A Mobile Agent-based Model for Service Management in Virtual Active Networks

Fábio Luciano Verdi† and Edmundo R. M. Madeira

Institute of Computing, University of Campinas (UNICAMP), Campinas-SP, Brazil
Telephone: +55 (19)-3788-5862
Fax: +55 (19)-3788-5847
e-mail: {fabio.verdi, edmundo}@ic.unicamp.br

Active Networking is an expanding field of research in towards a new way to provide services for customers. The facility to develop and put customized services in the network enables a new management architecture to support this evolution. In a telecom environment, for example, the services can be purchased according to the customer's requirements. Furthermore, a lot of different services is offered by service providers. In this work, we outline a mobile agent-based model for service management in Active Networks considering three of the five management functional areas: accounting, performance and configuration. The paper exploits the management of *Virtual Active Networks - VANs*, where the services can migrate from a VAN to another one, and presents a model and its components we have proposed for mobile service management in this context.

Keywords: Management of Virtual Active Networks, Mobile Service Management, Mobile Agent, Policy-Based Management, telecom.

1 Introduction

The flexibility that new Internet services are being developed presents a new challenge for the network management area. With the introduction of the active networking technology, the packets carry program codes in addition to data. These packets are called *active packets or capsules* and the code can be a program (service) or can instantiate new applications (Extensions) in the nodes [D.97]. A network node is responsible for executing the applications developed by users or service providers. Such applications are called in some cases *Active Services* that can be sent to the network nodes for satisfying customers and enabling dynamic services moving according to the users demand. Also, each user can develop services and put them inside the network in a customized way.

The nodes such as routers, switches, caches and mirrors are able to receive active services to be executed. The execution environments located in these devices manage basic tasks such as scheduling, access to the resources, security policies, and sending and receiving services in that node. The underlying network user can create an active network choosing the hosts and services needed. In a telecom environment, providers must offer a set of services that can be purchased according to the customer's requirements. This interaction between a customer and a provider can be easily reached by using active networks. An active network can be separated into *Virtual Active Networks - VANs*, allowing a customer leases it from a provider. On such VAN, the customer can install, configure and run active services. The provider must have capabilities for accessing the customer domain to perform management basic tasks related to creation, supervision, updating and removal of a service on a networking platform [MR99]. In this context, the management of these services, which are statics or can be moving inside a virtual active network or among virtual active networks, is necessary. Information about accounting, performance and configuration enables the manager

†Supported by CAPES and PRONEX Project

(human) to know the behavior of the managed environment to detect problems and take actions on them. In order to get this information, we have proposed a mobile agent based-model for service management in virtual active networks.

The scenario where the services can be moved among VANs is a very common one in telecom environments. Furthermore, aspects about policy are important in this context whereas a lot of services can be developed by different providers and by different customers, so that, each service has its own policy based on the domain it was created. The importance of management in this kind of environment is evident in the sense of many undesirable situations can be generated which affect the environment as a whole. The integration of static and mobile services, VAN, and policy currently reflects a perfect telecom environment.

Our model takes account the capacity a service has for migrating and we define a scheme on how the mobile agents are structured for collecting management data. The model is innovative in the sense of we are considering the migration of a service from a VAN to another one in order to satisfy the customer's demand. For supporting these features, we have developed the model based on a mobile agent platform for gathering management data using the facilities that a mobile agent technology offers. We have developed a telecom environment using the ANTS toolkit 1.3.1 [JVL98] for simulating some of the common telecom scenarios. The environment has a *Service Provider - SP*, that offers the services a customer can download, and a *Policy Manager - PM*, that stores and manages policies. A prototype has been implemented to validate our model supporting accounting, performance and configuration management. The management system we describe successfully offers functions for monitoring telecom environments by combining mobile active services, Virtual Active Networks, policies, and mobile agent technology.

The paper is organized as follows. In the next section some concepts about active networks and virtual active networks are presented, and some related works are commented. Section 3 outlines the model and its components, and illustrates some possible scenarios. Section 4 describes some aspects about the implementation. Section 5 presents the conclusion and gives an outlook on further works.

