ASMA: An Active Architecture for Dynamic Service Deployment

Habib Bakour and 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 deregulated telecommunication market tends to force network operators to open their infrastructure to third party service providers to offer a large variety of services. This trend is already appearing, particularly in the mobile environment, and it is widely admitted that this situation will not take a long time to be applied to the information technology (IT) communications, thereby creating a strong competition between the different network actors. In this context, one of the major stakes for both operators and service providers is to be capable of quickly providing new and attractive services. For instance, on-demand service creation and deployment will enable the diversification of the service offerings and will provide more business opportunities. This article presents an approach for a dynamic service deployment based on the virtual active networks (VAN) concept. In particular, we present the ASMA (Active Service Management Architecture) platform which we developed to provide an environment allowing flexible service deployment and management.

1 Introduction

Until recently, the deployment of new services within the network operator infrastructure could only be achieved with the commitment of manufacturers, due to the closed nature of network equipments. Any modification in network equipment functionalities would usually be expensive and require a long time before being operational, due to standardization and development processes.

Active networks provide programmability to enable third parties (end-users, application and service providers) to run their own services within a network. By *programmability* we mean that active nodes are able to execute code injected by the third parties to create the desired functionality at run-time. This provides an open environment allowing new services to be deployed more or less on-demand.

While facilitating the introduction of new services within the network, active networks make the management even more complex, in particular resource management. Indeed, several services belonging to different customers can be executed within the same active node and thus can share the same processing and network resources. The network provider must implement an efficient resource management scheme in order to avoid a specific service to monopolize the resources, thus penalizing and affecting other services.

A. Aagesen et al. (Eds.): INTELLCOMM 2004, LNCS 3283, pp. 264–272, 2004.

To better manage network resources, we use the VAN (Virtual Active Network) concept. As traditional virtual networks, a VAN is a logical network intended to organize network resources and to reduce the complexity of resource management.

Some recent works [1][2][3] have used the concept of VAN, however each of them has its own motivation and interpretation. The Genesis project [1] introduced the notion of "spawning networks", where a child network can be dynamically created, and inherit parts of the resources and functionality of the parent network. The specification of a child network is produced manually by a network architect. The VAN project [2] is mainly focused on the definition of abstractions through which applications can specify a VAN. The VAN [3] project achieved within ETH-Zurich focuses on the implementation of a service management toolkit that facilitates designing and managing services. Our aim is to define and implement a whole platform based on the VAN concept allowing a dynamic service deployment.

In this paper, we present ASMA (Active Service Management Architecture) 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.

This paper is organized as follows. In section 2, we give an overview on ASMA platform and present its interfaces and their functionalities. Section 3 is devoted to service management. In section 4, we provide details about the ASMA prototype. Finally, we conclude with a summary of this contribution and give some remarks.

2 The ASMA Architecture

In the ASMA platform, a VAN is considered as a service abstraction offered by a provider to its customers. We assume that the provider infrastructure is based on active networking technology.

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. A customer can be an end-user, a service provider or another network provider. The network provider is the entity that owns the network resources. The customer buys the resources from the provider in form of a VAN to install and run its own services.

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, etc.), an EE is created on the active node. The VAN's needs are translated within an EE in term of allocation of virtual memory and virtual processing capacities. The creation of several VANs involves the creation of several independent EEs running on the same active node.

At the higher level, the ASMA platform allows the customer and the network provider to communicate through specific interfaces (see figure 1).

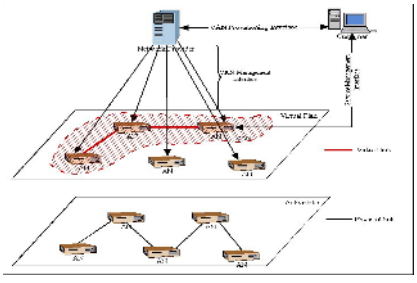


Fig. 1. ASMA Interfaces

The first interface (*VAN provisioning Interface*) gives the customer the ability to request a VAN from the network provider. The communication between the customer and the provider is achieved through a dynamic negotiation protocol which enables the customer to specify his needs (topology, resources…).

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 supervise the VANs running within his management domain.

