Active Technology as an Efficient Approach to Control DiffServ Networks: The DACA Architecture

Nadjib Achir, Nazim Agoulmine, Mauro Fonseca, Yacine Ghamri-Doudane, and Guy Pujolle

LIP6 - Pierre et Marie CURIE University
8, rue du capitaine Scott, 75015 Paris, France.

{Najib.Achir, Nazim.Agoulmine, Mauro.Fonseca, Yacine.Ghamri, Guy.Pujolle}@lip6.fr

Abstract. The objective of this work is to propose an architectural solution to achieve an efficient, distributed control and management of DiffServ enabled networks. DiffServ offers a scalable QoS provisioning solution but it introduces a certain amount of complexity in terms of management and control. The proposed solution is an alternative to the client/server policy-based approach put forward by the IETF[1], which assumes that complete predefined instrumentation is already implemented in DiffServ nodes. In this work, we suppose that DiffServ nodes offer an API with a minimum number of control functions. Based on this assumption, the proposed solution introduces a new control architecture to configure and control these nodes in a customized and flexible manner.

1. Introduction

One of the biggest challenges in present-day telecommunication networks is the deployment of high-quality, large-scale multimedia applications. Thus QoS management and control issues have become key points of ongoing research. While optical technologies such as WDM and DWDM allow broadband communications, end-to-end assurance of the bandwidth and the delay are still an open issue when using the IP protocol because of the bottlenecks occurring at access and intermediate nodes. Some standards and proprietary solutions have already been specified in an attempt to guarantee a certain level of QoS to end-user communications crossing a number of network devices. Among these solutions, we find RSVP/IntServ and DiffServ. While the former, based on per flow reservation [1], presents some scalability problems, the latter is a promising scalable solution [2,3]. It is based on aggregate classes of traffic where each class possesses a certain level of QoS. This approach processes packets according to the aggregate to which it belongs, not on a stream-per-stream base, which explains why it is scalable. The principal side effect of this approach is the complex management. In fact, in order to offer the target QoS, the DiffServ routers must be configured with great precision and a monitoring process is

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: 10.1007/978-3-540-45812-8_28

[1] IETF : Internet Engineering Task Force

K.C. Almeroth and M. Hasan (Eds.): MMNS 2002, LNCS 2496, pp. 170-183, 2002.

required to control the QoS provided. Actually, DiffServ does not provide a strict QoS but a relative one, so that if the desired QoS is not fulfilled, changes must be made in the configurations or in the conditioning algorithms. Looking at currently used DiffServ routers, the only way to configure them is to use a complex configuration command script. However, reconfiguration is based only on the existing capacity of the router, with the result that there is no way to add new functionalities dynamically without rebooting the system.

The aim of this work is to investigate the possibility of developing a new solution based on active networks (ANs) in an aim to make the management and control of todayís IP networks more dynamic. The proposed solution is aimed at improving the DiffServ network architecture by implementing DiffServ functionalities as active components and control their service logic through policies. This solution will allow the current IP-based network architecture solution to be maintained while providing a more predictable and manageable network. Thus, ISPs will gain more control over the way network resources are used and consequently will be able to support their customersí requirements in what are called Service Level Agreements (i.e. contractual relationships between the customer and the provider) with far more efficiency.

This paper is organized as follows. Section 2 describes the DiffServ, policy-based management and active network technologies. The following section presents the objectives of our work while section 4 outlines the proposed architecture, its components and internal mechanisms. A number of elements concerning the implementation of this new architecture are given in section 5. Finally, a conclusion and suggestions for future work are presented.

2. Underlying Technologies

Three main technologies have been used to define the proposed architecture: DiffServ, Active Networking and Policy-based Management. These technologies are briefly introduced in this section.