2 Active Networks and Virtual Active Networks

Active Networking breaks with the tradition where the nodes only route packets according to their routing tables. The nodes in active networks can receive and run external programs using computational resources for processing them. In this paper, the active nodes are called *Execution Environments - EEs*, following the terminology of the AN working group [Gro99].

The active networks are classified considering some basic architectures. There is an approach called *Active Packets* in which active code is carried by packets either to be executed on the data of the same packet that carries the code, or to be executed in order to change the state or the behavior of the node. Another particularly approach is called *Active Nodes* in which the packets do not carry the actual code, but instead carry some identifiers or references to predefined functions that reside in the active nodes [K.99]. By using active packets, an active network can carry programs and not only data and when the packets arrive in a node, its processing can change the node's local state. Some architectures for active networks enable a capsule to install applications which will stay in that node for a finite time. On the other hand, the capsule itself can execute some specific tasks when it is evaluated. We are considering both cases, i.e., a capsule can itself execute some code or the capsule can install some application on the node. The processing of the installed applications and capsules can generate other packets which will be send to other nodes through the active network, opening a new way for installing services on the network. The provider can offer an abstraction called VAN in the sense of the customer can install, configure and run network services without further interaction with the provider. Thus, a VAN can be described as a graph of virtual active nodes connected by virtual links transporting active packets internally and among different virtual active networks [MR99]. The VAN is a very useful solution for separating a customer from another one and to provide specific services for each customer's domain. In our model, we create VANs installing more than one Node (EE) object of the ANTS toolkit per host, so that, a single (physical) active node can run several virtual active nodes belonging to different VANs. In this way, a customer is isolated from another one and the active network resources are shared by them.

Many approaches taken today have focused on active networks. In [DY00] active networks are used

for management of devices in the network and in [RR00], a proposal of active distributed management is presented. In the latter, some typical management algorithms are developed and are available as patterns to be used by the management programs. In [MS99], a model for management of mobile agent-based services is described. An architecture for management of a network that offers active services is presented in [WHJC01]. This architecture uses a combination of policies and adaptive algorithms allowing multi-user management of network based service components. Service management in telecom environment is presented in [MR99, MR00]. Our paper assumes an infrastructure that supports VANs and services moving among them.

3 The Management Model

The proposed model for service management in active networks is based upon a mobile agent platform. The agents can easily execute tasks on behalf of other entities such as a human manager or an external application. The proposed model has two kinds of management agents. The first one is a static agent responsible for accounting area. The second agent is mobile, whereas it is sent by the manager for collecting performance data in a node. After collecting, it returns to the management node. Getting the responses about the accounting and performance, the manager (human) can take actions towards the configuration functional area. In the following, we present the information related to the questions we are interested to answer, the management model components, the management information bases, and the management scenarios.

3.1 Management Information

The OSI (*Open Systems Interconnection*) management model [Y.93] defined five management functional areas: FCAPS (*Fault, Configuration, Accounting, Performance, Security*). The proposed model in this work approaches three of these five functional areas: accounting, performance and configuration. The management questions we are interested to answer can be divided in four cases:

- per service;
- per host;
- per virtual active network; and
- the set of all virtual active networks of a single network (the network as a whole).

According to the accounting functional area, we are interested in answering some of the following questions:

1. What are the most and least required objects (services) per host, per VAN and in the network as a whole.
2. What is the number of requests to a specific service.
3. What are the most and least required hosts per VAN and in the network as a whole.
4. What is the number of requests to a specific host.
5. What is the service residence time.
6. What is the throughput in a period of time t per host and per service.

According to the performance functional area we are interested in answering some of the following questions:

1. What is the cpu and memory use in a host in a period of time t .
2. What hosts exceed the specified limits of the memory and/or cpu use per VAN and in the network as a whole.

The set of information gathered allows to identify problems in a more general way. The manager can collect more specific information about a host, a service or a VAN. In this case, the manager can desire to send another agent for collecting performance data and/or the manager can invoke remote methods for collecting accounting data.