The third interface (*Service Management Interface*) is related to service deployment and 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.

2.1 VAN Negotiation Protocol

The VAN creation requests are taken in charge by the network provider through the *"VAN Provisioning Interface"*. This interface is modeled as a dynamic negotiation protocol allowing 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 SM receives a VAN creation request, it maps the virtual VAN topology onto a physical topology through a proposed mapping algorithm [4].

To request a VAN installation, the customer sends a REQUEST message to the SM with the desired VAN level specification. The SM replies with a RESPONSE message indicating if it accepts or rejects the VAN request. In case of rejection, the SM 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 SM are depicted in figure 2.

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.

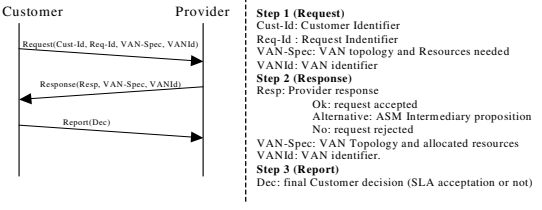


Fig. 2. Dynamic VAN negotiation protocol

2.1 VAN Management

The ASMA architecture consists of several functional components. The main component is the SM (System Manager, see figure 3) which resides in a node within the provider domain and is responsible for domain-wide VAN management.

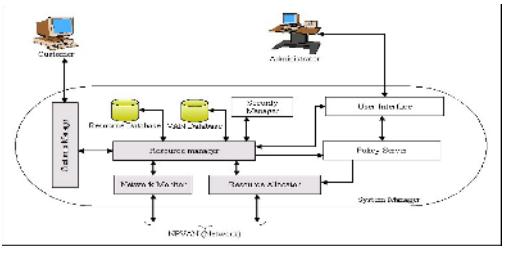


Fig. 3. SM Components

When the SM receives VAN creation requests through the Customer Manager (CM) it contacts the resource manager (RM) which determines if sufficient resources are available to meet that new demand. This decision is based on measurement state information which is maintained by the RM.

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 Network Monitor which receives the network information from the monitoring agents installed in the NPVAN nodes. 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.

In case of a successful negotiation, the resource manager sends the VAN specification (topology, resources…) to the policy server. The policy server builds policies – according to the VAN specification –, and sends them to the Resource Allocator (RA), which installs these policies in the corresponding nodes.

The security manager deals with the access control and enables to authenticate customers. Only authorized customers will access their corresponding VANs.

2.2.1 Resource Management

Active services require access to diverse system resources, such as forwarding network bandwidth, router CPU cycles, state-store capacity and memory capacity. The resource management is one important issue in our platform. The latter should provide capabilities to isolate the VANs owing to different customers. To each VAN a specific amount of resources, as specified in the VAN-Spec, should be allocated. In addition, services running in the same VAN should also be controlled.

Our ASMA architecture defines a resource model [5] consisting of three functions: admission control, resource allocation, and resource consumption and access control. These functions are ensured mainly by the Resource Manager. The resource consumption and access control are related to service execution and will be presented in 3.

The resource database (RDB), presented above, constitutes the basis for the resource admission control. Indeed, when a new VAN creation demand is received, the VAN manager 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 (i.e. all the active nodes of the domain).

It exists several EEs running in each active node. Each EE corresponds to a VAN. In particular, the NPVAN_EE corresponds to the NPVAN.

Each EE contains an agent which monitors the state of its resource consumption (CPU, Memory, Disk, bandwidth_out and bandwidth_in) through the NodeOS primitives.

The NPVAN_EE contains a monitoring agent which collects the monitoring information of the other agents and updates its local resource database (LRB). The LRB contains information related to the resource state in the active node.

The resource allocation is insured by the RA through the policy installation. Once the RM has decided to allocate resources to a VAN, it informs the policy server which instantiates policies using the VAN-Spec values. Then, the RA forwards these policies and installs them in the appropriate nodes through the NPVAN. In an active node, the policies are stored in the policy base in order to be used to control the resource consumption.

3 Service Management

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 customers ability to dynamically deploy their services. As mentioned above, in an active node, an execution environment (EE) is associated with a VAN (e.g. the NPVAN_EE is associated with the NPVAN). 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. The architecture of an active node is presented in figure 4.