2.1 QoS Management Using DiffServ

The IETF has specified the DiffServ architecture as a scalable QoS provisioning solution for the Internet. The proposed architecture introduces two main functionalities, namely Packet Marking using the IPv4 ToS byte and the Per Hop Behavior (PHB) [2] treatment. In the first case, the ToS byte field of the IPv4 packet is replaced by a new field called the Differentiated Services Code Point (DSCP) which identifies the treatment to apply to this particular packet [3]. The packet marking process is performed by edge routers at the ingress of the network to order to achieve a target end-to-end QoS. At the core of the network, packets with the same DSCP value will be treated in the same way by core routers, this treatment is the PHB, which is related to a Traffic Conditioning Block (TCB). The latter corresponds to the packet scheduling, queuing, policing or shaping behavior of a node on any given packet belonging to a Behavior Aggregate (BA).

2.2 Policy-Based Management

The policy-based management (PBM) approach aims to automate the management of complex networks by defining high-level management objectives that are enforced in the network equipment as a set of policies [4]. Policies are a set of pre-defined rules that govern network resource allocation. The IETF has specified a policy management architecture [5] composed of a PEP[2] component, a PDP[3] component and a Policy Repository component. Based on the resource allocation policies, the network administrator defines a set of rules that describe these policies and saves them in a Policy Repository. The PDP monitors this repository and applies policies as necessary. The PEP component is a policy decision enforcer located in or closely to the network equipment. Finally, the IETF has defined a Policy Core Information Model (PCIM) that allows policy objects to be represented in the DMTF[4] consortium Common Information Model (CIM) [6].

2.3 Active Networking

The active networking technology emerged from discussions within the Defense Advanced Research Projects Agency (DARPA) research community [7]. Enabling the programmability of networks was expected to allow a flexible network architecture thereby facilitating the integration of new services and accelerating network infrastructure innovation. Two approaches exist: the first, the so-called ìprogrammable networksî approach, has as its objective to open up network control to applications by means of standardized interfaces, the second, the ìactive packetsî or ìcapsulesî approach, allows application packets to carry their own processing.

3. Objective of the Work

The objective of this work is to investigate the possibility of using active networks coupled with policy-based management to propose a new solution for deploying and controlling DiffServ networks. The main characteristics we introduce in order to reduce the complexity of DiffServ networks management and control [8] are (1) more flexibility in provisioning DiffServ services, (2) better control of the DiffServ process at different points of the various TCBs (scheduler, meter, shaper, etc.) and (3) optimization of the network by allowing the deployment of customized control functions capable of reporting information that will allow problems in the network to be anticipated. The solution proposed here aims to introduces configuration and control based on dynamic introduction, modification and deletion of mobile code directly in the DiffServ routers to implement configuration and control functions. An architecture is therefore proposed for organizing the overall activity of the operator and provide him with a set of tools that will facilitate management activities: DACA, or DiffServ Active Control Architecture.

---

[2] PEP: Policy Enforcement Point

[3] PDP: Policy Decision Point

[4] DMTF: Distributed Management Task Force

4. The DACA Architecture

The proposed DACA architecture aims to facilitate and ideally automate the configuration of network equipment so that the provider does not have to do it manually for each node. This will avoid the administrator having to configure each router in the network individually to reflect a new or changing strategy. In present-day equipment, in a context of heterogeneous and high scale networks, these configuration tasks are very complex for the administrator. Reconfiguration of network equipments is a very fastidious task calling for highly skilled expertise.

The DACA architecture presented in Figure 1 provides a set of tools and functionalities allowing dynamic deployment and distribution of control services in an active network. It uses active codes and policies as basic paradigms to introduce specific functionalities rapidly and transparently in active nodes. The main control service in this work is DiffServ.

