Using COPS for managing Active Network Nodes

Abdelkader Lahmadi, Olivier Festor

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

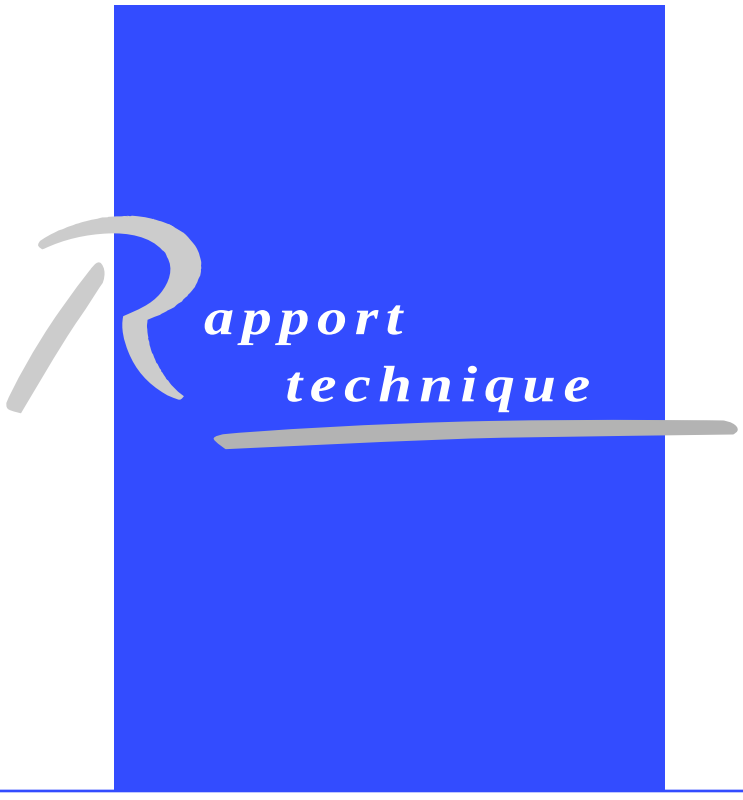
Using COPS for managing Active Network Nodes

Abdelkader Lahmadi,Olivier Festor

N° 0270

Octobre 2002

THÈME 1



ISSN 0249-0803

Using COPS for managing Active Network Nodes

Abdelkader Lahmadi,Olivier Festor

Thème 1 — Réseaux et systèmes
Project RESEDAS/MADYNES

n° 0270 — Octobre 2002 — 30 pages

Abstract: We propose a Common Open Policy Service (COPS) based architecture to manage Active Network Elements. We use the generic term `Active Elements` to refer to any active network component on a node, such as an Execution Environment (EE), an Active Application (AA) or any component of an EE or an AA. Our approach consists in the definition of a new client type, called COPS-ANEL, for the COPS protocol to support the management of these active network elements. The client type supports both the Outsourcing and the Provisionning models of COPS to achieve admission control, configuration and monitoring of Active Elements. We define a set of policies for the two models to be applied when an event occurs on the Execution Environement and needs a response from the policy server. By using a COPS-based management, an active network provider can control and monitor active elements. Our clients have been plugged into the FLAME Execution Environment, an active networking platform developed in our research group.

Key-words: Active Networks, Active Network management, COPS, Policy-based management.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Mise en œuvre du protocole COPS dans l'administration des reseaux actifs

Résumé : Nous proposons une architecture basée sur le protocole COPS (Common Open Policy Service) pour la gestion des éléments des réseaux actifs. Le terme élément désigne ici tout composant d'un nœud actif, comme un Environnement d'Execution (EE), une Application Active ou tout composant d'un EE ou d'une AA. Notre approche consiste à definir un nouveau type de client pour le protocol COPS nommé COPS-ANEL. Il a pour rôle la gestion de éléments actifs. Ce client supporte les deux modèles du protocole COPS: la délégation et l'approvisionement. Ces deux modèles servent pour le contrôle d'accès, la configuration et le monitorage des éléments actifs. Ainsi, nous avons défini un ensemble de politiques pour ce deux modèles, qui seront appliquées à toute occurence d'évènement dans l'Environement d'Exécution nécessitant une réponse du serveur de politique. Notre client a été implanté dans l'Environement d'Execution FLAME, qui est une plate-forme active developpée dans notre équipe de recherche.