Based on the information above and based on the policy each service has, we have defined three kinds of possible actions to apply, as follows:

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

We have created three representative policies for the model, but other policies can be developed and inserted into the database in a flexible way for satisfying specific customer's domains. Each provider or each service developer can define properties for using the services. When a new service is implemented and registered in the SP, its policies are included into the policy database. The Policy Manager (see forward in Section 3.2) is responsible for applying the policies in the environment. Then, we have defined the following policies:

1. A service has minimum and maximum bounds of times for migrating in a period of time.
2. A service has a limited number of copies in the network.
3. *Less Recently Used Service - LRUS*.

   This policy is related to received requests in a period of time and it is used when a customer requires a new service. The less recently used service will be the candidate for migrating to the destination host for satisfying the customer's requirement. If policy 1 does not allow the LRUS migration, then a new service copy can be created, based on the configuration action 2. But, if policy 2 does not allow that a new service copy is created, the system manager (human) can decide what action takes.

As we can see, the events generated by policies are treated using configuration actions. When some event related to policies is detected by the management system, the correct configuration action must be taken. For example, when the maximum bound for migrating is reached, the management system can create a new instance of a service. When the service is required for migrating, the system management looks for the LRUS in the network as a whole. When an overload is detected, the management system can move a service to another node for load balancing. Other situations like these can be interpreted.

The questions above represent what we have considered to implement, but the model is open and flexible for defining other questions and collecting other data from the environment.

3.2 The management model components

In this section we present the agents and the other components like the Policy Manager and the Migration Manager. We give a brief description about each one and their roles on the model. Thus, we have proposed the following components as shown in Fig. 1:

- **Global Manager Agent - GMA**: It is the centralized manager of the model. There only is a GMA in the network as a whole, i.e, in a domain. Interactions between domains are out of the scope of this work. This application presents to the manager (human) the interface with the methods related to management questions (accounting, performance and configuration). This component is responsible for analysing accounting data, performance data and configuration data on all VANs and its services. Furthermore, it notifies the MM (*Migration Manager*, see below) for sending performance agents to specific hosts. The GMA has a total view of the network as a whole and together with MM and PM it can offers a large set of management information to the manager.
- **Local Manager - LM**: There is an LM per Virtual Active Network. It is responsible for management of the VAN and collects data of each MAEA (*Managed Active Element Agent*, 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 Virtual Active Network. This agent is an interface between the ***Managed Service - MS*** and the management system and it has the following tasks:
  - Storing the events of each service it manages.
  - Activating and deactivating the management filter 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 agents about services are arriving.
  - Processing events which has been earlier defined, or notifying the LM about unrecognized ones.

- **A Policy Manager - PM**: This component is responsible for controlling, insertion, updating and removal of a policy in the database. Thus, any event about a service must be notified to the policy manager. The policies about the service are processed in order to analyse the situations defined in the early section.
- **A Migration Manager - MM**: When a new virtual active network is being created, the services in those hosts must be managed and, therefore, this component sends management agents to these ones. The SP (Service Provider, see below) always notifies the MM about the events on the network and, hence, the MM has to exchange information with the Policy Manager about the policies. Also, this component is responsible for sending performance agents for collecting performance data. This component, the GMA, and the PM perform the function of a *Management Center - MC*.
- **A 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.
- **A Service Provider - SP**: It is responsible for offering services to customers. When a client sends a capsule to the SP for creating a VAN and its services or for adding a new service to his/her VAN, the SP notifies the GMA and the MM. Also, in a service migration or removal, the SP notifies the MM for controlling its migration tables and notifies the GMA for updating its local information about the VANs. In some cases, the SP can create a new service copy as shown in Section 3.4.