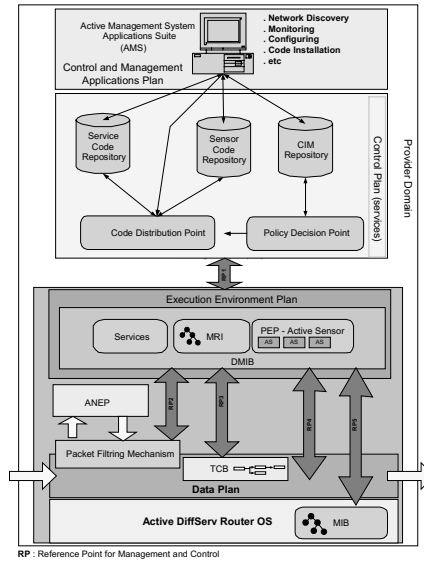

Fig. 1. The DACA architecture

4.1 Lifecycle of Differentiated Services

DiffServ deployment consists in specifying TCBs for ingress and core routers and implementing these TCBs in the network. We have identified four phases allowing configuration and assurance of the service:

- **Pre-service phase**: When an ISP has signed a SLA with a customer to provide a service with a certain QoS, the ISP administrator creates the corresponding policies in the policy repository.
- **Before-service phase**: The administrator must specify the exact TCB to be used for the customer or define a new one, depending on the service requested,

existing services and the state of the network. For example, if the network is overprovisioned, the administrator can decide to install only simplified versions of TCB.

- **In-service phase**: In this phase, the administrator can define high-level rules that will allow network utilization to be optimized. For example, if congestion occurs in the network for entire classes or a particular class, then a new complex TCB should be installed.
- **Post-service phase**: At this stage, the administrator is interested in customer care and, in particular, in SLA violation detection.

4.2 Component Description

The DACA architecture consists of a set of components distributed among five logical plans: the management and control applications plan, the provider control plan, the ADR OS plan, the active-router execution environment plan and, lastly, the active-router data plan. In the **management and control applications plan**, we find the applications that can be used by the manager to control the active DiffServ network. **A** non-exhaustive list of applications is: the **Configuration application** that helps the administrator to configure the local system as well as any remote active nodes; the **Network Discovering application** is used to discover the active network topology dynamically; the **Monitoring application** serves to monitor any target active node specifically or the entire network behavior; and, lastly, the **Code installation application** allows the administrator to define new control algorithms or management functions to be used in the network and to enforce them in any active node.

In the **provider active plan**, we find the following components: the **Active Management System (AMS),** i.e. the central point of management in the provider domain. All **management policies** are defined and stored in this system. It comprises a set of tools (for editing, distribution and management) that enables network manager to define the basic management and control functionalities to be deployed in the network. These functionalities concern the network level as well as the service level management. Each functionality can be developed on the basis of a service composition principle, which means that the manager can specify enhanced functionalities that are built on top of existing management components (information, services and sensor) already deployed in the managed active node. The **CIM Repository (CIMR)** is the virtual store that contains information about the network, the services, the customers, the SLA and so on. It also contains a description of the policies and of the association between all these objects. The **Service Code Repository (SCR)** is the virtual store containing active codes relating to DiffServ TCBs. Each code corresponds to a functionality that can be installed dynamically in the active node in order to implement a particular control function (classifier, meter, shaper, etc). The **Sensor Code Repository (SeCR)** is the virtual store that contains active codes relating to sensors. Sensors are management functionalities that can be installed dynamically in any active node in order to implement a particular management function (monitoring, event notification, summarization, etc). The **Policy Decision Point (PDP)** is the decision-making component described in section

2.2 which is designed to monitor events in general (time, network, etc.) and trigger policy rules of which the condition part is verified with these events. **The Code Distribution Point (CDP)** is a component responsible for deploying service codes or sensor codes on behalf of the administrator or when a PDP requests it in order to perform a policy.

The **ADR OS plan** comprises the following components: the **Active DiffServ Router OS (ADR OS), i.e.** the active node. ADR OS runs an operating system which supports the active plan as well as the execution environment. Normally at this level we have the conventional **Management Information Base (MIB),** defined by the IETF SMI (System Management Information), which contains a set of objects representing different resources of the active node viewed as an IP router.

