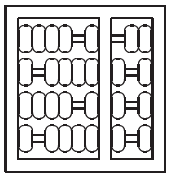

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

A Framework for Service Provisioning and Management in Virtual Active Telecom Networks

Fábio L. Verdi *Edmundo R. M. Madeira*

Technical Report - IC-02-13 - Relatório Técnico

October - 2002 - Outubro

A Framework for Service Provisioning and Management in Virtual Active Telecom Networks

Fábio L. Verdi* Edmundo R. M. Madeira

Abstract

The advent of Telecom over the last years brought up many challenges for service providers. The difficulty to offer services and, in the same time, to perform their management requires an integrated solution for both, service provisioning and management. In this paper, we outline an infrastructure for performing such tasks. This infrastructure allows to create Virtual Active Networks (VANs) and install services in Telecom environments based on the Active Networking technology. Besides service provisioning, the framework performs some management functions taking into account mobile services. The management system is based on a mobile agent platform and considers the management of VANs, where services can migrate from a VAN to another. We focus on three specific management areas: accounting, performance monitoring and configuration. We tested our system in the management of two Internet-based Telecom services, Call Forwarding and Virtual Private Network.

keywords: Active Networks, Service Provisioning, Service Management, Mobile Agent, Telecom.

1 Introduction

Service management in Active Networks (ANs) is currently a potential topic of research. Mobile Agent Technology (MAT) is a general name for facilities that support the transmission of code, as well as data, over a computer network. MAT-based services can be used for offering customized applications to customers [1]. In this context, the AN technology is particularly attractive to be used in Telecom environments, due to its facility for sending data and code to specific locations in the network. In the AN technology, the packets, also called *active packets* or *capsules*, can carry programs to be executed on routers and possibly change their state. These networks are active in the sense that intermediate nodes can perform computations on, and modify, the packet contents, in contrast to traditional networks in which the routers can modify a packet's header but the user data is not examined or modified [8].

In a Telecom environment, providers generally offer a wide range of telecommunication services that can be purchased according to the customer's requirements [9]. With AN

*Supported by CAPES

technology, the customization of services can be achieved by partitioning the network into *Virtual Active Networks* (VANs), which customers can lease from service providers. A possible solution for supporting this interaction between providers and customers is to develop a framework, which allows a customer to install, configure and run active services on his own VAN. These services can be static, or they can be moved from node to node within a VAN or between VANs. Information about accounting, performance and configuration enables the manager (human) to know the behavior of the managed environment to detect problems and take actions on them. In this article, we describe a framework for services provisioning and facilities for management of these services. The services offered by the infrastructure are VAN creation and service installation, both implemented using the Active Node Transfer System (ANTS) [16].

Our management system deals with mobile services by requiring that they use a common protocol for data reporting. Furthermore, we are considering the policy-based management due to the variety of services developed by different providers and customers. The policies we have developed are representative and other policies can be inserted further as necessary. A particular feature of our model is that a service can be only moved by a capsule, while in mobile agent environment, an agent has autonomy for moving itself without external action.

The infrastructure mainly consists of a Service Provider (SP) and a mobile agent based-system for service management in virtual active Telecom networks. The management system is used for managing the services installed over the VANs. In this work, the management system is tested on two telecommunication services, Call Forwarding and Virtual Private Network (VPN) [2].

Many approaches taken today have focused on AN technology. Some works argue on the capability AN technology has to manage traditional networks [14, 11]. Others claim at architectures or software architectures to provide and manage telecommunication services [13]. In this context, an architecture for network management that offers active services is presented in [12]. This architecture uses a combination of policies and adaptive algorithms allowing multi-user management of network based service components. In [15], a model for management of mobile services in VANs is described, and how the management information is collected across the Management Information Bases (MIBs) distributed in three hierarchy levels of management is shown. Considering that active networking has demonstrated its potential for Telecom environments, an approach for service management in this kind of environment is presented in [3]. The PANTS architecture, which is based on ANTS, is presented in [10]. Although this architecture provides the facility to install new services into the node dynamically, it does not perform static and mobile service management.

Taking into account the features from these approaches, our framework has incorporated some important aspects for Telecom environments such as: policy-based management, VANs, mobile services and mobile agent-based management. In contrast to [2], in which services are based on mobile agents, in our model the services are moved from a VAN to another using the active networking infrastructure.

The paper is organized as follows. In the next section we briefly review the basic concepts about active networking. Section 3 outlines our infrastructure and its components, and it illustrates four possible management scenarios. Section 4 describes some aspects about

implementation. Section 5 shows the tests we have performed. Finally, Section 6 presents the conclusion and gives an outlook on further research.

2 Active Networks and VANs

Active Networking [8, 5, 7] breaks with the tradition where the nodes only route packets according to their routing tables. Users can program the network by injecting their programs into it. The AN nodes can receive, send and run these external programs using local computational resources. The flexibility for developing new protocols reduces the difficulty of integrating novel technologies and standards into the shared network infrastructure [8]. These new protocols can represent a customized processing on a per-user or per-application basis. Some architectures for active networks enable a capsule to download applications and install them in that node. On the other hand, the capsule itself can execute some specific tasks when it is evaluated.

An important service offered by active networking is the VAN provisioning. A VAN can be described as a graph of virtual active nodes connected by virtual links where active packets can travel within a VAN or between different VANs [4]. The VAN is a very useful solution for isolating groups of customers and offering specific services for each customer's domain. In the ANTS toolkit a single (physical) active node can run several virtual active nodes belonging to different VANs.

Currently, AN technology has been presented as an appropriate solution for supporting service creation and deployment in the network infrastructure. In active networks, the flexibility to send and receive code and data to specific nodes has stimulated its utility as a facility to provide services in Telecom environments.

3 The Framework for Service Installation and Management

The framework we developed includes components for supporting VAN creation, service installation and management in virtual active Telecom networks. In this section we present these components, the defined policies, the configuration actions and, additionally, we focus on four possible Telecom scenarios which our model is able to handle. The management system we describe in this section is for managing a single domain. In Section 5.2 we address the extensions needed to support multi-domain management.

The proposed model is hierarchical in order to minimize the number of management agents that would be necessary to manage a great number of services. Most of the current management systems have a management agent which follows each mobile service all the time. If the service migrates to another local, the management agent follows that service on behalf of the management application [1]. In contrast to those approaches, our model has its management agents distributed in a hierarchical way and they do not follow the service when it moves to another local. This is a solution for reducing the number of management agents that would be necessary in order to manage all the services which are rapidly deployed and installed in today's Telecom.

