RESOURCE MANAGEMENT IN ASMA PLATFORM

Habib Bakour, Nadia Boukhatem
Computer Science and Network department
Ecole Nationale Supérieure des Télécommunications
46, Rue Barrault – 75013 Paris – France
{bakour, boukhatem}@infres.enst.fr

Abstract: The combination of virtual networks and active networking technology results in a new concept called virtual active network (VAN). Different interpretations are associated with the notion of VAN. We consider a VAN as a service abstraction offered by a provider to its customers. For the provider, the VAN represents the means to partition the network resources and isolate the customers from one another in virtual environments. For the customer, the VAN represents the environment in which it can install, configure and run its own services without further interaction with the provider. The partitioning of resources between customers is essential in such an architecture so that a customer does not monopolize resources, thus penalizing other customers. It can be implemented by performing a strict resource management.
In this paper, we present our ASMA (Active Service Management Architecture) platform which on the one hand, enables the customer to request a VAN from the provider, and manage his own services within his VAN and, on the other hand, allows the provider to manage the VANs created in his domain. We particularly focus on the resource control model defined in ASMA platform. Experiments are also presented to illustrate the behavior of the VAN nodes during service execution and show the efficiency of the proposed resource control schemes, allowing running services not to exceed negotiated resource consumption.

Key words: Virtual Active Networks, Resource Management, Active Service Management

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35703-4_21
D. Gaïti et al. (eds.), *Network Control and Engineering for QoS, Security and Mobility II*

1. INTRODUCTION

The development and deployment of new network services, especially services that operate on the IP layer, through best practice and standardization are too slow and cannot match the rapid changes in which requirements of various applications are growing. Emergence of active network concepts provide the users with the flexibility of deployment of customized services on the operator's infrastructure. This will be achieved through deployment of user customized control code in virtual executions environments (EE). In order to achieve efficient network operation, the resource consumption within these EEs must be managed.

We developed the ASMA [1] platform to provide customers a network environment where they can install and manage their own services. In this platform, the control of resources is performed through virtual active networks in order to avoid any resource monopolization or over consumption. The VAN concept has been developed in some recent works [2][3][4]. However, different interpretations are associated with the notion of VAN. In [4] the authors introduced the notion of "spawning networks" as a new class of open programmable networks. They believe that the VAN concept can be a foundation for the automation of the design, deployment and management of new network architectures. In [3], the authors introduce the notion of virtual active network which is defined as a dynamically constructed virtual network that provides application-specific services. In this work, the authors focus on the definition of abstractions through which applications can specify a virtual active network. In [2], the authors consider a virtual active network as a generic service abstraction offered by a provider to its customers providing a high level of autonomy in service management. From the customer's perspective, the VAN represents the environment in which the customer can install, configure and run network services without further interaction with the provider. From the provider's perspective, the VAN represents the means for partitioning the network resources and isolating customers from one another in virtual environments.

We consider a virtual active network following the definition given by [2]. In this paper, we present the ASMA platform which on the one hand, allows the implementation of the VAN concept and its management and, on the other hand, provides functionality allowing a dynamic and flexible service management, within a VAN. In particular, we focus on the resource management model which includes three phases:

First, the resource admission control which is performed by maintaining measurement state information globally. Next, the resource allocation is insured by the enforcement of specific policies which are installed according to the negotiated VAN level specification between the customer and the

provider. Finally, the resource consumption control is performed during service execution and is based on the enforced policies.

This paper is organized as follows. In section 2, we present the ASMA architecture and its interfaces. In section 3, we introduce the resource management model used in our ASMA platform. Section 4 provides the experiment results showing the behavior of active nodes. In particular, a cache service is developed and tested. We conclude with a summarization of this contribution and an outlook on further work.

2. ACTIVE SERVICE MANAGEMENT ARCHITECTURE

In this section, we present the architectural components which constitutes the ASMA platform.

As mentioned above, we consider a VAN as a means to allow the network provider, whose infrastructure is based on active networking technology, to support a large number of customers, all of whom independently install and run their own active services in the provider's domain [2][5]. A customer can be an end-user, an ISP or another network provider. The provider is responsible for creating, managing and monitoring VANs, in response to customer requests.