Mots-clés : Réseaux actifs, Gestion de réseaux actifs, COPS, Gestion par politiques, COPS-ANEL

1 Introduction

An Active Network [2] consists of a set of nodes (not all of which need to be "active") connected by a variety of network technologies. Each active node runs one or more Execution Environments. Each Execution Environment (EE) implements a virtual machine that interprets active packets that arrive at the node. Active Packets contain programs or pointers to program modules, in addition to data. The Execution Environment executes such a program, called an Active Application, which possibly modifies the node's state and possibly generates further active packets to be sent over outgoing links. Users obtain services from active network via Active Applications (AAs), which program the virtual machine to provide an end-to-end service. In comparison to traditional networks, the active network approach offers more dynamic, flexible and customisable solution, thus allowing active networks elements to be managed 'on-the-fly' per execution environment, per customer. These elements need to be configured, controlled and monitored. This can be done manually or by the using of a management protocol. This document describes the use of the Common Open Policy Service `COPS` [5] protocol to manage these Active Elements (AA, EE), or any component of an AA or EE).

COPS is a query response protocol used to exchange policy information between a network policy server called the Policy Decision Point (PDP) and a policy agent called the Policy Enforcement Point (PEP). In order to be flexible, the COPS protocol has been designed to support multiple types of policy clients. For Active Networks we define a new client-type called **"`COPS-ANEL`"** (COPS- Active Networks ELements). COPS-ANEL uses the two common models supported by the COPS protocol: outsourcing and provisioning. The outsourcing model addresses the kind of events on the PEP that require an authorization. The provisioning model [3] used to feed the PEP with configuration elements taken from the Policy Information Bases (PIBs) [7] that define the data policy model. The PIB is maintained both by the PDP and the PEP. It is pushed from the PDP to the PEP for provisioning policy or sent to the PDP as a notification. Combined, the outsourcing related to the policy model and provisioning gives network managers centralized monitoring and control of active elements. In so doing, large number of management tasks are automated

via the use of COPS. Therefore, the usage of COPS on active networks needs to support the following Active Network crucial features:

- Admission control : provide an authorization mechanism to the EE to control the behavior of Active Applications. e.g an AA requested on a node can be accepted or rejected.
- Automated configuration: provide means to facilitate automated configuration both for AAs and EE.
- Performance guarantee : controls the performance behavior of AA's running on an EE and Active Packets arriving on the node (e.g some active packets will be dropped, if processing them will be a performance killer for the EE, AA that exceeds their ressource limitation will be stopped). The decision will be taken according to an execution model of this AA [6] and it will be dropped on all nodes.

We have implemented the COPS-ANEL client on the FLAME [1] Execution Environment [4]. FLAME is an active networking platform developed in our research group. The FLAME EE supports the creation, deployment and execution of Active Applications.
The report is organized as follows : In section 2, we present the COPS-ANEL architecture. In section 3 we define the functionnality of the COPS-ANEL client and the interaction between the client and the policy server. In section 4, we describe the information exchanged between the client and the server. Section 5 defines the protocol objects. Section 6 focuses on the detailed content of exchanged messages. The implementation layers are described in section 7. The use of the framework is illustrated in section 8. The proposed work is summarized in section 9.