Our model has a static manager responsible for the global management getting a global view of all VANs. There are also one-hop mobile agents for collecting accounting and performance data in each node. Getting the responses about accounting and performance, the manager (human) can take actions towards the configuration functional area. According to the accounting area, we are interested in answering some questions related to the number of requests to a specific service, VAN or host. According to the performance functional area we are interested in some questions related to the CPU and memory use. These management questions are divided into four cases: per service, per host, per VAN, and per domain (the network as a whole) [15].

Next, we present the policies we have defined. Afterwards, we outline the configuration management, the components of the infrastructure and, finally, the scenarios.

3.1 Policies

Telecom environments have to face some challenges in order to offer and manage services to their clients. Typically in these scenarios, the number of services to be provided is relatively large and, therefore, it is important to define some policies in order to control and avoid undesirable situations. Taking into account this implication, two possible solutions can be realized when a client requires a service. The first solution considers that whenever a client needs a service, a new copy of this service will be created and sent to the client. The second alternative considers that whenever a client requires for a service, possibly a copy of this service from a node will migrate to attend the customer requirement. Both solutions are unfeasible whether their consequences are considered. To install a new service copy for each customer will flood the network with services. To migrate a service from a node to another to attend customer requirements will result in many migrations. Thus, we have tried an intermediate solution defining some policies to minimize both, the number of service copies and the number of migrations.

We created six representative policies:

1. There are two kinds of service: internal services which can only migrate in the same VAN, and external services which can migrate between VANs;
2. A service S has a limited number of copies in the network (**L1**): S_1,S_2,...,S_n, where n$\leq$**L1**;
3. A service S has a maximum bound of times for migrating in a period of time (**M1**). Each copy of S, S_i, has to follow this threshold;
4. The SP can install a new service copy requested by a user until a threshold of **L2** (L2$\leq$L1).
5. The Least Recently Used Service (LRUS) will be the candidate copy for migrating to the destination host when a customer requires a new service. This happens if the threshold L2 was already reached;
6. A service copy may be removed if it does not migrate a minimum bound of times in a period of time.

Next, we present the cases that can happen considering these defined policies.

3.1.1 Analysis of Cases

Policies 2, 3, 4 and 5 defined above were created to implement our intermediate solution. There are four possible cases to be considered when a client requests a service. Let $\mathbf{S}_i$ to be the ith copy of service S, $\mathbf{C(S)}$ to be the number of copies of S in the network and $\mathbf{M(S}_i\mathbf{)}$ the number of migrations performed by S_i:

1. $\mathbf{C(S) < L2}$: in this case the number of copies of S is less than L2 and, therefore, the SP can install a new copy of S. This minimizes the number of migrations;

2. $\mathbf{C(S) \geq L2}$ **and** $\mathbf{C(S) \leq L1}$ **and there is the LRUS,** $\mathbf{S}_{lrus}$**, with** $\mathbf{M(S}_{lrus}\mathbf{) < M1}$: in this case, the threshold L2 was reached but the maximum number of migrations of S_{lrus} not and, therefore, S_{lrus} will migrate to attend the customer requirement. This minimizes the number of copies;

3. $\mathbf{C(S) \geq L2}$ **and** $\mathbf{C(S) < L1}$ **and the LRUS,** $\mathbf{S}_{lrus}$**, with** $\mathbf{M(S}_{lrus}\mathbf{)=M1}$: in this case, the maximum number of migrations of S_{lrus} was reached, so this service will not migrate during a period of time, but it is possible to create a new service copy whereas C(S) < L1. During the interval from L2 to L1 there is a balance between creating new copies and migrating services;

4. $\mathbf{C(S) = L1}$ **and the LRUS,** $\mathbf{S}_{lrus}$**, with** $\mathbf{M(S}_{lrus}\mathbf{) = M1}$: in this case, the maximum number of migrations of S_{lrus} and the maximum number of copies of S were both reached. The service will not be installed.

In case 3, we are not considering the second LRUS, the third LRUS and so on. We assume that if the LRUS has migrated a lot, it is more suitable to create a new service copy to attend the customer requirement.

Policy 3 is responsible for avoiding a service to migrate many times. If a service is being required for many clients in the same time, policy 3 will not permit more migration than specified by the service developer. In this case, a new service copy can be created whether or not policy 2 permits to. Policy 4 lets the SP creates a new copy of a service until the threshold L2 allows it. After this, for each new client requirement, a new copy of a service will be created if the policy 3 does not allow the required service to migrate. For instance, suppose a service S with the following policies:

- Internal or External: external;
- Maximum bound for migrating (M1): 10;
- Maximum number of copies allowed (L1): 8;
- Number of copies allowed following policy 4 (L2): 4;

Following the above specifications for service S, the SP can install 4 required copies of this service without testing policy 3. When the limit (L2) is achieved, each new copy to be provided have to follow the policy 3. In this case, the SP only installs a new copy in the network whether a required service cannot migrate during the customer invocation. In our example, the maximum migration number for service S is 10. Therefore, after creating 4 copies of the service S in the network (S_1,S_2,S_3,S_4), next copies will be created only if there is no possibility to migrate the LRUS of service S (one among the copies of S) from a node to another in order to attend the customer requirement. In summary, policy 4 is useful to minimize migrations of a service just after its installation. As depicted above, the first four copies of S will be installed without migration.

Policy 1 gives to the developer of the service some flexibility to define a service to migrate only in the same VAN or among VANs. Policy 6 implies that a service can become a candidate to be removed taking into account its number of migrations. Although the few number of migrations of a service turns it a candidate to be removed, if a service copy is receiving a great number of requests in a host, it will not be removed in that time.

Besides the defined policies, other policies can be developed and inserted into the database in a flexible way to satisfy specific customer's domains.

3.2 Configuration Management

Based on accounting and performance information and based on the policy each service has, we defined three types of possible actions to apply. These actions are related to the configuration area, and as the defined policies, they are representative and other actions can be inserted as demanded. The following configuration actions has been defined:

1. Creating a new service copy;
2. Moving a service for load balancing;
3. Deleting a service from the environment.

The first configuration action is responsible for creating a new service copy when cases 1 and 3 (as explained in Section 3.1.1) are considered. The second configuration action allows the manager to move services from a node to another in order to balance the load. This is a typical action to be taken when the manager detects an overloaded node. The last configuration action we have defined is useful to remove services which were created in order to attend a period of demand and no longer are being used. Configuration action 3 is not applied on services which were installed together the VAN (see first scenario in Section 3.4). Thus, those services installed with the VAN are extinguished only when the VAN is released. Configuration action 3 implements the policy 6.