The following Execution **Environment Plan** comprises the following components: the **Dynamic MIB (DMIB),** a special active application which supports the local installation and management of active management services. In this case, therefore, it will support the installation and management of services, active sensors, and ADRs. Unlike the MIB, the DMIB offers a view of the ADR seen as an active node, i.e. a management information view as well as an active-services view. It can be extended on the fly with new objects and associated instrumentation. The **Managed Resources Information (MRI),** like the MIB, is a repository of information about each manageable resource in the ADR. Passive, it requires instrumentation to update the information. Active Sensors (AS) that push information into the MRI or any other function defined by the manager can apply the instrumentation. The **Services** are thus active control services available in the ADR that can be used by other components such active sensors. The services are allowed to use managed resources information, DiffServ services or active sensors. Not all services are active all the time. They need to be activated in order to be operational. An example is the FlowID update process, which sets up a FlowID table used at the ADR level by the filter. The **PEP active sensor** performs the PEP functions in the active router as presented in section 2.2. Its role is to enforce policy actions requested by the PDP and also local PDP functions, or LPDP. Thus it monitors managed resource information and applies management policies when events occur locally. Policies are defined in terms of rules with a condition part and an action part. The LPDP implements active sensors, individual processes that handle each rule. Each Active Sensor is responsible for performing one rule and monitoring the action part to verify whether it is valid. It uses other components of the architecture to monitor ADR resources (MIB, DMIB) and other active sensors to create a complex monitoring process. An active sensor can be active or passive, depending on whether its condition part is verified (all corresponding actions being applied) or passive in the sense that it is still monitoring the validity of the condition part.

Finally, in the **execution data plan**, we find the Packet **Filtering Mechanism (PFM)** and the **Traffic Conditioning Block (TCB),** which is defined in the DiffServ architecture. In order to optimize the performance of the system and not intercept all packets in the network, we have defined a packet filtering mechanism capable of intercepting and redirecting packets based on specific filters, which define the set of active applications installed in the active node. From the data plan viewpoint, the installation, modification and removal of a TCB is transparent (depending, of course, on the performance of the system and how fast it is able to react to particular events).

4.3 Filtering Capsules in the Data Plan

When a capsule crosses the network, it goes through several active routers. If the capsule belongs to an active application whose protocol is deployed in a particular node, it should be handled by the active protocol before being forwarded to the following node. Not all active-application protocols will be installed in all active routers, so that the role of the PFM is to detect capsules to be captured and forward them to the EE[5] as fast as possible in order to avoid cumulative delay. The PFM uses a special FlowID table which identifies the installed set of active applications in order to select the capsules to be captured. The Flow ID is stored in an option field of the IPv4 packet or in the FlowID field of the IPv6 packet. To keep the FlowID table up to date, the so-called FlowID update service is activated. Its role is to monitor the installed active application protocols and update the FlowID by adding or removing an entry in the table (Fig.2).

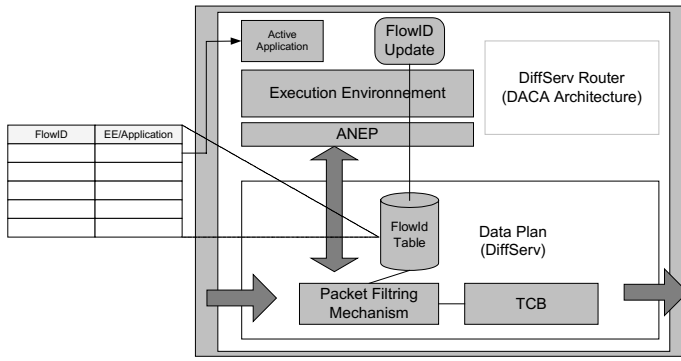

Fig. 2. FlowID table updates and packet filtering mechanism

PFM plays a central role in the architecture and, as such, has an important impact on the network performance. It therefore has to be installed as close as possible to the active router OS.

4.4 Monitoring ADR Resources

As stated at the beginning of the paper, it is important for the operator to have detailed information about the network behavior. The DMIB offers facilities for monitoring and notifying events dynamically, i.e. the information model is not static as in the classical IP management protocol SNMP (Simple Network Management Protocol) and its corresponding SMI. The DMIB offers the means to monitor any resource in the ADR efficiently using two APIs:

- **Dynamic Information Interface** (DInfo) is an object-oriented API that allows access to management information, service objects or active sensors. This API comprises five primitives: getAbout(), getVersion(), get(), set() and function().