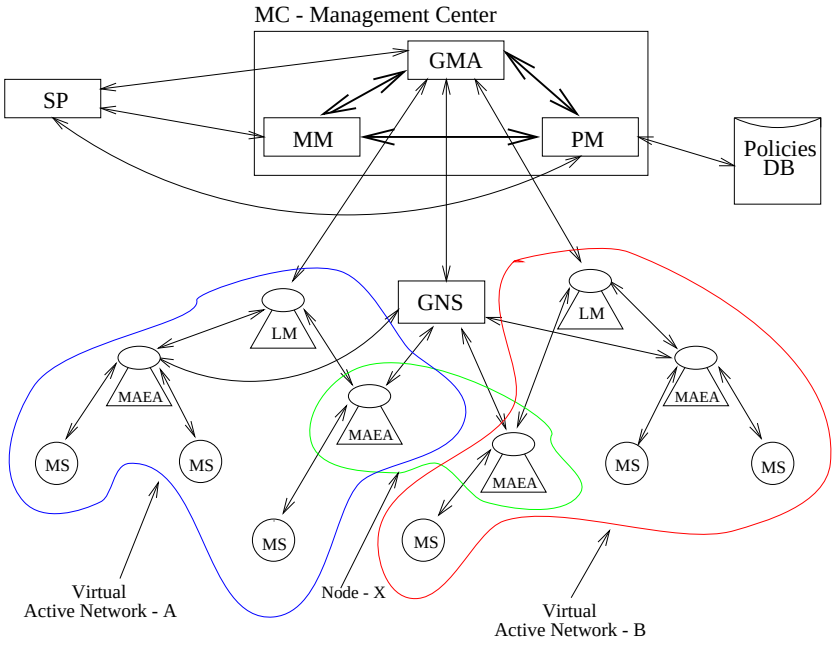

Fig. 1: Management Model Components and their relationship

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

a host in a virtual active network environment can be part of more than one VAN at the same time. The interaction between agents (typically MAEA) of different VANs is not shown, but an agent from a VAN can access another agent from another one for changing data, since the customer domain policies allow this interaction. In our model, a MAEA from a VAN changes data to another MAEA from another VAN only on an inter-VAN service migration. To collect accounting data the sequence, shown in Fig. 2, is as follows:

1. The manager (human) invokes methods on the GMA.
2. The GMA notifies each LM for starting to collect data. After this point, all management is done in a parallel way and each LM gathers data from its VAN. The GMA waits all LMs send the answers.
3. Each LM invokes methods on the MAEA and waits all these ones answer. After this, the LM sends back the results to the GMA.
4. Each MAEA invokes methods on the services in order to get the most recent data and sends back the results to the LM.

The existence of different management levels facilitates the coordination among management agents.

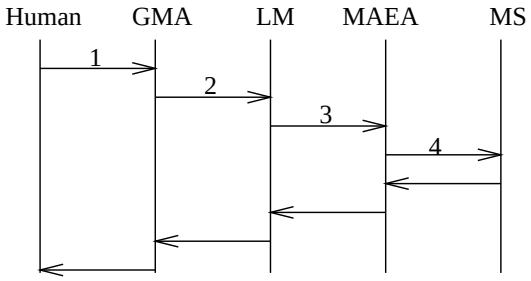

Fig. 2: Sequence Diagram for collecting accounting data

To collect performance data, unlike accounting data, the management system sends mobile agents to the hosts where the services are located (1). They stay on the nodes during a period of time (2) and return back to the management center with the data (3). The management system sends an agent for each host, so that all services are managed at the same time. Fig. 3 illustrates the management system behavior in this case.

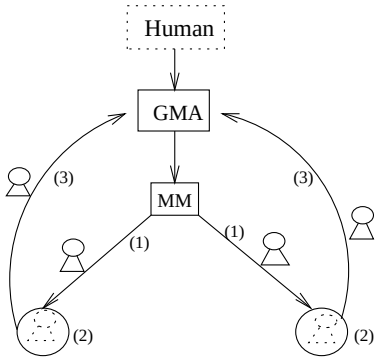

Fig. 3: Sending mobile agents for collecting performance data

When the manager decides to send agents for gathering performance information, the management system starts collecting accounting data. Hence, at the end of the period, the management center can also provide to the manager a set of information about the environment. Based on accounting and performance information, the manager will be able to take actions towards a possible reconfiguration.

3.3 Management Information Bases - MIBs

The proposed management model needs to maintain management information at different levels: MAEA, LM and GMA. In this work, the bases used to store this information are called MIBs.