3.3 Components of the Infrastructure

In this section we give a brief description about each component of the infrastructure and their roles. This infrastructure is composed of a Management Remote Application (MRA),

a Management Center (MC), a Service Provider (SP), a Global Naming Service (GNS) and the Distributed Management Agents as described below (Fig. 1).

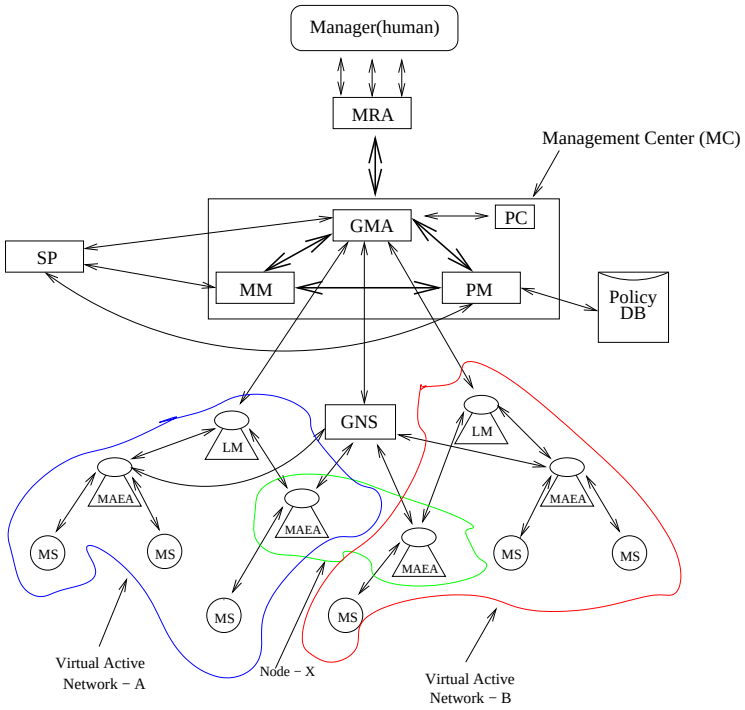

Figure 1: Components of the Infrastructure and their relationship

- **Management Remote Application - MRA**

  This application presents to the manager (human) the interface with the methods related to the management questions (accounting, performance and configuration). The MRA invokes remote methods on the *Global Manager Agent* (GMA, see below) for collecting management data. The MRA acts as the human-computer interface for the service management system. This application allows the manager to start service management actions and displays monitoring information.

- **Service Provider - SP**

  It is responsible for offering services to the customers. When a client sends a capsule to the SP to create a VAN and its services or to add a new service to his VAN, the SP notifies the GMA and the *Migration Manager* (MM, see below). Also, in a service migration or removal, the SP notifies the MM to controll its migration tables and notifies the GMA to update its local information about the VANs.

- **Management Center - MC**. It is composed of a Policy Manager, a Global Manager Agent and a Migration Manager.

- **Policy Manager - PM**: This component is responsible for controlling, insertion, updating and removal of a policy in the database;
- **Migration Manager - MM**: When a new VAN is created, the services in its hosts must be managed and, therefore, this component sends accounting management agents to the nodes. The MM controls each service migration over the network applying the policies on each one. Also, this component is responsible for sending performance agents to collect performance data.
- **Global Manager Agent - GMA**: It is the centralized static manager of the model. There is only a GMA in the network as a whole, i.e, in a domain. This component is responsible for analyzing accounting, performance and configuration data on all VANs and their services. Furthermore, it notifies the MM to send performance agents to specific hosts. The GMA has a view of the domain as a whole and together with MM and PM it can offer a large set of management information to the manager;

- **The Distributed Management Agents**. There are Local Managers and Managed Active Element Agents.

  - **Local Manager - LM**: There is an LM per VAN. It is responsible for management of its VAN and collects data of each *Managed Active Element Agent* (MAEA, see below) located in the hosts of the VAN. One of the aims here is to filter the information before sending it to the GMA;
  - **Managed Active Element Agent - MAEA**: It is responsible for management of one or more services in a host of a VAN, being the lower level of the management. There is an MAEA per host per VAN. This agent is an interface between the ***Managed Service*** (MS) and the management system and it has the following tasks:
    * - Activating and deactivating the management filter for obtaining account monitoring data in a period of time;
    * - Gathering information from services using the management interface which each service has;
    * - Sending information about services to other management agents when the services are migrating;
    * - Receiving information which was sent from other management agents about services are arriving.

- **Global Naming Service - GNS**

  This component receives the service location and the service identifier and creates a reference that indicates where the service is located. The management agents search in this component for getting service references.

All agents are distributed through the network. Fig. 1 shows two different VANs, A and B. We can see that node X belongs to both, but the model separates each one for service management.

The management system collects accounting and performance data. Accounting information is collected throughout the hierarchical model following the sequence: Human → GMA → LM → MAEA → MS. The metrics collected in accounting management operations are the *received requests* and *residence time* in order to answer some questions, such as:

1. What is the most used service in a host, VAN or domain?

2. What is the most used host?

3. What is the throughput in a period of time t per host and per service?

The MS (Managed Service) has a management interface enabling some filters to be activated or deactivated. The filters are related to specific attributes for management and they can count or not the metrics in different periods of time.

On the other hand, to collect performance data, the management system sends one-hop mobile agents to the hosts where the services are located. They stay on the nodes during a period of time and return back to MC with the gathered data. Managers are able to detect hosts overloaded considering that the metrics collected in performance management operations are memory and CPU use. Furthermore, when accounting and performance data are analyzed together, the manager has a great amount of useful information enabling him to take actions using configuration management operations [15].

The system needs to maintain management information at different levels: MAEA, LM and MC. In this work, the bases used to store this information are called MIBs. In this context, the MIB at each level of management is responsible for answering some questions related to the accounting and performance areas.

The interaction between agents (typically MAEAs) of different VANs is not shown, but a manager from a VAN can access another manager from another VAN for exchanging data, since the customer domain policies allow this communication. In our model, this interaction between MAEAs of different VANs only occurs in an inter-VAN service migration.

3.4 Service Provisioning and Management Scenarios

The next four scenarios are associated with the customer's *Active Application* (AA), that is a program for customers using the available environment. The first scenario describes in details the VAN creation and the mobile service installation. The second one presents a service installation following case 1 (see Section 3.1.1), i.e., without migration. The third scenario presents a mobile service migration between two VANs to satisfy a customer requirement. This scenario follows case 2 from Section 3.1.1 in which the threshold L2 was reached. The last scenario follows case 3 from Section 3.1.1 and it represents the situation in which the LRUS cannot migrate but a new service copy can be created. These four representative scenarios allow to demonstrate how our framework can be applied for offering VAN services and managing a typical Telecom environment. We show how is the behavior and the relationship of the components when a customer wishes to install a VAN and when a customer requires a new service.

