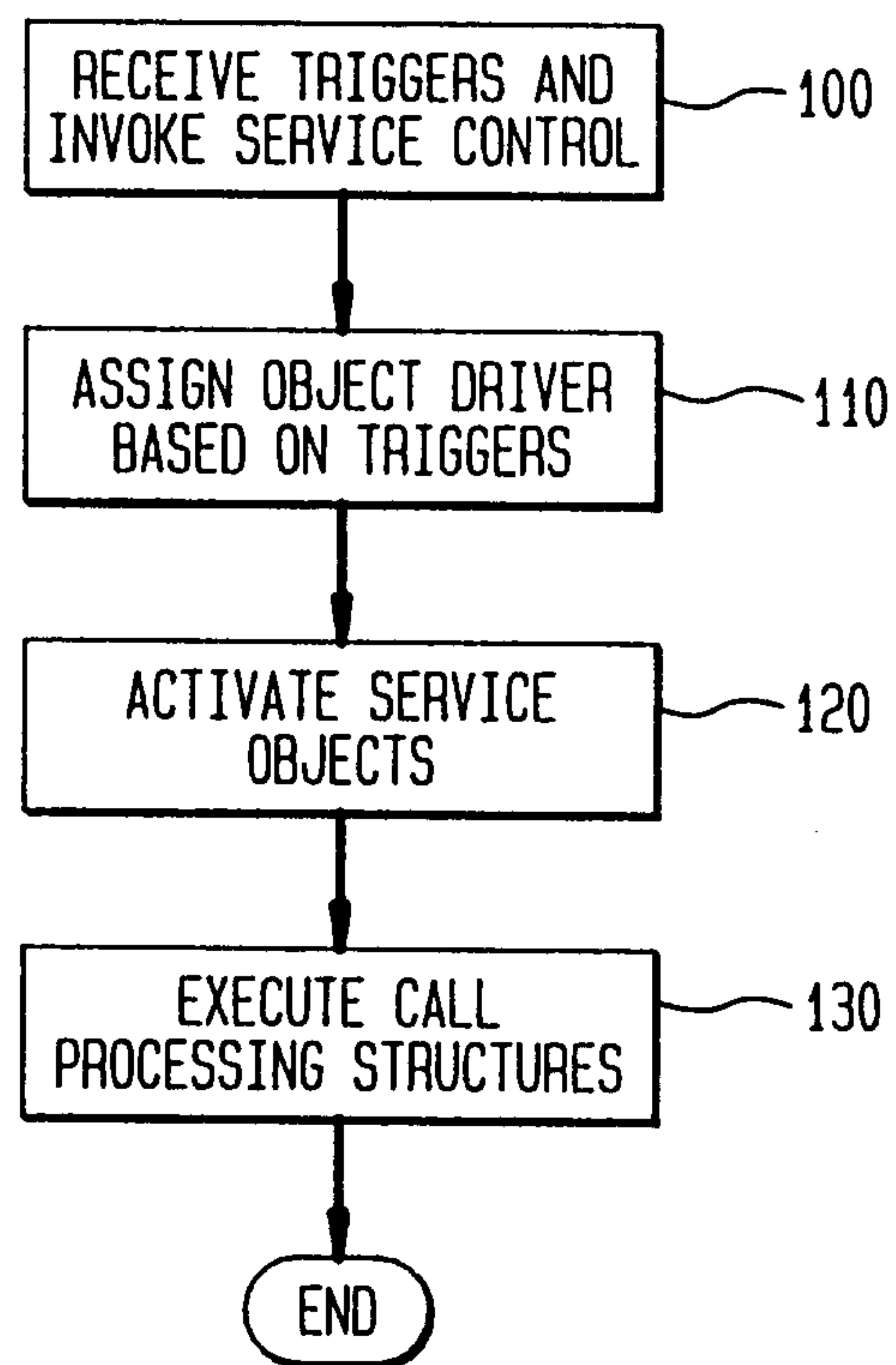