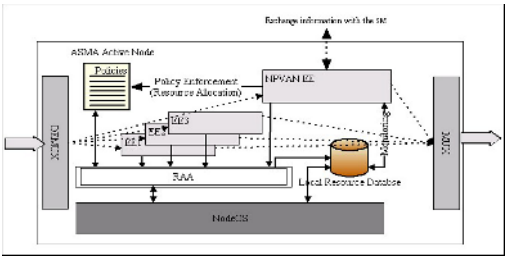


Fig. 4. Active Node Architecture

In ASMA, each service is characterized by: its *owner*, its *execution place*, and its *control* and *behavior* applications. The *owner* corresponds to the customer who owns the service. The *execution place* represents the list of nodes where the service will be executed. This place can contain a part of the VAN or all the nodes of the VAN.

The *control application* is the program that enables a customer to control his services. Through this application, the customer can install, activate/deactivate, update, delete his services. The *behavior application* is the program that defines the service behavior in the active node.

Once the service is activated, arriving packets to the active node are processed by the corresponding *behavior application* at the corresponding EE (each packet contains the identifier of the service which will process it). If the EE does not find a corresponding *behavior application* (when the service is deactivated, for example), then the active packet will be forwarded to the next node (following the routing table).

The execution of an active service implies the consumption of active node resources. The resource consumption and access control is ensured by an agent, called RAA (Resource Access Agent), which is located between the NodeOS and the

different EEs (figure 4). The RAA authorizes resource access according to the installed policies, and sends a warning message to the SM server and to the service owner in case the service exceeds its limits.

To guarantee a safe execution of the services, the RAA uses a virtual resource control mechanism. This consists 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 corresponding RAA access function.

4 Implementation and Experiments

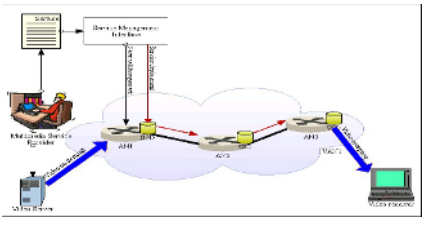
In this section, we present an experiment illustrating the feasibility of a VAN-based platform allowing a flexible active services deployment. This experiment is based on a scenario that presents an example of a service deployment. In particular, we developed an active multimedia streaming control service. This active service is used in the case of a video on demand allowing video clients to request video movies from a multimedia server.

The ASMA platform is deployed over PC/Linux using the Java NodeOS [6] and ANTS (Active Network Transfer System) [7] as execution environment. In this prototype, we distinguish two main applications: the System Manager Application, which is responsible for managing VANs and supervising the network behavior, and the Customer application which allows the customer to request a VAN from the System Manager, install services and manage them.

Different packages are developed, and integrated to the ANTS environment. These packages concern particularly the RAA agent and the monitoring services.

A graphical interface was developed. It enables the administrator to browse the SM information databases of system resources and installed VAN information.

In our scenario, an active video control service is deployed on a VAN which has the characteristics depicted in figure 5.



Characteristics of VAN1 :

- VANId: VAN1
- VAN Topology: Chain
- Number of nodes: 3 (AN1, AN2, AN3)
- Maximum Bandwidth: 40 kbytes/s
- Maximum Disk Consumption: 20 Mbytes
- Maximum CPU usage: 30 % of the CPU time

Fig. 5. Example of a video service deployment

The developed service consists in an MPEG-4 [8] video streaming control mechanism. We remind that in the MPEG compression algorithm, the MPEG encoder generates three types of compressed frames: Intra-coded (I), Predictive (P), and Bi-directional (B) frames. The different parts of the video data stream have not the same

importance for the quality of the decoded video. The damages caused by some data loss in a reference picture (I-Frame or P-Frame) will affect subsequent picture(s) due to inter-frame predictions. Subsequently, I-Frame must be protected more than P-Frame and P-Frame more than B-Frame [9].