At the lowest level, a VAN can be described as a graph of virtual active nodes interconnected by virtual links. Virtual active nodes provide active packet processing functionality inside the network. They constitute execution environments (EE) having their own resources (Memory, CPU, ...). A virtual link is built on top of physical links connecting two virtual nodes (a virtual link can cross several physical links). Each virtual link has an amount of bandwidth allocated to it.

When a customer requests the creation of a VAN with specified needs (topology, memory, CPU, bandwidth, ...), an EE is created on the active node. The VAN's needs are translated within an EE, in terms of allocation of virtual memory and virtual processing capacities, as well as allocation of virtual ports. The creation of several VANs involves the creation of several independent EEs running on the same active node. The consumption of the resources is carefully controlled within an EE.

At the higher level, the ASMA platform allows the customer and the network to communicate through specific interfaces, as shown in figure 1.

The first interface (*VAN Request Interface*) is a dynamic negotiation protocol for VAN level specification. When a customer wants to create a VAN, he indicates to the provider the requirements he needs for his VAN. The provider may not be able to accommodate the customer's needs. Thus,

the customer and the provider enter in a negotiation process to come to an agreement on the resources required and the provided VAN.

The second interface (*VAN Management Interface*) offers to the provider an environment for managing the VANs of his domain. Through this interface, the provider can create, modify, remove and monitor the VANs running within his management domain.

The third interface (*Service Management Interface*) is related to service management. As stated previously, a VAN constitutes an environment in which the customer can install, manage and run active services in an independent manner. The service management interface provides the customer with a means to manage and control his own services within his own VAN.

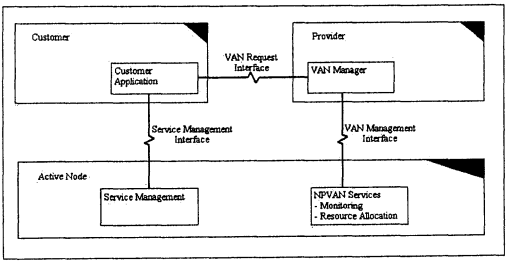

Figure 1. ASMA Interfaces

2.1 Dynamic VAN level negotiation protocol

The VAN Request interface is modeled as a dynamic negotiation protocol. This protocol allows the customers to request the creation of a VAN according to a defined VAN level specification. It allows also the customers to renegotiate this service level on-demand.

A VAN can be seen as a specific service which has particular requirements. To specify a VAN, we define the following parameters (VAN-Spec): Resource requirements and VAN Topology. Resource requirements specify the amount of resources needed by the customer (Memory, Bandwidth, Disk and CPU). VAN topology specifies the topology features of the VAN. This concerns the number of links, the number of nodes, and the way they are interconnected (chain, ring, star, etc.). When the provider receives a VAN creation request, it has to map the virtual VAN topology onto a physical topology while satisfying the resource constraints. To deal with this problem, we proposed a heuristic algorithm [1].

To request a VAN installation, the customer sends a REQUEST message to the provider with the desired VAN level specification. The provider

replies with a RESPONSE message indicating whether it accepts or rejects the VAN request. In the case of rejection, the provider proposes another VAN specification. In both cases, the customer sends a REPORT message either to confirm the acceptance or rejection of the proposed VAN specification. The exchanged messages between the customer and the provider are depicted in figure 2.

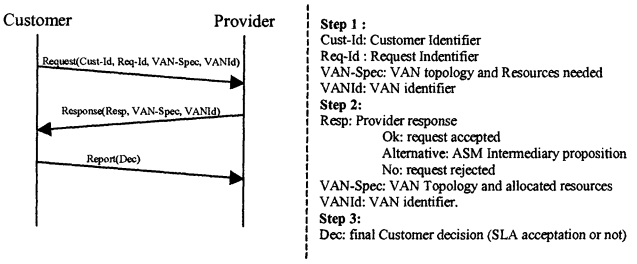

Figure 2. Dynamic VAN negotiation protocol

During the VAN lifetime, the customer has the possibility of modifying his VAN-Spec. He can add/remove nodes or reconsider the reserved resources. For this purpose, he has to renegotiate his VAN-Spec. The renegotiation proceeds in the same manner as the negotiation, except that the customer has to specify the VAN identifier (VANId) in the REQUEST message. The exchanged requests and responses are sent through XML [6] documents.