2 COPS-ANEL architecture

Figure 1 depicts the overall architecture of the COPS-ANEL client. The architecture is loosely based on the IETF Policy Framework [9], whereby the main components of the pol-

[1] Project realized within a cooperation between ALCATEL and INRIA Lorraine under convention 101 F 0036 00 51596 12 2

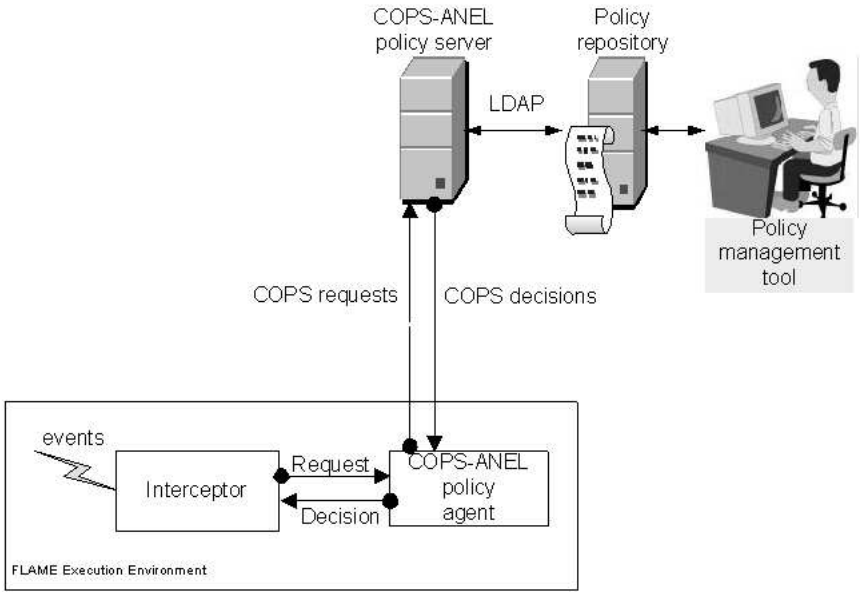


Figure 1: COPS-ANEL architecture

icy framework are the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP). The Execution Environment contains a policy agent which represent the PEP. It interacts with the PDP to support management functions. The interceptor is a generic component that issues the requests for the authorization or configuration operations e.g admitting an Active Application on the node. Each event that occurs on the Execution Environment and needs a decision will be captured by the interceptor, that will issue the corresponding request to the Policy Decision Point (PDP).

The policy server is responsible for taking decisions regarding authorization requests and collecting configuration elements for the EE or an AA in case of configuration requests. An administrator uses a policy management tool to define policies for authorization and provisioning, that are to be enforced in the Execution Environment. The policy repository is used to store the policies generated by a management tool. The repository can be implemented as an LDAP directory storing COPS-ANEL policies.

3 COPS-ANEL operations

Our management approach is based on policies. In the context of the Flame EE, we identified following policies as the minimal set of policies that need to be supported by any active network execution environment for management purpose. These policies are: admission control policies and configuration policies. Admission control policies define permitted operations for active networks elements and they are used to specify access control. Configuration poli-

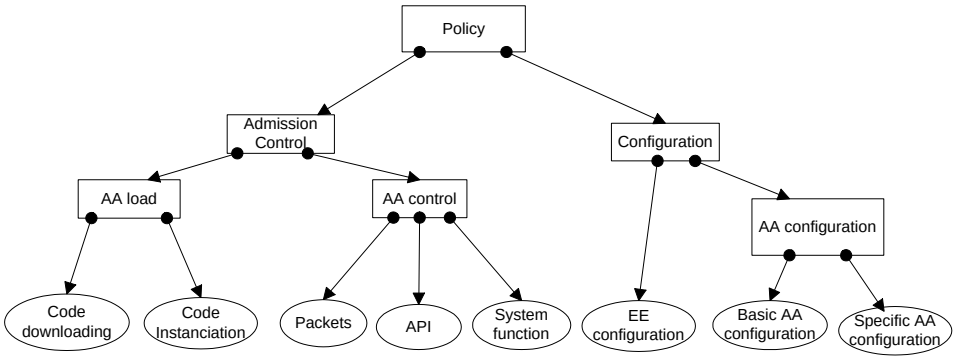


Figure 2: COPS-ANEL architecture

cies describe the needs of active elements to obtain their configuration from the management platform. Several types of policies are derived from these two basic classes in figure 2. Those policies will be applied according to active network operations on the active node.