In this section we present what kinds of information are located in each MIB and what questions can be answered by each one. The managed service has a management interface enabling some filter to be activated or deactivated. The filters are related to the specific attributes for management and they can count or not the metrics in different periods of time. At each level, some questions can be answered based on the data gathered up to that level. The Tables I through III show the questions answered in each level of management. The questions are those in the Section 3.1 organized per service, per host, per VAN and in the network as a whole. We label *A.question number* for accounting questions and *P.question number* for performance questions, for example: A.1 means question 1 from accounting area. Thus, we present the following MIBs:

1. MAEA $\rightarrow$ MS: This is the lower level of management. The metrics in this MIB are: (a) received requests and (b) residence time. Tab. I shows the questions that can be answered with these metrics.

Tab. I:

per service	per host	metric
	A.1,A.4	a
A.2		a
A.5		b
	A.6	a,b
A.6		a,b

2. LM $\rightarrow$ MAEA: The metric in this MIB is: (a) received requests. Tab. II shows the questions that can be answered with this metric.

Tab. II:

per VAN	metric
A.1, A.3	a

3. GMA $\rightarrow$ LM: This is the upper level of management. Here, the manager can have a general view about the network as a whole. The metrics in this MIB are: (a) received requests, (c) memory use and (d) cpu use. Tab. III shows the questions that can be answered with these metrics.

Tab. III:

per host	per VAN	whole net.	metric
		A.1,A.3	a
P.1			c,d
	P.2		c,d
		P.2	c,d

In addition to information about accounting and performance, in this MIB there is information about the service migration: source host, destination host, what time the migration happened and how many times it has migrated. All this information is useful when some service wants to migrate and, then, the policies can be applied to the service.

In our proposed model, the metrics can be collected all the time, in some periods of time, or not be collected. Collecting management data all the time for all metrics in the network as a whole, is not a convenient option because the MIBs' size in each level of management can become too large. A typical use of the management system is for collecting some metrics during a period of time to detect some potential problems. After getting these metrics, the management operations are only to collect data related to the specific problem.

3.4 Management Scenarios

The management scenarios we have created are related to a telecom environment. The scenarios are associated with the customer's *Active Application - AA* , that is an interface for the customers using the available environment. Thus, clients that have an active application in their domain can perform the following functions:

1. **Creating a new virtual active network specifying what hosts and what services are required**.
   In this scenario the following sequence is needed, as shown in Fig. 4:

   (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 management agents to the hosts and defines the virtual active network manager (LM).
   (e) Services register themselves in the GNS.

   Note that in this case the PM is not notified. In fact, when a new VAN is created, the SP always allows to provide new services. The policies are applied to services which are required after creating VANs.

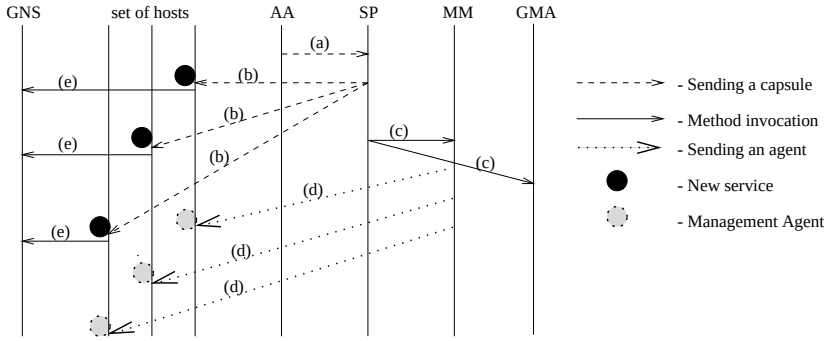

Fig. 4: Customer creating a new VAN.

2. **Adding a service to the VAN**.
   In a telecom environment the customer frequently needs to add new services to the VAN. This is the main scenario that can be created whereas there are many steps before a service migrates to another node. In this scenario, the LRUS can belong to the other VAN (this is the normal case) and, then, the service must migrate between VANs. This a way to coordinate several instances of the same service. The sequence, shown in Fig. 5, is as follows:

   (a) Customer sends a capsule to the SP indicating what service is required and to what destination host the service will be send.
   (b) The SP gets the LRUS in the network. In this search, all the information about the LRUS also is available.