In this paper, we focus on resource management of the ASMA platform. Details related to the dynamic negotiation protocol and the VAN provisioning algorithm are presented in [1].

2.2 Service Management

After a VAN installation, the customer can install and execute his own services within this VAN.

The processing of packets inside traditional networks is limited to operations on packet headers which are mainly used for routing purposes. Active networks allow more efficient processing by allowing the active nodes to perform customized computations.

Active processing gives the customers the ability to dynamically deploy their services. As mentioned above, in an active node, an execution environment (EE) is associated with a VAN. The creation of several VANs within an active node implies the creation of several independent EEs running in this active node. In order to differentiate between different VANs, each active packet contains an identifier to specify the VAN and so, the EE

that will process it. The active node integrates a demultiplexer which forwards the packets to the corresponding EEs for processing.

In an active node, the customer can install new services as active components. These components are downloaded/injected using a storage mechanism in the node and executed in the corresponding execution environment. The customer can modify or remove his services as he pleases. He has the possibility of installing, activating, deactivating, removing and modifying the service.

2.3 VAN Management

In ASMA platform, the domain-wide VAN management is a task performed by a server which resides in a node within the provider domain. This server is called ASM (Active Service Manager) (figure 3).

When the ASM server receives a VAN creation request, it determines if sufficient resources are available to meet that new demand. This decision is based on measurement state information which is maintained by the server. When there are sufficient resources to accommodate the customer needs, the server allocates the resources and enforces policies within the network nodes.

The network information is stored in two databases: resource database (RDB) and VAN database (VANDB). The RDB stores information concerning the available resources within a domain. This database is updated by the monitoring agents installed in the NPVAN (Network Provider VAN) nodes. The NPVAN contains all the node of the domain and helps the provider to control these nodes. The functioning of the monitoring agents is detailed in section 3. The VANDB contains the information related to the VANs topology and VANs allocated resources. This database is updated at each VAN creation, modification or removal.

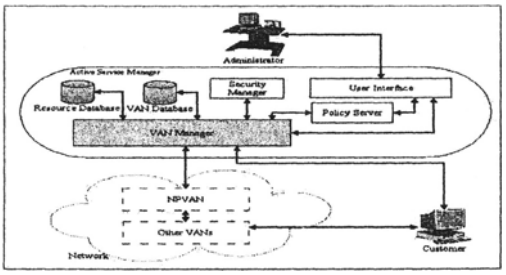

Figure 3. ASM Server Architecture

These two Databases are managed by the VAN Manager which is the main component of the ASM. In addition to the maintenance of these databases, the VAN Manager performs several tasks:

- It is in charge of the reception of customer requests (VAN creation demand, for example).
- It is in charge of accepting new VAN installation by controlling the resource availability and checking the access authorization (via the security manager)
- It applies the decisions of the policy server in the network nodes, through NPVAN services.

3. RESOURCE MANAGEMENT

The ASMA resource management model consists in three functions: Admission control, resource allocation and resource consumption and access control.

The resource database, presented above, constitutes the basis for the resource admission control. Indeed, when a new VAN creation demand is received, the ASM server uses this database and the VAN-spec to determine if the new demand can be satisfied. This database is updated using network-monitoring agents, which are installed in all NPVAN nodes. Each agent collects information about the resource consumption of the node (CPU, Memory, Disk, Bandwidth), updates the local database, and sends messages periodically to the ASM server to update its resource database.

The resource allocation is insured by the policy installation. Once the ASM has decided to allocate resources to a VAN, it instantiates policies using the VAN-Spec values, and sends them to the appropriate nodes. In an active node, policies are stored in the policy base in order to be used to control the resource consumption (see figure 4).