3.1 Start of operations

In order to start operations, the policy agent must open the dialog with its PDP. First, a TCP connection is established between the client and the server. Over this connection, the Policy Agent sends a Client-Open message with its Client-Type attribute set to COPS-ANEL. If the PDP supports this client type, it responds with a Client-Accept (CAT) message. If the client type is not supported, a Client-Close (CC) message is returned by the PDP to the Policy Agent. After the reception of the CAT message, the Policy Agent can send requests to the server.

When the EE is activated on a router it should have it's initial configuration available.

To this end, it sends a "EE configuration request" to ask the PDP to provide the initial provisioning information which describes its capabilities. In response, the PDP downloads all provisionned policies that are relevant to the requesting EE. The EE will simply install the information received by the PDP and there is no acceptance / refusal of a configuration object. When the EE is running, Active Applications can be launched by EE adminstrators or by the arrival of Active Packets. These applications are activated automatically by the EE bootstrap or by an unsollicited decision message from the PDP (e.g launch the Accounting Active Application each day at 18:00 PM).

3.2 Active Network Elements Authorization mechanism

Outsourced Active Application Requests (AARs) are used by the EE to check authorization for Active Application operations. In response to these requests, the PDP sends a reply where it accepts or rejects the AAR. Below is a brief summary of Active Application operations that require authorization from the PDP.

- In Flame, the loading of an Active Application into a node can be triggered by three kind of events :

  1. The arrival of an active packet carrying information about its corresponding AA.
  2. From a User Agent connection by an adminstrator on a node.
  3. From an unsollicited decision message sent by the PDP asking the node to load an AA at a specfic time and date. This kind of decision concerns only transient AAs.

  Before loading the AA, the EE should download the AA code from a repository if the code is not present locally. Before starting the operation the EE sends a request to a policy server asking the authorization for downloading the code. In response to the request, a decision message is sent, with the sollicited flag set in the COPS message header, from the server to the Policy Agent indicating to accept or reject this AA. On reception, the Policy Agent handles the response, and notifies the EE of the decision. If the PDP specifies an `ACCEPT` decision, the EE will download the code for

this Active Application. If the PDP specifies a REJECT decision the request for this AA will be discarded. In case of an AA requested from an unsollicited DEC message sent by the PDP this authorization procedure is not initiated at all and download starts automatically.

- Once downloaded, the configuration of the AA is necessary before launching. To download the configuration, an Active Application request will be sent by the policy agent asking the policy server to provide the configuration for this AA. The request includes the EE identifier and the AA identifiers. In response, the server sends a DEC message containing the basic configuration information for this AA, e.g the transport layer identifier used by this AA, the ressources allowed (CPU, memory) and the number of failures (crashes) allowed. When receiving the DEC message, the policy agent decodes the bindings contained in the message and passes them to the EE. The EE will install the configuration elements and is now ready to instanciate the AA.
- The downloaded code instanciation is also subject to an outsourced decision.
- When the Active Application is running, the EE regularly checks to stop the execution of its activity. The decision to stop the AA is taken if it exceeds the ressources allowed or if it crashes (an abnormal stop). After stopping the AA, the EE sends a Report State to the PDP signaling that this AA is stopped. The report message includes the AA identifier and in case of an AA crash it includes also a failure object (see section 4). When receiving the report message, the PDP will send unsollicited DEC messages to other nodes asking them to stop or to blacklist this AA.
- When running, an Active Applications requirs APIs and invokes methods from theses APIs or system functions (e.g access the MIB or modify the routing table). When the AA requests an EE ressource (API, system function) the EE outsources an Active Application Request to the PDP asking the authorization.
- Some Active Packets may be ignored or dropped by the EE if their processing will degrade the overall performance. The decision to this kind of packets is outsourced to the PDP as part of the authorization policy. The AAR includes the source address of this packet and authentication information provided by the Active Packet itself.