[5] EE: Execution Environment

Of these, get() and set() make it possible to read and write the attributes of a particular managed object while function() allows a particular method to be applied to one or several particular managed objects.

- **Dynamic Management Information Base Interface** (DMIB) is an API to access DInfo objects. It is composed of the following primitives: getAbout(), getVersion(), get(), set(), function(), getObject(), nextObject(), restart(), and putObject(). It allows DInfoís objects to be manipulated for a particular management activity. To ensure the specific identification of each Dinfo object, the DMIB uses a specific ID similar to the SMI naming tree, whose leaves represent the DInfo objects.

Management applications have the possibility of creating specific information in the ADR and instrumenting them using these two APIs.

4.5 QoS Policy Definition

Policies are defined according to the SLS[6] agreed between the customer and the ISP. The negotiation process is not addressed in this work. We assume that this phase has been performed and that the SLS has been converted into a set of policy rules. These rules have a condition part and an action part. For example, if the ISP has made a commitment to provide a Gold Service to the customer on a certain Day/Time, then the policy rules will be defined as follows:

IF (PolicyTimePeriod == TimeValidityPeriod) **THEN** Provide Customer with a Gold Service.

This business level rule is converted into a network level rule identifying the customer flows at the ingress of the network and the corresponding DSCP value. It is created in the CIM repository and enforced in the ingress router.

IF (PolicyTimePeriod == TimeValidityPeriod) and (IPAddr IN Customer IP Address Set) **THEN** mark Flow with DSCP = Gold.

The basic concepts remain the same as those of IETF and DMTF policy work. However, in our proposal, it is also possible to use policies to deploy new services or sensors in the controlled router. For example, if an ISP has decided to shape the traffic aggregates of an AF Class only if more than 80% of the link capacity is used, then the policy rule will be defined as follows:

IF (LinkCapacityUsage > Threshold) **THEN** Install(Shaper (TBF(r,b)), AF).

This approach gives the operator a very powerful mechanism for efficiently controlling the network behavior, which does not exist in todayís networks.

4.6 TCB Definition and Deployment

The service code of the DACA architecture described earlier contains a set of active codes which implements different algorithms of the DiffServ model. It provides the ISP with a very flexible approach for specifying the behavior of its network. The internal structure of this repository is described in Figure 3. The highest level of the

[6] SLS : Service Level Specification, set of performance parameters that agreed in a SLA.

tree is the abstract DiffServ active services. Under this entity, we find five different sub-trees, each treating one aspect of TCB: classifiers, meters, actions, droppers and schedulers. All these entities are containers for active codes that implement different types of algorithms.

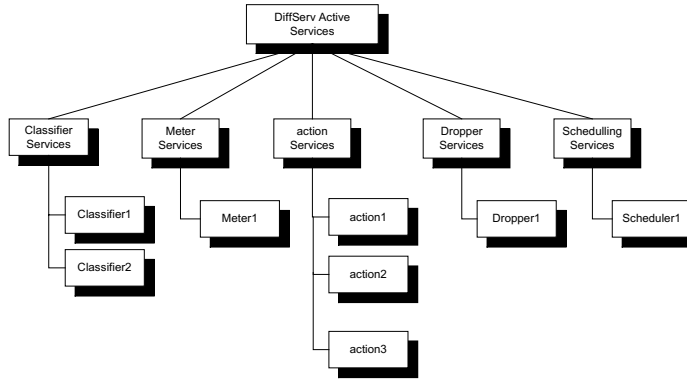

Fig. 3. TCB Active Services

The TCB in the DiffServ model consists of a set of TCB blocks connected in a specific sequence according to the desired behavior. Thus, in this case, the administrator defines the set of blocks to be deployed in the underlying routers in terms of his objectives. Once this has been done, the administrator asks the Code Distribution Point to install this path in the target node. This task is performed using a specific active installation protocol. The selected blocks are installed in the target routers and are then visible in the Service block at the execution environment layer of the active router. This installation affects only the capsules concerning this new TCB. In general, the manager will define various classes of TCB (called TCBClass). Each class will be assigned to a particular service and installed in the appropriate active node. A chain of DiffServ active services will compose each TCBClass.