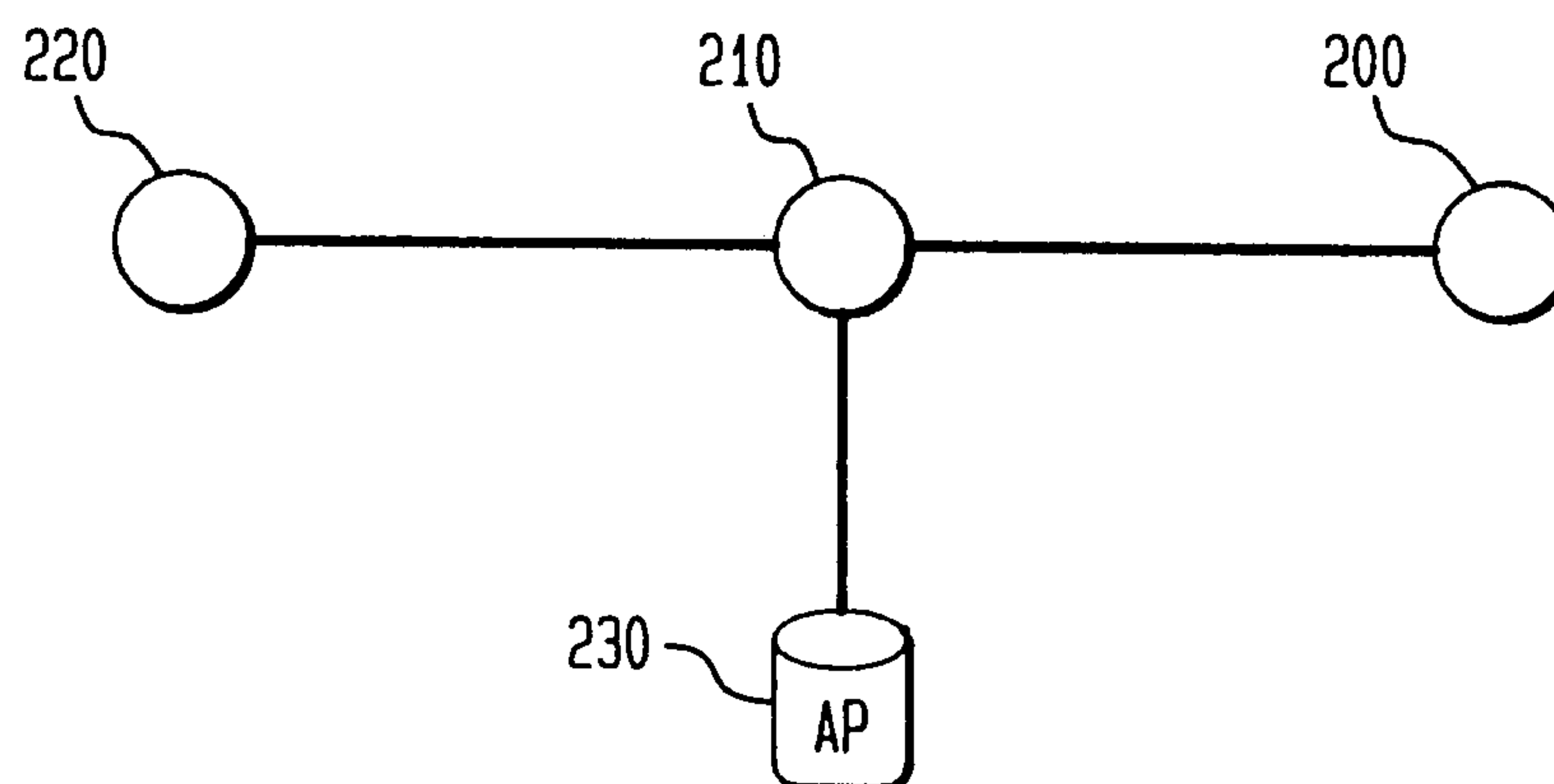
FIG. 3**FIG. 4**

FIG. 5