3.3 Active Networks Elements provisioning mechanism

The configuration request contains (possibly empty) the identifier of the Active Application information (e.g for Active Application the AA name and the versionning number). It is empty in case of the EE configuration request. The PDP responds to the configuration request with a DEC message containing the set of configuration elements. The Active Application configuration may be managed by the EE or by the AA itself. Thus, each Active Application would have a basic configuration and a specific configuration. The Basic Configuration Information is requested by the EE and consists of a set of configuration elements for instanciating the AA. The Specific Configuration Information is requested by the AA when it starts running and consists of a set of configuration elements specific to each AA. The AA must have a Basic Configuration Information. This information will be useful to start it. The Specific Configuration Information is optional in the sense that it is up to the AA developer to decide wether or not such information is required for its application. Thus, it requires the development of a method within the AA code to decode the bindings contained in the DEC message sent as a response to such a configuration request.

4 COPS-ANEL specific information in COPS message

Like the COPS usage for RSVP [1], the usage of COPS on Active Network Elements needs to define its own client-specific information for various messages. This section describes the messages exchanged between the COPS server (PDP) and the COP-ANEL policy agent that carry client specific data objects. Each object has a class number (C-Num) which identifies the class of information contained in the object, and a class type (C-Type) which identifies the subtype of the information contained in the object.

4.1 Client-type field of the common Header of every COPS message

All the ANEL client-type COPS messages must contain the COPS Common Header with the 2-byte encoded Client-type field valued with the CS_ANEL_CLIENT_TYPE (Client type number to be assigned through IANA).

4.2 Context object

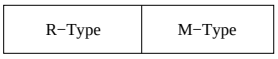


Figure 3: Context object

C-Num = 2, C-Type = 1

The Context Object (Figure 3) specifies the type of request that triggered the query. It is included in the REQ and the DEC message. For the COPS-ANEL client, the request Type flag is set either to 0x01 (Outsourced request) or 0x08 (Configuration request). The Message Type flag (M-Type) is used to specifiy the type of either the Client Specific Information Object carried in the REQ message or the Name Decision Data carried in the DEC message. Depending on the M-Type, a different content of the Client Specific Information object for the REQ message is used.

R-Type:

0x01 = Active Network Element request

- M-Type = 1 Outsourced Code Downloading
- M-Type = 2 Outsourced AA instanciation
- M-Type = 3 Outsourced AA Running
- M-Type = 4 Outsourced API Function
- M-Type = 5 Outsourced System Function
- M-Type = 6 Outsourced Active Packet

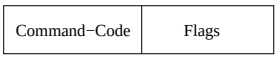
R-Type :

0x08 = Active Network Element Configuration request

- M-Type = 7 Configuration EE
- M-Type = 8 Configuration Basic Information AA
- M-Type = 9 Configuration Specific Information AA

4.3 Decision Flags Object for DEC message

Figure 4: Decision flags object



C-Num = 6, C-Type = 1

The Decision Flag Object (Figure 4) specifies the response for outsourced authorization requests. The answer will be in command-code field. The Command-code will be `Admit` , meaning a positive answer or `Reject`, meaning a negative answer. This object is included only in decisions related to Outsourced requests.

Commands :

1 = Admit request

2 = Reject request

Flags : Flags are ignored.

4.4 Named Decision Data Object for DEC message

C-Num = 6, C-Type = 5

This object is carried on DEC messages only on response to a Configuration request. It contains configuration objects either for the EE or the AA.