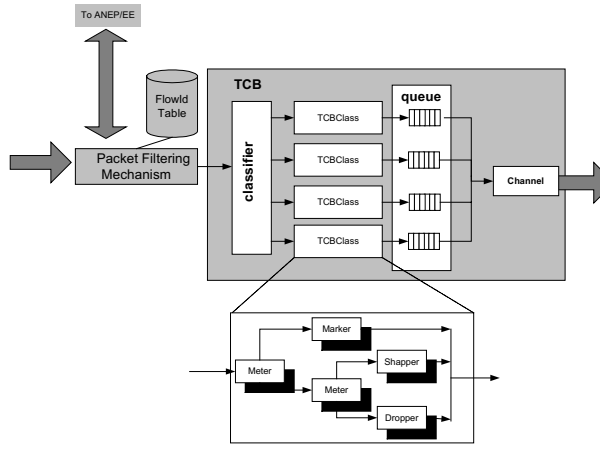

Fig. 4. TCBclass definition and TCB class path

When the Filter Service identifies a capsule belonging to an existing active application, it sends the capsule to the corresponding execution environment. If the capsule is not filtered, it is sent directly to the TCB block where is classified according to the DSCP and sent to the appropriate TCB. The TCBClass Service is composed of a set of TCBElem Services (Fig.4), each corresponding to a particular algorithm (i.e. meter, dropper, marker, etc.).

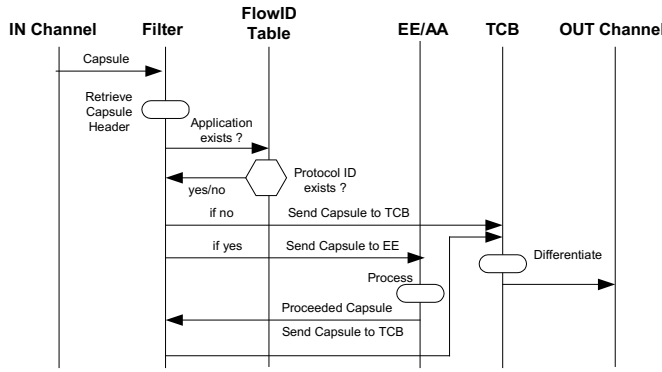

Fig. 5. Interaction process between the filter, the TCB and the FlowID table

If the administrator would like also to automate the installation and reconfiguration process and can do so by defining special policy rules dedicated to the installation of new TCBs. These rules allows the installation, modification or removal of TCBs according to a set of conditions such as the load on the network, performance data collected via the DMIB, etc. The interaction process between the Packet Filtering Mechanism, the TCB and the EE/Active Application that allows these functions to be carried out is described in the following Figure 6.

4.7 Sensor Deployment

When policies are introduced in the CIM repository, they are converted into a set of low-level policies, as described briefly in section 4.6. When a PDP detects the introduction of a new policy, it decides whether this policy should be handled locally or if it should be downloaded directly into the LPDP of the set of active routers concerned by the policy. The decision depends on whether the policy can be handled at the active router boundary. An example of a policy that can be handled locally is one that changes the configuration of the local metering algorithms of the active router depending on a Time Condition. A policy that can not be achieved locally is one that has a condition part containing a variable concerning the end-to-end delay between two access points. A sensor cannot resolve this information locally.

When the LPDP receives new policy rules to handle, it creates a new sensor that takes charge of the rules concerned. Depending on the condition part and the action part, the sensor could require functionalities that do not exist locally, which means that the LPDP may obtain these functionalities directly from the Provider Sensor Code Repository through the Code Distribution Point (CDP).