1. **Creating a new VAN specifying what hosts and what services are required**. In this scenario the following sequence is needed, as shown in Fig. 2.

   (a) Customer sends a capsule to the SP informing what hosts and what services each host will have;

   (b) The SP sends the required services to the set of hosts;

   (c) The SP notifies the MM and GMA about the new VAN;

   (d) The MM sends accounting management agents to the hosts and creates the VAN manager (LM);

   (e) Services register themselves in the GNS.

Note that in this case the PM is not consulted. In fact, based on the policies we have defined for this work, when a new VAN is created the SP always provides new services. However, if other policies are included, it can be necessary to consult the PM.

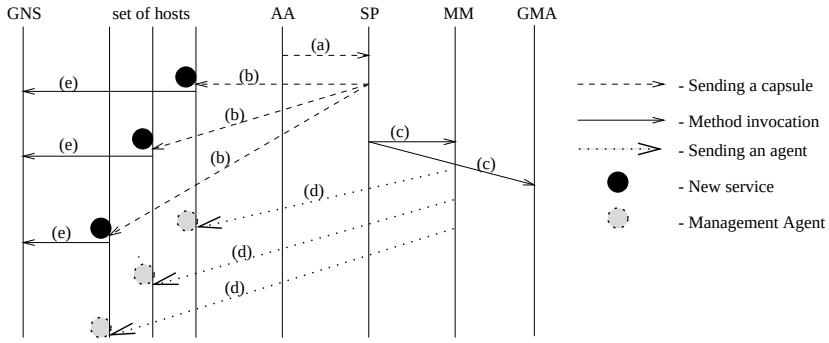

Figure 2: Customer creating a new VAN.

Next, we present the three possible scenarios to add a new service copy to the VAN. Fig. 3 shows the first three steps (a-c) which are necessary in all of the three scenarios, as described below.

(a) Customer sends a capsule to SP indicating what service is required and what destination host this service copy will be sent to;

(b) SP interacts with the MM in order to apply the policies on the required service;

(c) MM gets the service policies from the PM to analyze them;

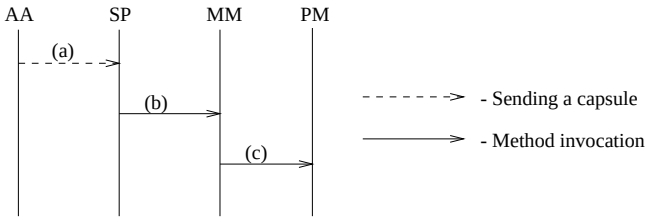

Figure 3: Customer requiring a new service.

After performing these three first steps, one of the following three scenarios can be executed based on the results obtained by the SP.

2. **Adding a service to the VAN following case 1**.
   In this case, the threshold L2 has not been reached, so the SP installs a new service copy in the VAN to attend the client request. In addition to the first three steps from Fig. 3, the following steps are necessary (Fig. 4):
   (d) SP sends a capsule to the destination host for creating a new copy of the required service;
   (e) SP notifies the GMA about the provisioning of a new service copy;
   (f) The capsule arrives in the destination node, instantiates the new service copy, and registers it in the GNS;
   (g) The service notifies its local MAEA that it arrived;
   (h) The capsule is forwarded to the new AA to notify that the required service has migrated. The AA can finally use the service.

3. **Adding a service to the VAN following case 2**.
   In this scenario it is considered that the limit L2 from policy 4 was achieved but the number of migrations not. If the required service is external, the LRUS can belong to another VAN and, then, the service must migrate between VANs. In addition to the three steps from Fig. 3, the following steps are necessary (Fig. 5):

   (d) SP gets the LRUS in the network. In this search, all the information about the LRUS is also obtained;
   (e) SP interacts with the MM in order to verify whether the migration is possible or not;
   (f) MM gets the service policies from the PM to analyze them;
   (g) MM returns back the results to the SP;
   (h) SP sends a capsule to AA of the VAN where the LRUS is located. This is necessary to notify the AA about the LRUS migration;
   (i) SP notifies the GMA about the migration;
   (j) The capsule is forwarded to the host where the service is located at this moment;

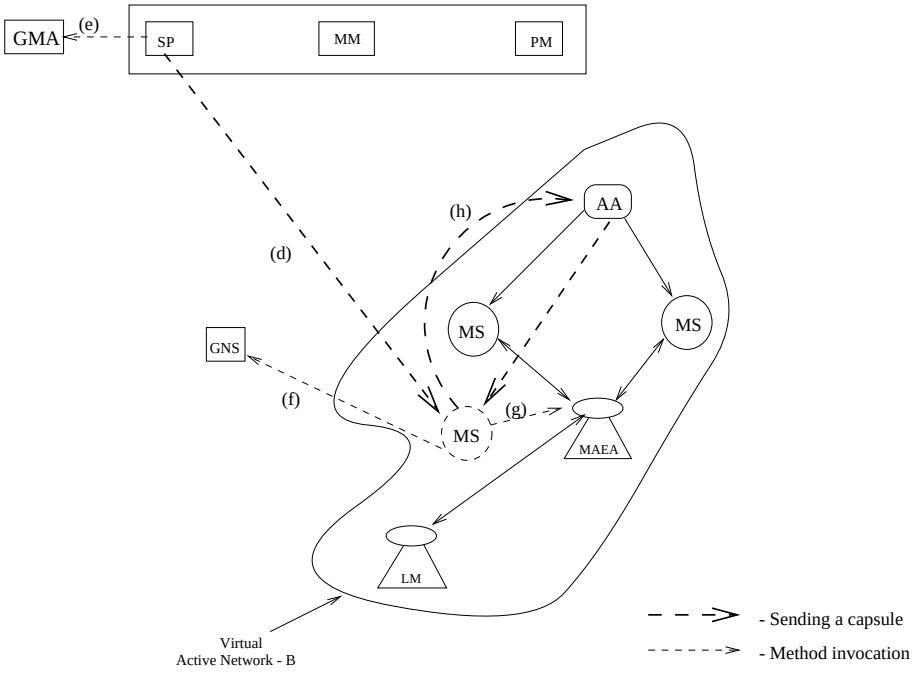

Figure 4: SP installing a new service (case 1).