4.5 Client Specific information Object for REQ message

C-Num =9, C-Type = 2

The Client Specific Information Object carried in the REQ messages for COPS-ANEL client contains the description of the Active Application and has a different format depending on the type of the request, as specified by the context object.

4.6 PEP Identifier Object

C-Num = 11, C-Type = 1

The PEP identifier object is used to identify the COPS-ANEL client (policy agent) to the remote PDP (policy server). In our case, this object will be the identifier of the Execution Environment. One of it's binding IP address will be choosen for being this identifier.

4.7 Report Type Object

C-Num = 12, C-Type = 1

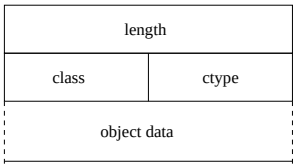
The report type is contained in the Report State message. For COPS-ANEL client the Report-type is always 2 (Failure).

4.8 Client Specific Information Object for Report State message

The Client Specific Information Object for Report State is used to communicate Active Application problems to the PDP. It will carry either a failure object when the Active Application crashes or a Ressource Object when the AA exceeds its allowed ressource.

5 COPS-ANEL protocol objects

Figure 5: Policy object format



COPS-ANEL uses new COPS protocol object that carry client-specific information. The format for these new objects is depicted in the Figure 5.

The body object contains the data that carried a parameter of the policy. For each object we will specify the object type, a description, the COPS operation (outsourcing or Configuration), the message Type on which it will be encapsulated (REQ or DEC) and the data that is carried.

5.1 AA identifier object

Object Type	Description	Operation	Message	Data
1	Identifier of the active application, this will be done by using AA name and versionning numbers.	Notify Install	Request Decision	STRING version STRING aaname

5.2 User authentification object

Object Type	Description	Operation	Message	Data
2	This will be either a `user:password` token from the User Agent Connection or the authentification information provided by the Active Packet	Notify	Request	STRING username STRING pwd

5.3 Ctype object

Object Type	Description	Operation	Message	Data
3	The connection type to the AA. Two possibilities are: User Agent or Active Packet	Notify	Request	INTEGER ctype

5.4 IPSource object

Object Type	Description	Operation	Message	Data
4	Either the address source of the User Agent connection or the address of the Active Packet emitter, the choice will depend on the connection type (CType)	Notify	Request	ADDRESSTYPE addr

5.5 URL object

Object Type	Description	Operation	Message	Data
5	The URL used to download the code.	Notify	Request	STRING url

5.5.1 Failures object

Object Type	Description	Operation	Message	Data
6	Number of times and palces where the AA has terminated due to a failure (crash, ...).	Notify	Request	INTEGER failures

5.6 AA Type object

Object Type	Description	Operation	Message	Data
7	Type of AA, either persistent or transient.	Notify	Request	INTEGER aatype

5.7 AARess

Object Type	Description	Operation	Message	Data
8	Ressource allocated to an active application, including CPU and memory percentage.	Notify Install	Request Decision	DOUBLE pctcpu DOUBLE pctmem

5.8 Function object

Object Type	Description	Operation	Message	Data
9	Name of a function that will be invoked.	Notify	Request	STRING funcname

5.9 API identifier object

Object Type	Description	Operation	Message	Data
10	Identifier of the API, the name and versionning number will be used	Notify	Request	STRING apiname STRING version

5.10 Signature object

Object Type	Description	Operation	Message	Data
11	The digital signature of the Active Application code. This object defines also the type of the algorithm used to obtain the signature.	Notify	Request	INTEGER Stype STRING signature

5.11 Permission Object

Object Type	Description	Operation	Message	Data
12	The permission object represents access to the EE ressources. Each permission has a type which identifies the ressource's type. A target name specifies the ressource name and an action field required for many permissions types such as FilePermission (where it specifies what type of file access is permitted.	Install	Decision	INTEGER PERMType STRING PERMTarget INTEGER PERMAction

5.12 AA defined Object

Object Type	Description	Operation	Message	Data
13	variable length object and its internal format should be specified in the Active Application code.	Install	Decision	INTEGER ObjType STRING Data

6 COPS-ANEL Message Content

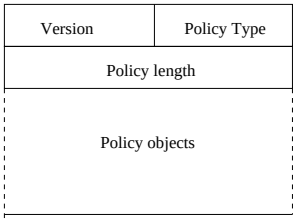
COPS-ANEL messages contain policy control information and are carried by COPS messages.