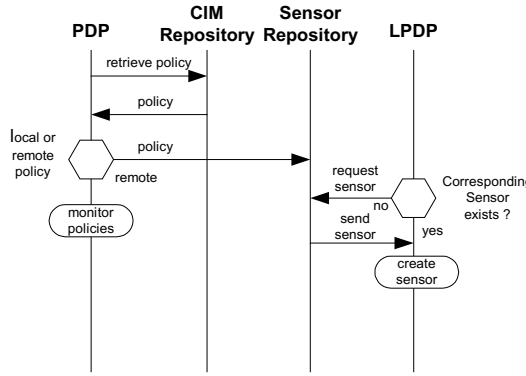

Fig. 6. Policy deployment using sensors

5. Prototype Implementation

Implementation of the DACA architecture is performed using a set of Java classes developed on the top of the ANTS[7] toolkit [9] used in the Amarrage project [10]. This toolkit allows the straightforward and efficient deployment of a full Java-based active node on IPv6. The ANTS is based on a mobile code as well as remote-loading-on-request and cache techniques. It allows new protocols to be deployed dynamically in network nodes without synchronization. ANTS defines capsules as active packets that customize network services and active nodes. The active nodes process incoming capsules and maintain soft-store. ANTS has a flexible code distribution scheme in which code capsule fragments can be loaded on request into an active node. Capsules are injected into the network by Applications and forwarded to their destination with customized routines. Each time a capsule enters an active node, it is treated according to the specifications defined in its own code. In our implementation we also use the Active Network Encapsulation Protocol [11] to provide another mechanism for routing user packets to particular EEs. The ANEP header includes a "Type ID" field which are assigned to specific execution environments. If a particular EE or various EEs are present at a node, packets containing a valid ANEP header with the appropriate Type ID will be routed to the appropriate EE.

Classes in DACA implementation represents the components defined in the DACA architecture presented in Figure 1. The classes are developed as Java 2 classes that extend or interact with the existing ANTS native classes. The ANTS classes implement the EEs and the channels.

The first component extending the ANTS platform is the **ActiveQoS** class, which corresponds to the Data Plan in the DACA architecture and is connected to ANTS via four JAVA interfaces, ***SendToEE***, ***ReceiveFromEE***, ***SendToChannel*** and ***ReceiveFromChannel***. The ANTS capsule is extended by **QoSCapsule**, containing two specific fields, ***FlowID and DSCP*** identifying the application that generates this capsule and the TCB that must be applied to it, respectively. The ActiveQoS Class

[7] ANTS : Active Node Transport System

consists of many classes. The first, the **Filter** Class, recognizes then intercepts the capsules belonging to active applications installed in the DACA node by checking the FlowId field in the QoSCapsule. If the capsule is not intercepted by the Filter class, then it is transmitted to the **Classifier** class, which checks the DSCP field and forwards the capsule to the appropriate **TCBClass**. The latter is a thread that contains a chain of **TCBElems** corresponding to the different algorithms of DiffServ. Each **TCBElem** present in the TCBClass applies a function ì**execî** to the capsule. This function is implemented as an abstract Java function which must be redefined for each particular DiffServ algorithm (i.e. meter, dropper, marker, etc). After that, the capsule is put in a buffer corresponding to the TCB queue. Lastly, we find the SchedulClass, which forwards the capsules to the output (i.e. to the ANTS channel). Like the TCBElem class, the SchedulClass class implements an abstract ì**Schedulî** function which must be redefined for all the scheduling algorithms. The DACA architecture implements two types of scheduling algorithm, namely Priority Queuing and Round Robin Scheduling, but it is possible to implement others.

To allow the TCBElem and SchedulClass algorithms to be changed on-the-fly, the DACA architecture uses a hierarchic tree repository, the DMIB, to manage its classes. This repository is able to keep and to schedule Services and other classes such as the PEP services, active sensors, MIB, TCBElem, SchedulClass, etc. In order for a class to be kept in the DMIB, it has to inherit from the DInfo class presented in section 4.5.