The video streaming control mechanism developed in our experiments is based on the Video Group of Picture Level Discard algorithm. This algorithm enables maintaining the state of a whole Group of Pictures (GoP). Thus, when the I-Frame is dropped, all the corresponding P-Frame and B-Frame are also dropped. To identify a Frame, the active program should be aware about the video structure.

For our experiments, the MPEG-4 presentation was obtained by using a set of multimedia components. We used the video components of the MPEG-4 coded Jurassic Park movie as an example of a medium video activity traffic. This trace file of this movie is freely available at [10].

Figure 6 shows the traffic entering at the active node AN1. This figure shows that the bandwidth never exceeds the level of 40 kbytes/s. This is due to the strict control performed by the RAA agent over the VAN1 (Max bandwidth = 40 kbytes/s), in particular at the entrance of the AN1 node.

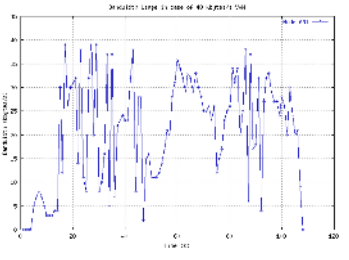


Fig. 6. Bandwidth usage at AN1

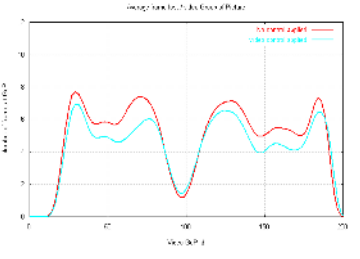


Fig. 7. Frame loss per GoP

When our service is activated, the active node drops randomly the video packets but when I-Frame is dropped, all the corresponding P-Frame, and B-Frame are also dropped.

The experiment is performed twice: before service control installation, and after its installation. Figure 7 shows the average of frame loss (due to frame drop or unusable frames) per GoP.

This figure shows that if the service control is activated, the video destination will receive more usable frames. This means that the installation of a video control service enables enhancement of video quality. In fact, dropping unusable frames (when the I-Frame is missing) guarantees delivery of only usable frames, and therefore frees network resources to guarantee transfer of a maximum of usable GoP.

We notice that, the nodes AN2 and AN3 do not make significant treatment on the service traffic, since the latter is regulated at the network access, e.g. at AN1 level. This is due to the nature of the implemented service itself.

5 Conclusion

Rapid deployment of new services based on an efficient resource control has motivated us to design and implement ASMA platform.

ASMA allows the customer to request VAN through a dynamic negotiation protocol, to manage his VANs and to deploy freely his active services. Moreover, the ASMA platform allows the provider to manage and supervise all VANs created in his domain.

A resource management model is proposed. The resource admission control function is used to determine if a new VAN can be created on not, according to the resource availability. Once a VAN is installed, the resource allocation is materialized by the installation of policies according to the negotiated VAN-spec. The execution of a service is subjected to a strict resource consumption and access control performed by the RAA.

An ASMA prototype is developed using the Janos operating system. Some experiments have been presented to show that a strict control is applied if the installed services exceed the negotiated limits.

We aimed through this work to show that the network virtualization allows the active networks technology to find in dynamic service deployment an extremely promising applicability.

References

1. M. Kounavis et al.: “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.
2. G. Su, Y. Yemini: “Virtual Active Networks: Towards Multi-Edged Network Computing”- Computer Networks, pp. 153-168, July 2001
3. M. Brunner, R. Stadler : “Virtual Active Networks – Safe and Flexible Environment for Customer-managed Services” – DSOM’99, Zurich, October 1999
4. Habib Bakour, Nadia Boukhatem: “ASMA: Active Service Management Architecture” Technical report – ENST Paris, March 2003
5. Habib Bakour, Nadia Boukhatem: “Dynamic service management in active networks” – IEEE SoftCOM 2003, Italy/Croatia, October 2003
6. 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
7. 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
8. ISO/IEC 14496-1 “Coding of audio-visual objects: Systems -final committee draft.”, may 1998
9. T. Ahmed et al.: “Encapsulation and Marking of MPEG-4 Video over IP Differentiated Services”, IEEE ISCC'2001, Tunisia, July 2001
10. http://www-tkn.ee.tu-berlin.de/~fitzek/TRACE/ltvt.html