(c) SP interacts with the MM for verifying if migration is possible.
(d) MM gets the service policies from PM for analysing them.
(e) MM answers to the SP about the possibility of migration.
(f) If migration is possible, SP sends a capsule to the active application of the VAN where the LRUS is located at this moment. This is necessary for the active application stops to use the service in that VAN. On the other hand, if migration is not possible (according to policy 1), and the policy 2 is satisfied, the SP sends a capsule to the destination host for creating a new instance of the required service (f'). The steps *l, m* and *n* are the same for both cases. But, if the policy 2 is not satisfied, the manager (human) has to decide what action takes in this case.
(g) If a new service copy was created, the SP notifies the GMA and MM about this. If the migration occurred, the SP only notifies the GMA (g').
(h) The capsule is forwarded to the host where the service is at this moment.
(i) The service unregisters itself from the global naming service and notifies its MAEA that it is migrating.
(j) The capsule removes the service from this node and migrates to the new node.
(k) The source MAEA notifies the destination MAEA that a new service is arriving.
(l) The capsule arrives in the new node and registers the service on the GNS.
(m) The service notifies its new MAEA it has arrived.
(n) The capsule is forwarded to the new active application to notify that the required service has migrated. The active application can use the new service.

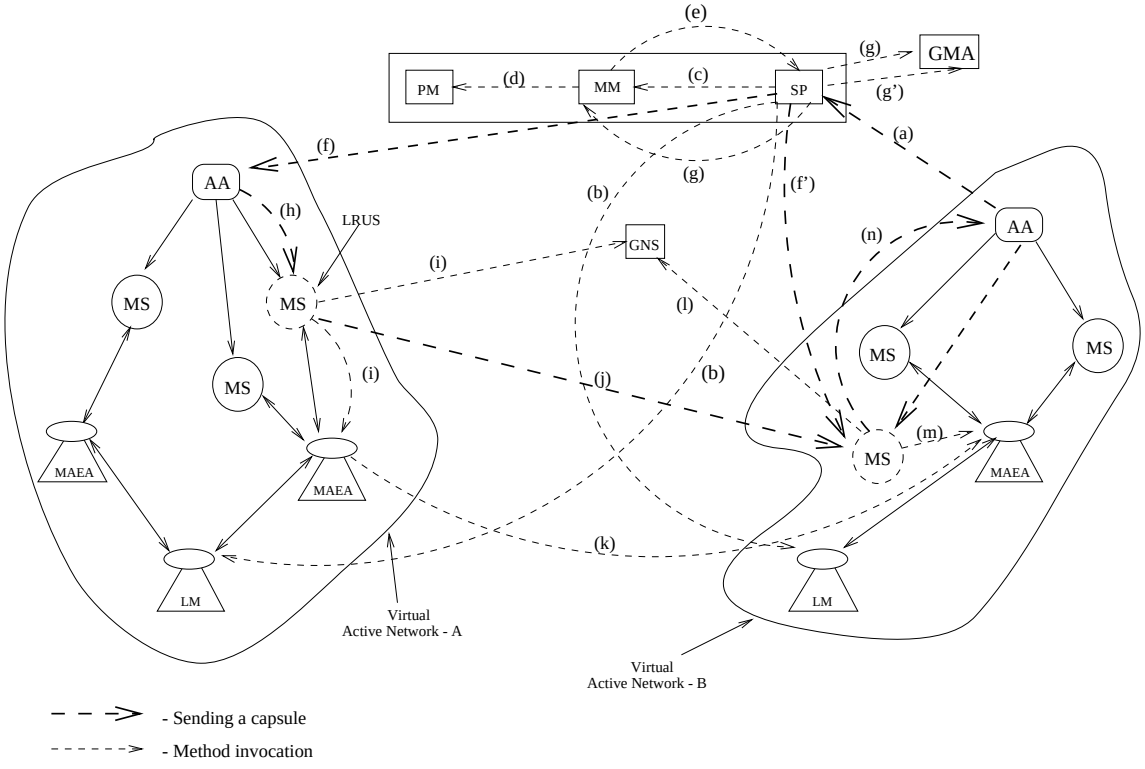

Fig. 5: Customer requiring a new service.