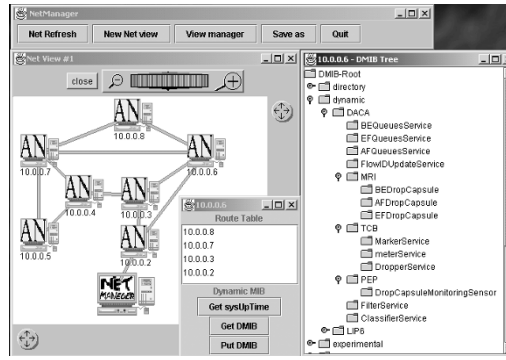

Fig. 7. Network discovery and code installation applications.

The GUI of the Active Management System of the DACA architecture implemented to manage the prototype is shown in Figure 7. It allows on-the-fly installation but also replaces, upgrades and deletes services and active sensors in all nodes of the DACA network.

The service and active-sensor repositories are currently implemented as class files in a directory while the CIM model is implemented as a straightforward file. Thus, three applications allowing the discovery of the active network topology are already operational, namely the manager of installed active services (e.g. TCB, Scheduler and active sensors), code installation and MRI object monitoring, as presented in Figure 7. Implementation is still under way to extend the existing components with new functionalities and to test the performance of the deployed network.

6. Conclusion and Future Work

QoS and Service Level Agreement management are two important concepts for ISPs. As customer needs change, ISPs have to deploy a quality of service solution that is manageable and customizable in order to address different customersí requirements. The solution we propose here aims to integrate the advantages of DiffServ, Active Networking and policy-based management in a single architecture. The proposed DACA architecture offers the ISP operator a set of powerful tools allowing him to enforce a business strategy in the network in terms of policies and to optimize network utilization by maintaining better control over the DiffServ resources.

At the moment, this work is still progressing toward complete implementation over DACA architecture but initial results show how easy it is for an administrator to modify the DiffServ configuration and how very promising this architecture appears for the future. The next step, therefore, will be to finalize the implementation of the PDP and the PEP in order to deploy the policy-based management part of the architecture and test it in a real trial.

7. Acknowledgments

This work was carried out partly within the framework of the Amarrage research project funded by the French Ministry of Research under the RNRT telecommunication research program. The authors express their thanks to the rest of the project team and, in particular, to the members of WP1 and WP4 for fruitful comments and discussion during the various meetings.

8. Bibliography

1. WROCLAWSKI J., ´ The Use of RSVP with IETF Integrated Services ª , RFC 2210, September 1997.
2. BLAKE S., ET AL., ´ An Architecture for Differentiated Services ª , RFC 2475, December 1998.
3. NICHOLS K., JACOBSON V., ZHANG L., ´ A Two-bit Differentiated Services Architecture for the Internet ª , RFC 2638, July 1999.
4. RAJAN R., VERMA D., KAMAT S., FELSTAINE E., AND HERZOG S., ´ A policy framework for integrated and differentiated services in the Internet ª , IEEE Network Magazine, vol.13, no.5, pp.36-41, September / October 1999.
5. DURHAM D., ET AL., ´ The COPS (Common Open Policy Service) Protocol ª , RFC 2748, January 2000.
6. MOORE B., ELLESSON E., AND STRASSNER J.,"Policy Core Information Model -- Version 1 Specification", RFC 3060, February 2001.
7. PSOUNIS K., ´ Active Networks: Applications, Security, Safety, and Architectures ª , IEEE Communications Surveys Magazine, 1st quarter issue, 1999.
8. CISCO White paper, ì DiffServ - The Scalable End-to-End QoS Modelî , www.cisco.com.
9. AMARRAGE PROJET, http://www.telecom.gouv.fr/rnrt/projets/res_d41_ap99.htm.

10. WETHERALL D. J., GUTTAG J., AND TENNENHOUSE D. L., ì ANTS: A Toolkit for Building and Dynamically Deploying Network Protocolsî , IEEE OPENARCH'98, San Francisco, CA, April 1998.
11. ALEXANDER D. S., et al., ´ Active Network Encapsulation Protocol ª , Draft, DARPA AN Working Group, July 1997.