(l) The service unregisters itself from the GNS and notifies its MAEA that it is migrating;
(m) The source MAEA notifies the destination MAEA that a new service is arriving;
(n) The capsule removes the service from local node and migrates it to the destination node.

To conclude the service installation, steps *f*, *g* and *h* from scenario 2 (Fig. 4) are executed.

4. **Adding a service to the VAN following case 3**.
In this case, the LRUS cannot migrate during a period of time, but a new service copy can be created and sent to the client. In addition to the steps from Fig. 3, steps *d*, *e*, *f* and *g* from scenario 3 (Fig. 5) are performed to find the LRUS and apply the policies on it. After the SP realizes that the migration is not possible but a new service copy can be created, steps *d*, *e*, *f*, *g* and *h* from scenario 2 (Fig. 4) will be performed to conclude the service installation.

In scenario 3, each LM is responsible for finding the LRUS in its VAN and it returns back the result to the SP. If the required service is internal (see policy 1), the same steps are performed, but in step *d*, SP obtains the LRUS only for the local VAN. For all other steps, the destination VAN is the local one.

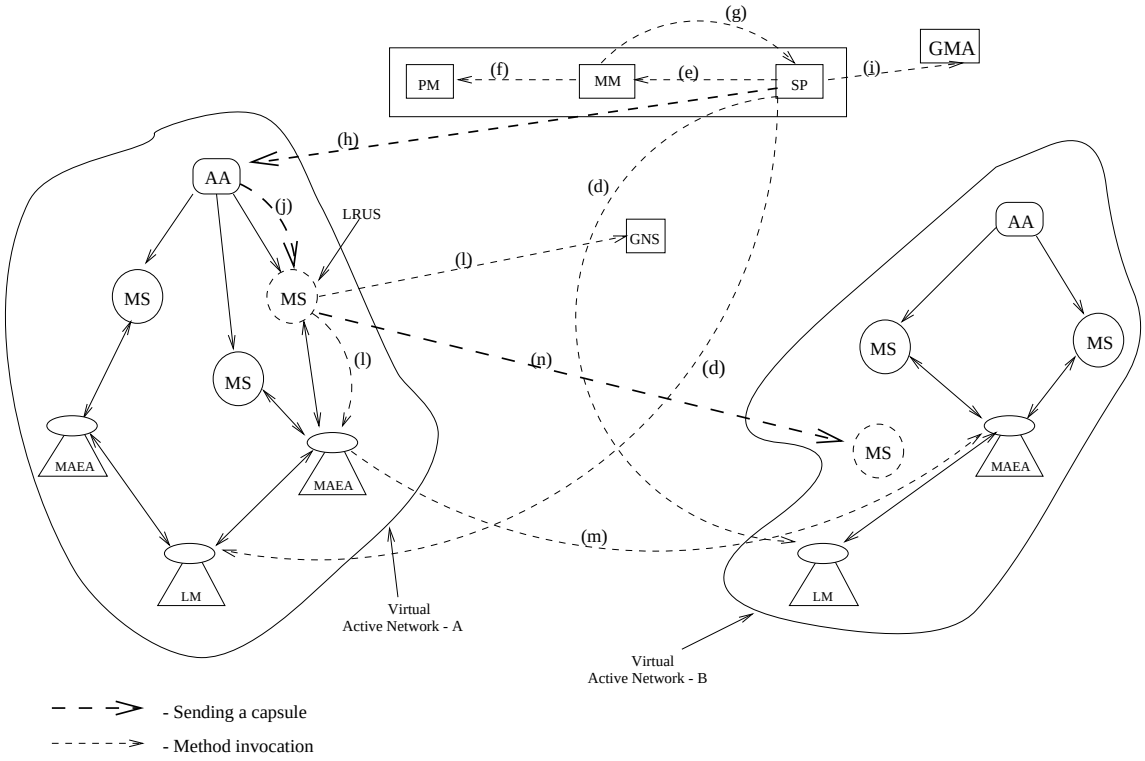

Figure 5: Migrating the LRUS to attend the customer requirement (case 2).

If there are more than one service copy in the network and they are migrating few times, some copies can be deleted from the network. Thus, the number of service copies in the network can change according to the intensity of the use.

In addition to the four scenarios above, the customer's AA can also remove services from its VAN, migrate a service already added to its own VAN from a host to another (for instance, for load balancing), and extinguish a VAN and its services. In our model, a migration between VANs only occurs when a customer from a VAN is requiring an external service from another VAN and all policies are satisfied, as shown by the scenario 3.

4 Some Issues about the Implementation

In this section we present some aspects about the implementation. To create the environment with VANs and services migrating among them, we used the ANTS toolkit. A network based on ANTS consists of a group of connected nodes and each group's element runs the ANTS environment. We have mainly used the following classes from ANTS:

- *Application*: in order to allow the customer to install and use services in the active nodes. This class is inherited by the customer for creating the AA according to the customer's requirements;

- *Capsule*: to send and receive code to and from the nodes. Communication between customers and SP is by means of capsules. Customers use the services by sending capsules to the nodes where the services are located into the VAN. The capsule to install a new service into a node is presented in Appendix A;
- *Node*: this class represents the execution environment of a VAN in a host.

The active environment and the management system have been developed using Java 1.2. The services are implemented as Java objects. The mobile agent platform is the Grasshopper 2.1, and the active nodes of each VAN are running on Sun Workstations executing SunOS 5.5. A service has a single name in the network and it can be only identified by this name. This name is generated by the concatenation of the VAN name, the host name where the service is located, and the name given by the service provider. For instance:

- VAN name: **VAN2**;
- host name where the service is located (ANTS style): **1.1.1.3**;
- name given by the service provider: **serviceA**.

Full final name = VAN21.1.1.3serviceA.

The generated name is single and it is registered and located in the GNS. The GNS is a naming service developed using JNDI (*Java Naming and Directory Interface*).

Figure 6 depicts a typical management system use after a management operation. It is performed for getting a general view of the domain including the hosts with potential problems and the monitoring information about the accounting and performance areas. Based on this information, the manager can take actions collecting more specific data about a VAN, a host or a service.