Each LM is responsible for finding the LRUS in its VAN and it returns back the result to the SP. If there is more than one service copy in the network and whether they are migrating few times,

some copies can be deleted from the network. Thus, the number of service copies in the network can change along with the intensity of the use.

3. **Removing services from the VAN**.
   In this scenario, the customer sends a capsule to the SP notifying the removal of the service. The SP notifies the GMA about this. Before removing the service, it unregisters itself from the GNS and notifies its local MAEA about the removal. All the service information is sent to the GMA.
4. **Migrating a service from a host to another one into the VAN**.
   The customer sends a capsule to the SP notifying about this. The SP notifies the MM and the GMA about the migration. Before migrating, the service unregisters itself from the GNS and notifies its local MAEA about migrating. When the service arrives to the destination node, it registers itself on the GNS and notifies its local MAEA.
5. **Deleting a VAN and its services**.
   The client decides to extinguish a VAN but before it has to send a capsule to the SP. The SP notifies the GMA about this scenario. All information about the services and the VAN is sent to the GMA.

Items 3, 4 and 5 represent some of the steps from the Fig. 5, whereas the item 2 is the main scenario. Migrating a service between VANs is a common task in a telecom environment, but at the same time, it has to be a synchronized sequence of phases, so that, the customer does not use the services during the migration. Likewise, the management system cannot manage the service whether it is migrating or the service does not exist anymore. In the proposed model, the customer cannot migrate a service to another VAN. This is an important issue whereas a VAN can become overloaded with a lot of services and the malicious customers can send untrusted code to the VANs. In this way, aspects on security must be addressed here, but this is not the scope of this paper. The service can migrate between VANs whether a customer from a VAN is requiring it from another one and all the policies are satisfied.

4 Implementation

In this section we present some aspects about the implementation. To create the environment with virtual active applications and services migrating among them, we have used the ANTS toolkit. ANTS is a toolkit developed to support active applications. A network based on ANTS consists of a group of connected nodes and each group's element runs the ANTS environment. The ANTS has three kinds of main components [JVL98]:

- The packets in traditional networks are replaced by capsules. These capsules refer to the processing to be performed on their behalf;
- Routers and end nodes are replaced by active nodes that run capsules and maintain state; and
- A code distribution mechanism guarantees that the routines will be automatic and dynamically transfered to nodes where they are needed.

We have used mainly the following classes from ANTS:

- *Application*: for allowing the customer installing and running services in the nodes. This class is inherited by the customer for creating the Active Application according to the customer's requirements.
- *Capsule*: for sending and receiving code to and from the nodes. The communication between the customer and the SP is through capsules. The customer uses the services by sending capsules to the nodes where the services are located into the VAN.
- *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. The active nodes of each VAN are running on Sun Workstations executing SunOS 5.5. The MIBs located in each management

level are developed using hash tables and bidimensional arrays. Each management agent is responsible for its MIB and it cannot access other ones of another VAN located at the same host. In the following, we comment some of available functions from the management interface.

1. What are the most and least services required (per host, per VAN and in the network as a whole).
2. What are the most and least hosts required (per VAN and in the network as a whole).
3. What is the throughput (per host, per VAN and in the network as a whole).
4. Sending agents for gathering performance data (per host, per VAN and in the network as a whole).
5. Collecting accounting data (per service, per host, per VAN and in the network as a whole).
6. Showing the migration table (per service and all services).
7. Moving a service for load balancing.
8. Deleting services from the environment.

The questions above are some of the possible that can be supported by the management system. More general and specific information can be extracted from the environment in a flexible way by creating other questions and gathering other metrics on the managed service. This is possible due to the model is open to include new operations in order to support a large number of domains. Items 7 and 8 are related to the configuration functional area. The last item requires an excessive search across the VANs for getting candidate services for removal. If this operation, based on the policy 1, finds some services which can be deleted, the manager (human) can decide if the service will be removed or not. Therefore, before removing services, the manager can invoke the operation 5 for collecting accounting data and be sure that a determined service can be removed and not affect the environment.