6.1 Request Message (REQ) PEP->PDP

The REQ is sent by the COPS-ANEL client to issue an Active Network Element Request (ANER) to the PDP. The Outsourcing requests contain an authorization request for a single Active Application. Each request has the following format:

<Request> ::= <Common Header>
<Client Handle>
<Context>
[<Client SI : ANER data>]
[<integrity>]

Figure 6: COPS-ANEL policy format



The context object specifies the request types (Outsourced, Configuration) and the requested operation defined by the Message-Type field. The ClientSI data object is used to specify the parameters of the request. This object format is depicted in Figure 6.

6.1.1 Authorization ClientSI data

If the Context Object specifies an Outsourced request, the ClientSI object contains the parameters of the authorization operation requested. The different operations that have been identified as requiring authorization from the Policy Server (PDP) are :

- Active Application code downloading.
- Active Application code instantiation.
- Invoking an API function by the Active Application.
- Calling a system function by the EE or an Active Application
- Processing an active packets.

These operations are detailed in the next section.

6.1.2 Code downloading

Message Type	Description	Paramters
1	When code implementing the AA is not present in the EE, it will be downloaded accross the network. The control can go as far as ensuring that no code is remotely installed, or that code is only downloaded from a well known site.	EE Identification AA Identification Signature CType IPsource URL

6.1.3 AA instantiation

Message Type	Description	Paramters
2	The code is present on the EE, and a request either by a User Agent or by packet has arrived for this AA, the decision to instantiate the code will generally be based on the AA name and the number of times this application has wrongly behaved (crash).	EE Identification AA Identification failures CType

6.1.4 API function

Message Type	Description	Paramters
4	This allows a control over the API an AA can load, as well as on the methods it can invoke (if it has been taken into account during the API development).	EE Identification AA Identification API function

6.1.5 User Agent or System function

Message Type	Description	Paramters
5	The system functions here are a superset of the user agent functions. They are important functions that can be triggered by the user agent or by the application when processing some packets. No distinction is made between the EE and the AA, but the AA case will need extra programming.	EE Identification AA Identification CType IPSource user-auth function

6.1.6 Packets

Message Type	Description	Paramters
6	It would be possible to ask if a packet should be processed by the AA, but doing so will certainly be a performance killer.	EE Identification AA Identification CType IPSource user-auth

6.1.7 Configuration ClientSI data

If the Context Object specifies a Configuration request the ClientSI object contains either the Identifier of the Active Application for which the configuration is requested. It is empty when the Execution Environment configuration is requested.

For an Active Application, we have defined two kinds of configuration :

- The Basic Active Application which is requested by the EE to configure the AA before loading it. In response to this request, the DEC message contains general configuration objects which are defined for all AAs. This configuration includes the programming language of the Active Application (Java, C), the transport protocol (UDP, ANEP,

...), the allowed resources, the number of times it is allowed to crash and its corresponding permissions to use node ressources (APIs and functions).

- The Specific Active Application configuration is requested by the AA when it starts. The DEC message for this request contains configuration objects which are specific to each AA. This configuration includes objects defined by the Active Application developer and are transparent to the COPS-ANEL module. These objects will be decoded by a method implemented within the Active Application code. When receiving the configuration elements, the Active Application policy agent decodes the bindings in the DEC message by using the decode method specified by the AA.