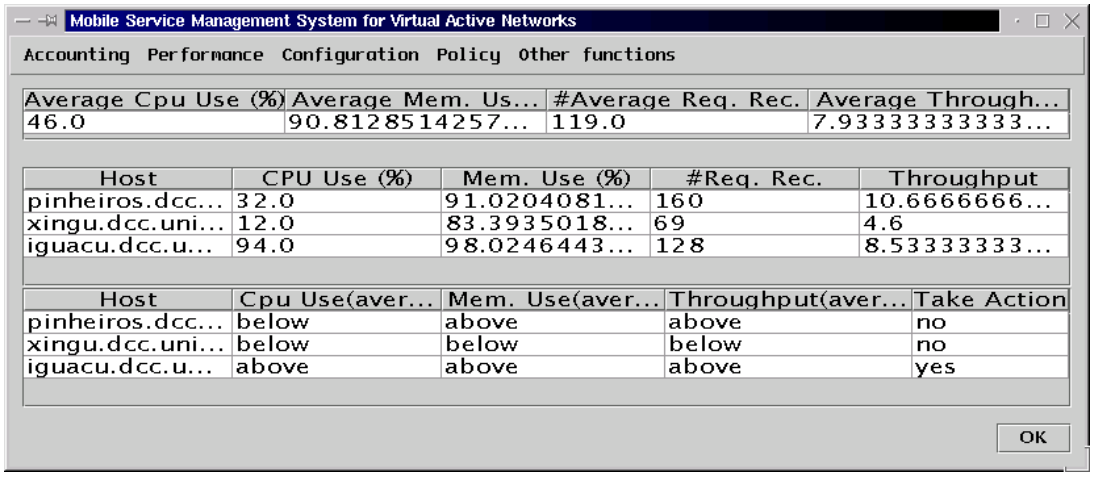

Figure 6: Management Remote Application in a general view.

A more specific management operation is shown by Figure 7. In this case, the operation is only applied to VAN2 to get the most used service in that VAN.

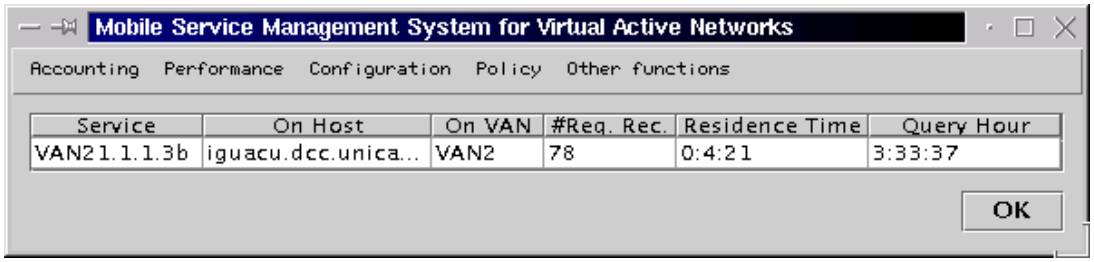

Figure 7: Management Remote Application getting the most used service in VAN2

5 Telecom Applications

Up to now we have concentrated on the infrastructure for service provisioning and management. Now we illustrate the capability our system has for achieving management of Telecom environments based on active networks. To validate the system, we have tested it on two typical Telecom services, *Call Forwarding* and *Virtual Private Network*, presented in [2].

5.1 Active Network-based Call Forwarding Service

The active network-based Call Forwarding service enables users to initiate an automatic routing of incoming calls to other destination devices, depending on the time of day or specific events [2]. We have managed this service in order to validate the capability our model has to manage mobile services. We consider that a customer has installed and configured the *Call Forwarding* service in his VAN following his preferences. Furthermore, we are considering that the customer is mobile and, thus, can migrate from a VAN to another. This feature is very common to environments which consider mobile users and an example of it can be found in [6]. Taking into account this scenario, the *Call Forwarding* service has to be external in order to migrate as the customer moves to other VANs.

We have created two VANs denominated VAN1 and VAN2 to simulate the scenario. Based on this, a customer located in the host 1.1.1.5 (xingu) belonging to VAN2 installs the *Call Forwarding* service in his node. Afterwards, the customer migrates some times after installing the service in his node. Past some time, we invoked a management operation for gathering information about the *Call Forwarding* service. Figure 8 depicts the *Call Forwarding* service migration and some accounting information in each host. In this management operation, the *Call Forwarding* service is named *VAN11.1.1.2CallForwardingG*.

We can see that the *Call Forwarding* service has migrated between VAN1 and VAN2 in order to attend the customer roaming. Initially, it is in the host 1.1.1.5 (xingu) belonging to the VAN2. After some migrations, it is in the host 1.1.1.2 (pinheiros) belonging to the VAN1. Also, below each host, we can see some information like the number of received requests and residence time.

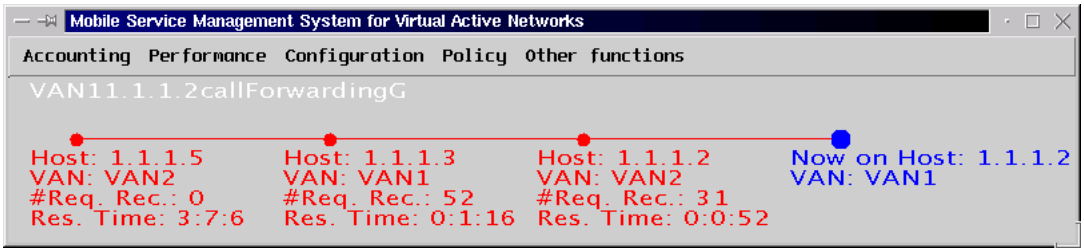

Figure 8: CallForwarding service migration.

5.2 Active Network-based Virtual Private Network Service

The VPN service enables different organizations or companies to create private networks by using public network resources. In our model, a VPN can be created from different VANs becoming a *Virtual Private Active Network* (VPAN). This is a common case when different organizations or companies located in different domains want to define a private network among them. So far, we have managed a single domain, but in the following we describe some extensions in order to allow our system to manage a VPAN belonging to two or more domains.

Taking into account that the model is flexible, we only need to add some functionalities to the GMA in order to achieve the new management facility. We consider that the management system is used in each domain, so that, there is a GMA for each one and they communicate one to another to install and manage the VPANs.

In this new kind of scenario, a customer can choose the topology of the VPAN informing a host from his local VAN and specifying other VANs located in different VAN provider domains in order to connect them into a single VAN for building a virtual enterprise. The VPAN installation has some different steps to follow if compared to traditional single domain scenarios. In this new case, the customer contacts the GMA of the local domain to start the VPAN installation. The local GMA interacts to the GMAs from other domains in order to define and negotiate the provisioning of the VPAN. If they achieve an agreement, the VPAN is installed according to what they have negotiated. Consequently, the companies/organizations can send and receive services from one to another and the service provider from each domain can reach many end users in a very short time. A service management functionality needs to be added to the GMA for attending the management of this new scenario. To get information about the VPANs (accounting and performance data), the management system from a domain needs to contact the management systems from the other domains. This communication among them is only made through the GMAs. Figure 9 depicts this scenario in which a manager from the domain 2 wants to collect monitoring information about the VPAN.