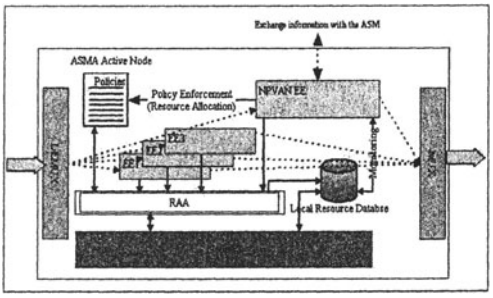

Figure 4. Active Node Architecture

The policy instantiation is performed by the Policy Server (figure 3). There are different types of policies tied to the different types of resources. The resources are classified in three categories: process (CPU), storage (disk and memory) and network (bandwidth). The CPU resource control support a

simple relative-share CPU reservation, and CPU rates are specified in terms of processor cycles per second.

Storage resources that include memory and disk are characterized by the available space. The allocation is performed by allowing each service a maximum number of bytes. Network resources are defined by the available network bandwidth.

The resource consumption and access control is ensured by an agent, called RAA (Resource Access Agent). The RAA authorizes resource access according to installed policies, and sends a warning message to the service owner in case the service exceeds its limits.

The RAA agent uses a virtual resource access mechanism. This results in the definition of resource objects (ROs), where each RO corresponds to a resource category (Process, Storage, Network). When an active service wants to access a physical resource, it invokes the RAA access functions. A function checks if the service is authorized to access the resource, by consulting the installed policy.

A policy is characterized by three parameters: the VAN identifier, the policy condition (which concerns the resource access and consumption constraints), and the policy action. These policies control the VAN resource.

Figure 5 illustrates an example of a policy formatted in XML. This example represents a policy that concerns disk resource consumption. This policy enables to stop access to a disk and sends a Failure message to the VAN customer. We have also defined other policies that concern each of the above-motioned resources (Memory, CPU, Bandwidth).

```
<policy>
        <owner>
                <VAN>VAN_Id</VAN>
        </owner>
        <condition>
                <operand>Disk_Usage</operand>
                <operator>Equal</operator>
                <operand>Negociated_Disk_Amount</operand>
        </condition>
        <action>
                <target>
                        <role>VAN_Id_Customer</role>
                </target>
                <data>
                        <methode>Send_Failure</methode>
                </data>
        </action>
</policy>
```

Figure 5. Example of a disk access policy

The policies are expressed in XML which has recently emerged as a widely accepted way of representing and exchanging structured information. Furthermore, XML can be extensible according to user needs, having a fixed syntax but an unlimited vocabulary, this extensibility (via the definition of new tags) enables the straightforward addition of new capabilities [7].

4. EXPERIMENTS AND RESULTS

4.1 ASMA platform

ASMA is deployed over PC/Linux using the Java NodeOS [8]. For the experiments, we used the ANTS (Active Network Transfer System) [9] execution environment. In this prototype, we distinguish two main applications: the Provider Application, which is responsible for managing the VANs and monitoring the network resources, and the Customer application which allows the customer to request a VAN creation from the provider, manage the VAN, install services and manage them.

For the experiment, we deployed the ASMA platform over four nodes (AN1, AN2, AN3, AN4). A graphical interface was developed (see figure 6). It enables the administrator to browse the ASMA information databases and installed VAN information.