A service has a single name in the network and it can only be identified by this name. This name is generated by the concatenation of the virtual active network name, the host name where the service is located, and the name given by the service provider. For instance:

- virtual active network name: **vn1**;
- host name where the service is located: **1.1.1.2**;
- name given by the service provider: **serviceA**.

Final name = vn11.1.1.2serviceA.

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*). The GNS receives the location and the identifier from the service and creates a reference that points where the service is located. The management agents get the service references searching in this component.

5 Conclusion and future work

In this paper we present a model for service management in virtual active networks. In these networks, the services can migrate from a VAN to another one according to the customer's requirements. This is the common case in a telecom environment whereas the customers can install a VAN for satisfying their requirements. The environment has a Service Provider, and a Policy Manager responsible for controlling and processing policies. The customer has an Active Application for using the available environment.

The solution presented mainly is for mobile services, and due to this facility, the proposed model is based on a mobile agent platform and considers three management functional areas: accounting, performance and configuration. This model is flexible in the sense of management questions can be answered from different levels of management: per service, per host, per VAN, and in the network as a whole. We have also described where the MIBs are located throughout these different levels of management and the hierarchy for gathering data. We have created some scenarios for simulating a telecom environment and exploited the service provisioning and management. The scenarios were created using the ANTS toolkit which has sufficient features for implementing our experimentation environment. We have also developed a Global Naming Service in order to allow the service remotely registers itself.

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 the Java objects (services). In this way, we intend to include a new naming service for supporting these CORBA objects and to update the interface of management agents to support MASIF for management of these new services. We plan to allow the customer migrates a service to another VAN and to enable a customer from a VAN accesses a service from another one. Finally, we intend to develop a Security Manager for customer authentication and for avoiding untrusted code in the VANs in order to consider the security functional area. Simulations have shown that the proposed model can handle telecom environments with mobile services and Virtual Active Networks successfully, opening up a potential further study.

Acknowledgements

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

References

[D.97] Wetherall, D. Developing Networks Protocols with the ANTS Toolkit. *Design Review*, August 1997.

[DY00] Raz, D. and Shavitt, Y. Active Networks for Efficient Distributed Network Management. *IEEE Communications Magazine*, pp. 138-143, March 2000.

[Gro99] AN Architecture Working Group. Architectural framework for active networks. *Calvert, K. (editor)*, July 1999.

[JVL98] Wetherall, D., J., Guttag, J., V., and Tennenhouse, D., L. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *IEEE OPENARCH'98*, San Francisco, CA, April 1998.

[K.99] Psounis, K. Active Networks: Applications, Security, Safety, and Architectures. *IEEE Communications Surveys*, pp. 2-16, First Quarter 1999.

[MR99] Bruner, M. and Stadler, R. 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.

[MR00] Bruner, M. and Stadler, R. Service Management in Multiparty Active Networks. *IEEE Communications Magazine*, pp. 144-151, March 2000.

[MS99] Breugst, M. and Choy, S. Management of Mobile Agent Based Services. *Intelligence in Services and Networks (IS&N'99)*, Barcelona, Spain, Springer, pp. 143-154, April 1999.

[MT98] Breugst, M. and Magedanz, T. Mobile Agents - Enabling Technology for Active Intelligent Network Implementation. *IEEE Network*, pp. 53-60, May/June 1998.

[RR00] Kawamura, R. and Stadler, R. Active Distributed Management for IP Networks. *IEEE Communications Magazine*, pp. 114-120, April 2000.

[WHJC01] Marshall, I., W., Gharib, H., Hardwicke, J., and Roadknight, C. A Novel Architecture for Active Service Management. *Seventh IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, Washington, USA, May 2001.

[Y.93] Yemini, Y. The OSI Network Management Model. *IEEE Communications Magazine*, pp. 20-29, May 1993.