The figure shows that the VPAN is created from domains 1, 2 and 3. A management operation is initiated from domain 2 for collecting data about the VPAN (1). The domain 2's GMA interacts to the other GMAs for getting VPAN management information (2). Finally, the GMA, which has started the management operation, returns back the results to the

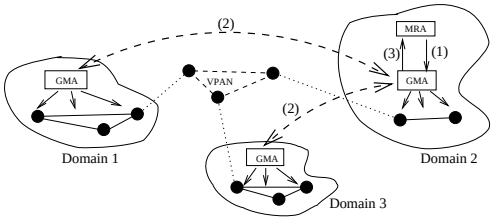

Figure 9: Steps to get management information from a VPAN belonging to three different domains.

MRA (3). Each GMA has information about the VPANs which it belongs to. Furthermore, every management system located in different domains can initiate a management operation over the VPAN. The only restriction is the necessity for contacting GMAs from different domains in order to get management information about that part. Figure 10 depicts the MRA after a management operation for displaying the local VANs (VAN1 and VAN2) and the VPANs which the domain 2 belongs to. The VPAN1 includes the host (1.1.1.2) from the local domain (domain 2) and other two domains (domains 1 and 3).

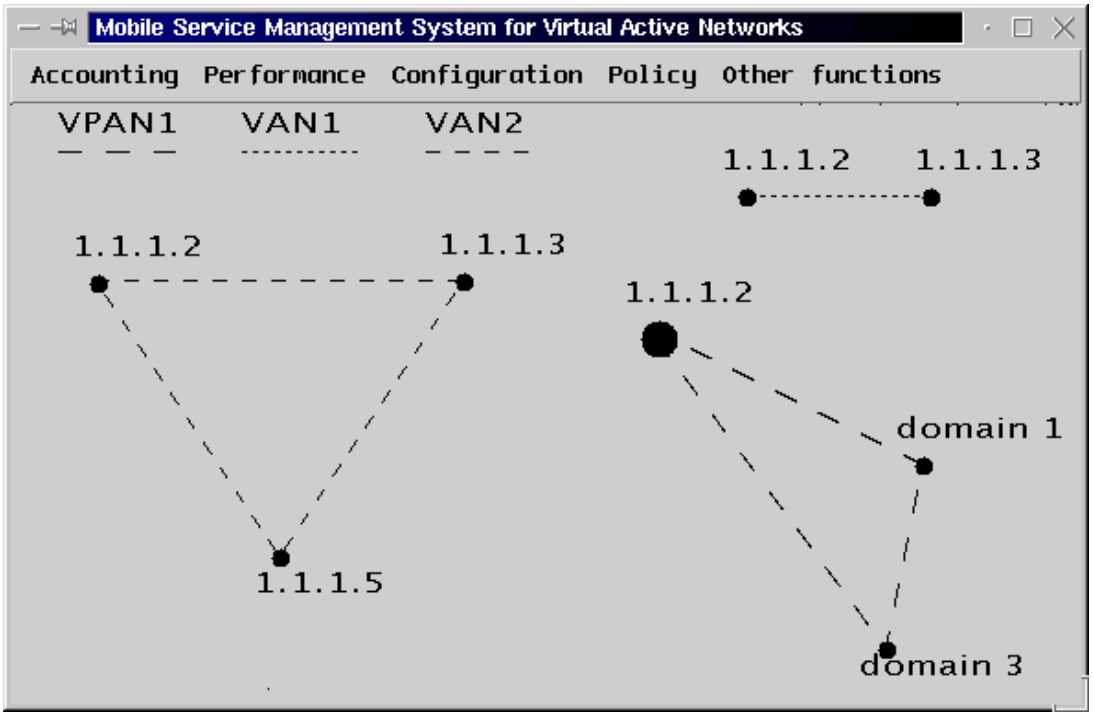

Figure 10: MRA after collecting information about VANs and VPANs.

6 Conclusion and Future Work

In this paper we present a framework for service provisioning, such as VAN creation and services installation in Telecom environments based on VANs. The framework also performs the management of these services and their migration. The environment has a Service Provider, and a Policy Manager responsible for controlling and processing policies.

The proposed management model is based on a mobile agent platform and considers three management functional areas: accounting, performance and configuration. We have used a mobile agent plataform due to its facilities to send management agents to the hosts. This model is flexible in the sense of management questions can be answered from different levels: per service, per host, per VAN, and per domain.

We create some scenarios to simulate a Telecom environment: exploiting the VAN creation, installing mobile services, and managing a service migration. The scenarios were created using the ANTS toolkit which has sufficient features to implement our experimentation environment. Furthermore, we applied our system for managing two Internet-based Telecom services, Call Forwarding and Virtual Private Network. The former considers mobile users so that customers and services can migrate from a VAN to another. The latter presents how our management system can be used for multi-domain management. The implementation has showed that the proposed framework can handle Telecom environments with mobile services and VANs successfully, opening up a potential further study.

We have concentrate our attention in resolving both problems, service provisioning and management. It is important that a service provider offers not only services but also performs some management functions. The framework we have developed can be used either by the SP in order to manage its VANs or by the customer who wishes to manage his VAN.

Even though the framework is responsible for both service provisioning and management, there is a well defined separation between provisioning and management. Thus, it is easy to add new management functions, e.g., VPAN management as shown in Section 5.2. On the other hand, the framework is flexible to introduce new functionalities for service provisioning whereas we only need to modify the specific component to do it. For instance, if new policies are created, only the Migration Manager (MM) and the Policy Manager (PM) will be changed in order to consider those novel policies.

In a customized service environment, the customers can develop services in different programming languages like Java, C++, Visual Basic, among others, requiring a solution for interaction among these different components. In order to take account this issue, in future works, the CORBA objects are also introduced whereas in this work we have only considered Java objects (services). In this way, we intend to integrate a new naming service for supporting these CORBA objects. The used Grasshopper mobile agent platform supports MASIF (*Mobile Agent System Interoperability Facility*) interfaces for management of these new services. In order to improve the service provisioning, we plan to introduce a negotiation step between clients and the service provider. With this new functionality, both client and provider could define some agreements on QoS (*Quality of Service*). Finally, although currently we have only one SP, we intend to have more than one service provider offering same or different services.

Acknowledgements

The authors would like to thank CNPq, CAPES and PRONEX SAI for their support.