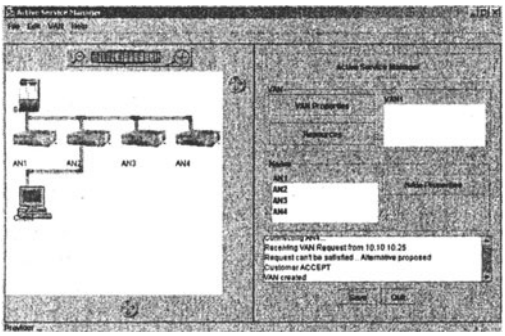

Figure 6. ASM Server user interface

```
<?xml version="1.0"?>
<REP_VAN_SPEC>
        <Topology>
                <Rep>Accept</Rep>
                <Nodes>3</Nodes>
        </Topology>
        <Resources>
                <Node>
                        <CPU>
                            <Rep>Accept</Rep>
                            <Val>40</Val>
                        </CPU>
                        <Memory>
                            <Rep>Accept</Rep>
                            <Val>None</Val>
                        </Memory>
                        <Disk>
                            <Rep>Accept</Rep>
                            <Val>20</Val>
                        </Disk>
                </Node>
                <Link>
                        <Bandwidth>
                            <Rep>Accept</Rep>
                            <Val>300</Val>
                        </Bandwidth>
                </Link>
        </Resources>
</REP_VAN_SPEC>
```

Figure 7. Provider VAN-Spec Answer

4.2 A cache service scenario

In this experiment, we aim to validate our resource management model by analyzing the resources consumption during a service execution within an active node. We are particularly interested by showing the control performed at network and disk levels.

In this experiment, a customer Cust1 requests a VAN with the following characteristics:

- VANId: VAN1
- VAN Topology: Chain

- Number of nodes: 3 (AN1, AN2, AN3)
- Maximum Bandwidth: 300 kbytes/s
- Maximum Disk Consumption: 20 Mbytes
- Maximum CPU usage: 40%

Then, he enters in a negotiation phase with the ASM server, obtains a positive response and the VAN1 is installed (figure 7 shows the provider answer in XML format).

For the experiment, we develop a cache service installed in each node of VAN1. This service consists in storing the downloaded files in the visited VAN nodes.

Two web servers are defined and connected respectively to the nodes AN2 and AN3 (see figure 8). When Cust1 installs his cache service, he sends it to the node AN1 which in turn deploys it over all the nodes of VAN1.

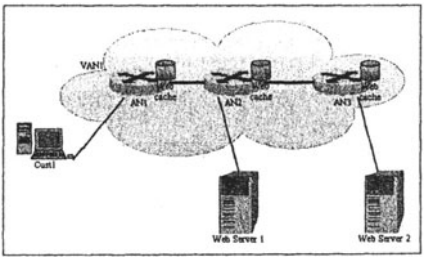

Figure 8. VAN1 topology

Cust1 downloads the files F1 from Web Server 1 and F2 from Web Server 2. The size of the two files are respectively: size F1 = 5,754 Mbytes and size F2 = 11,683 Mbytes. The first file is requested at t_{11}=18s and the second at t_{12}=42s.

Figure 9 shows the bandwidth consumption in the node AN2. In this figure, four phases are identified.

In the range time t_{11}=18s and t_{12}=42s, the bandwidth consumption due to the transfer of file F1 turns around 200 Kbytes/s and does not exceed the bandwidth limit of the VAN1 (300kbytes/s). The file F1 downloading does not consume the total bandwidth because of the limitation of the Web server1 in terms of time processing.

When Cust1 downloads the file F2 (t_{12}=42s), the total bandwidth consumption in AN2 exceeds 300 kbytes/s, however the resource access agent installed in the node AN2 refuses the traffic over 300 kbytes/s and sends a warning message to Cust1. Figure 9 shows that the total bandwidth does not exceed the limit of VAN1 during the range time t_{12}=42s and t_{13}=57s.

At t_{13}= 57s, the file transfer of file F1 is completed and the transfer of the file F2 continues (time range t_{13}=57s and t_{14}=111s) with a bandwidth consumption which does not exceed the limit of 300 kbytes/s.

The time range t_{14}=111 and t_{15}=143s corresponds to the downloading of a third file F3 from the Web server 2. The size of F3 is fixed to 6,792 Mbytes. The downloading of this file is considered to show the resource control performed at disk level.

Figure 10 shows the disk occupation in the node AN2. In this figure, three phases are identified. The first phase (range time t_{21}=57 and t_{22}=111s) corresponds to the storage of the file F1 in the cache. The second phase between the instants t_{22}=111s and t_{23}=143s corresponds to the storage of the two files F1 and F2. The total size does not excess the limit fixed to 20 Mbytes.

The instant t_{23}=143s corresponds to the beginning of the storage of file F3. However, the total size of the three files exceeds the limit of the disk. We defined a policy which performs a FIFO queuing discipline to remove the first saved file and free the disk space, in case the disk is full. In our scenario, the file F1 has been removed and replaced by F3. The disk occupation is less that the space disk limitation of VAN1.

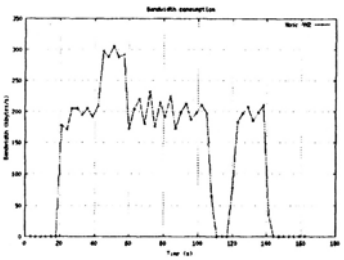

Figure 9. Bandwidth usage in AN2

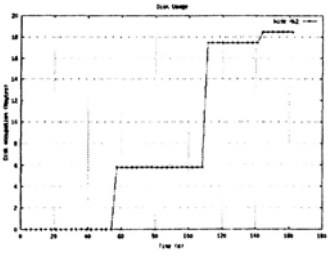

Figure 10. Disk occupation in AN2

The goal of the above experiment is to validate the implementation of the ASMA prototype and to show that a strict resource control is applied, even if the service presented is simple. Obviously, we plan to develop and test other services over more scalable test-beds considering the different functions of the ASMA platform.

5. CONCLUSION

In this paper, we presented our ASMA platform which is based on the VAN concept. ASMA allows the customer to request a VAN through a dynamic negotiation protocol. In addition, an environment for VAN

management is developed to enable the provider to supervise his network, and in particular, to control the installed VANs within his administrative domain. The resource allocation for VAN creation is achieved through the enforcement of policies within active nodes. Policies expressed in XML give a flexible way of describing resource requirements without implementation of specific details.

In addition, the ASMA platform offers the customers an environment to deploy and customize their active services. The control of the consumed resources during the service execution is performed through the RAA agent which takes its decisions according to the installed policies.

An ASMA prototype was developed using the ANTS EEs. A simple cache service was also developed to validate the prototype. Experiments illustrate the behavior of VAN nodes during service execution and show that a strict control is applied if the installed services exceed the negotiated limits.

Up to now, VAN resource allocation mechanism is deterministic, i.e. the resource admission control is based on the peak rate required. We are currently studying a dynamic resource management model. In addition, new monitoring tools are under test and will be presented in forthcoming papers.

REFERENCES

[1] Habib Bakour, Nadia Boukhatem: *"ASMA: Active Service Management Architecture"* Technical report – ENST Paris, March 2003

[2] M. Brunner, R. Stadler: *"Virtual Active Networks – Safe and Flexible Environment for Customer-managed Services"* – 10th International workshop on Distributed Systems, Operations and Management (DSOM'99) Zurich, October 1999

[3] G. Su, Y. Yemini: *"Virtual Active Networks: Towards Multi-Edged Network Computing"*- Computer Networks, pp. 153-168, July 2001

[4] M. E. Kounavis, A. T. Campbell, S. Chou, F. Modoux, J. Vicente, H. Zhuang: *"The Genesis Kernel: A Programming System for Spawning Network Architectures"* – IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Active and Programmable Networks, Vol. 19, No.3, pp. 49-73, March 2001.

[5] M. Brunner: *"A Service Management toolkit for Active Networks"* – IEEE/IFIP Network Operations and Management Symposium (NOMS 2000), Hawaii, USA, April 2000.

[6] World Wide Web Consortium, Extensible Markup Language www.w3c.org/XML

[7] P. Mckee, I. Marshall: *"Behavioural Specification Using XML"*, Proc IEEE FTDCS'99 – Capetown – pp 53-59

[8] P. Tullman et al: *"Janos: A Java-oriented OS for Active Networks"* – IEEE Journal on selected Areas of Communication. Volume 19, Number 3, March 2001.

[9] D. Wetherall, J. V. Guttag and D. L. Tennenhouse: *"ANTS: A Toolkit for Building and Dnamically Deploying Network Protocols"*, IEEE Openarch, April 1998 http://www.cs.utah.edu/flux/janos/ants.html