Appendix A
Capsule to install a new service into a node

```
package mgmtsw;

import ants.*;
import java.util.Hashtable;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.*;
import java.rmi.server.*;

public class InstallServiceCapsule extends DataCapsule {

    protected String serviceName,agentName,className,
                    hostDestFormatAnts,shortServiceName;
    InetAddress hostAddress;
    public boolean newService=false;
    int aplicationAddr;

    final private static byte[] MID = findMID("mgmtsw.InstallServiceCapsule");

    protected byte[] mid() {
        return(MID);
    }

    final private static byte[] PID = findPID("mgmtsw.InstallServiceCapsule");

    protected byte[] pid() {
        return(PID);
    }

    public int length() {
        return(super.length()+Xdr.STRING(serviceName) +
            Xdr.STRING(agentName) +
            Xdr.STRING(className)+Xdr.BOOLEAN+Xdr.INT+
            Xdr.STRING(shortServiceName)+Xdr.STRING(hostDestFormatAnts));
    }

    public Xdr encode() {
        Xdr xdr = super.encode();
```

```
    xdr.PUT(serviceName);
    xdr.PUT(agentName);
    xdr.PUT(className);
    xdr.PUT(newService);
    xdr.PUT(applicationAddr);
    xdr.PUT(shortServiceName);
    xdr.PUT(hostDestFormatAnts);
    return (xdr);
}

public Xdr decode() {
    Xdr xdr = super.decode();
    serviceName = xdr.STRING();
    agentName = xdr.STRING();
    className = xdr.STRING();
    newService = xdr.BOOLEAN();
    applicationAddr = xdr.INT();
    shortServiceName = xdr.STRING();
    hostDestFormatAnts = xdr.STRING();
    return(xdr);
}

public void setNameService(String s){
    serviceName = s;
}
public void setClassName(String s) {
    className = s;
}
public void setAgentName(String agentName) {
    this.agentName = agentName;
}
public void setAddressOfApplicationDest(int address) {
    applicationAddr = address;
}
public void setDestFormatAnts(String dest) {
    hostDestFormatAnts = dest;
}
public String getDestFormatAnts() {
    return hostDestFormatAnts;
}
public void setShortServiceName(String shortName) {
    shortServiceName = shortName;
}
public String getShortServiceName() {
```

```
    return shortServiceName;
}

public boolean evaluate(Node n) {
    if (n.getAddress()==getDst()) {
        try {
            hostAddress = InetAddress.getLocalHost();
        }catch (UnknownHostException e) {
            e.printStackTrace();
        }
        String hostIP = hostAddress.getHostAddress();
        String rmiurl = "rmi://"+hostIP+":2000/"+serviceName;
        try {
            Class serviceClass = Class.forName(className);
            ManagementInterface s =
                    (ManagementInterface)serviceClass.newInstance();

            s.setAgentName(agentName);
            Naming.rebind(rmiurl,s);
            n.getCache().put(serviceName,s,60000);
            BindInterface s1 =
            (BindInterface) Naming.lookup("rmi://araguaia:2000/register");
            s1.bind(rmiurl,serviceName);
            if (newService) {
                s.notifyMAEA(serviceName);
                ServiceMigrationCapsule c = new ServiceMigrationCapsule();
                c.origem=2;
                c.setShortServiceName(shortServiceName);
                c.setDestFormatAnts(hostDestFormatAnts);
                c.setDst(applicationAddr);
                c.prime(this);
                return n.routeForNode(c,c.getDst());
            }
        }catch (Exception e) {
            e.printStackTrace();
        }
        return(true);
    }else
        return(n.routeForNode(this,getDst()));
}

public InstallServiceCapsule() { }

public InstallServiceCapsule(short sa, short ds, int na, ByteArray da) {
```

```
        super (sa,ds,na,da);
    }
}
```

References

[1] M. Breugst and S. Choy. Management of Mobile Agent Based Services. *Intelligence in Services and Networks (IS&N'99),* Barcelona, Spain, Springer, pp. 143-154, April 1999.

[2] M. Breugst and T. Magedanz. Mobile Agents - Enabling Technology for Active Intelligent Network Implementation. *IEEE Network,* pp. 53-60, May/June 1998.

[3] M. Brunner, B. Plattner, and R. Stadler. Service Creation and Management in Active Telecom Networks. *Communications of the ACM,* pp. 55-61, March 2001.

[4] M. Brunner and R. Stadler. The Impact of Active Networking Technology on Service Management in a Telecom Environment. In *Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, USA, 1999.

[5] K. L. Calvert, S. Bhattcharjee, E. Zegura, and J. Sterbenz. Directions in Active Networks. *IEEE Communications Magazine*, 1998.

[6] W. Chang and R. Popescu-Zeletin. Mobile Agent based Service Provisioning in Integrated Networks. *IEEE International Workshop on Autonomous Decentralized Systems,* pp. 19-27, 2000.

[7] A. Campbell et al. A Survey of Programmable Networks. *ACM SIGCOMM. Computer Communication Review,* pp. 7-23, April 1999.

[8] D. L. Tennenhouse et al. A Survey of Active Network Research. *IEEE Communications Magazine,* pp. 80-86, January 1997.

[9] P. Farjami, C. Görg, and F. Bell. Advanced Service Provisioning based on Mobile Agents. *Computer Communications 23 (8) (2000), Elsevier,* pp. 754-760.

[10] A. Fernando, D. Williams, A. Fekete, and B. Kummerfeld. Dynamic Service Installation in an Active Network. *Computer Networks 36 (1) (2001), Elsevier,* pp. 35-48.

[11] J. A. Kornblum, D. Raz, and Y. Shavitt. The Active Process Interaction with its Environment. *Computer Networks 36 (1) (2001), Elsevier,* pp. 21-34.

[12] I. W. Marshall, H. Gharib, J. Hardwicke, and C. Roadknight. A Novel Architecture for Active Service Management. *Seventh IFIP/IEEE International Symposium on Integrated Network Management,* Seattle, Washington, USA, May 2001.

[13] A. Patel. Current Status and Future Directions of Software Architectures for Telecommunications. *Computer Communications 25 (2) (2002), Elsevier,* pp. 121-132.

[14] D. Raz and Y. Shavitt. Active Networks for Efficient Distributed Network Management. *IEEE Communications Magazine,* pp. 138-143, March 2000.

[15] F. L. Verdi and E. R. M. Madeira. A Mobile Agent-based Model for Service Management in Virtual Active Networks. *IFIP/IEEE 12th International Workshop on Distributed Systems: Operations & Management - DSOM'01,* Nancy, France, pp. 101-112, October 2001.

[16] D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *IEEE OPENARCH'98*, San Francisco, CA, April 1998.