6.2 Decision Message (DEC) PDP->PEP

The DEC message is the answer of the PDP to a request sent by the COPS-ANEL policy agent. Unsollicted DEC messages can be sent for this client type (therefore the sollicited decision flag is unset). The DEC message for the COPS-ANEL client type has the followinfg format:

```
<Decision Message> ::= <Common Header>
                       <Client Handle>
                       <Decision> | <Error>
                       [<integrity>]
```

Each DEC message contains a single decision. If the Context Objetc specifies a reply on an Outsourcing request, the Decision Object will have the following format:

```
<Decision> ::= <Context>
               <Decision : Flags>
```

where the Decision Flags object contains the response of the PDP. The response will be `Accept` or `Reject`.

If the Context Object specifies a reply on a Configuration request, the Decision Object will have the following format:

```
<Decision> ::= <Context>
               <Decision : Flags>
               <Decision : Named Decision Data>
```

The Command-Code field on the Decison Flags object will be always `Install` for sollicted decisions. Unsollicted DEC can have decisions flags set to either `Install` or `Remove`. COPS-ANEL Configuration objects are contained on the Named Decision Data.

In case of an answer to EE Configuration Request, the Named Decision data will have the following format:

```
<Decision Named Data> ::= <Node ID>
                          <Transport Protocol>+
                          <Bootstrap AA>+
```

In case of an answer to a Basic AA Configuration request, the Named Decision data will have the following format:

```
<Decision Named Data> ::= <AA ID>
                          <Language ID>
                          <Transport Porotocol>
                          <Permissions>+
                          <AA Ressources>
                          <Failure>
```

In case of an answer to a Specific AA configuration request, the Named Decision data will have the following format:

```
<Decision Named Data> ::= <AA ID>
                          <AA defined object>+
```

The AA defined Object will be specific for each Active Application and will be defined by the developer of this AA. In so doing, a method that decodes this defined objects must be implemented in the AA code.

Unsollicted DEC messages are triggered by two events:

- Launch an Active Application at a specific time. In this case, the DEC message has an `Install` decision flags and the Named Decision Data has the same format as an answer to a Basic AA Configuration request. After receiving this DEC message, the EE will launch the corresponding AA and it will find the Basic Configuration objects on the same message so it will not need to send a Basic Active Application Configuration Request to the PDP.
- Stop an Active Application on all nodes when it crashes on a single node. The DEC message will have a `Remove` decision flag and the Named Decision Data contains only the AA Identifier Object. When receiving this message the EE will stop the corresponding Active Application and it will be blacklisted.
- Some nodes may be elected not to execute particular packets. The node will act as an IP-Like forwarding engine for theses packets. When this happens, the PDP sends an unsollicited DEC message to the EE that indicates to forward only these packets. The DEC message will have `Install` decision flags, the Named Decision Data will contains the Active Application Identifier and An IP-Like Forward object.
- When permissions for an Active Application changes, the PDP will send an unsollicted message to the COPS-ANEL agent to update its local copy of permissions for the coresponding AA.

6.3 Report State PEP->PDP

For the COPS-ANEL client type, the Report State message is sent in case of an Active Application failure. We distinguish two failure types that should be considered: Active Application crash and ressource allowed exceeds. In this case the EE will send a report state message to inform the PDP of an abnormal behavior of this AA. On reception of such a Report State, the PDP could send Unsollicited DEC messages to all nodes to stop this AA. The Report State message for the COPS-ANEL client type has the following format:

```
<Report State Message> ::=<Common Header>
                          <Client Handle>
                          <Report Type>
```

<Client SI :
<AA Identifier>
<Failure Object> | <AA Ressource> >
[<Integrity>]

The Report Type will always be `Failure`. The Client Specific Information Object will carry a failure object in case of an AA crash or an AA Ressource object in case of an AA that exceeds the allowed ressources.

7 COPS-ANEL implementation and testbed

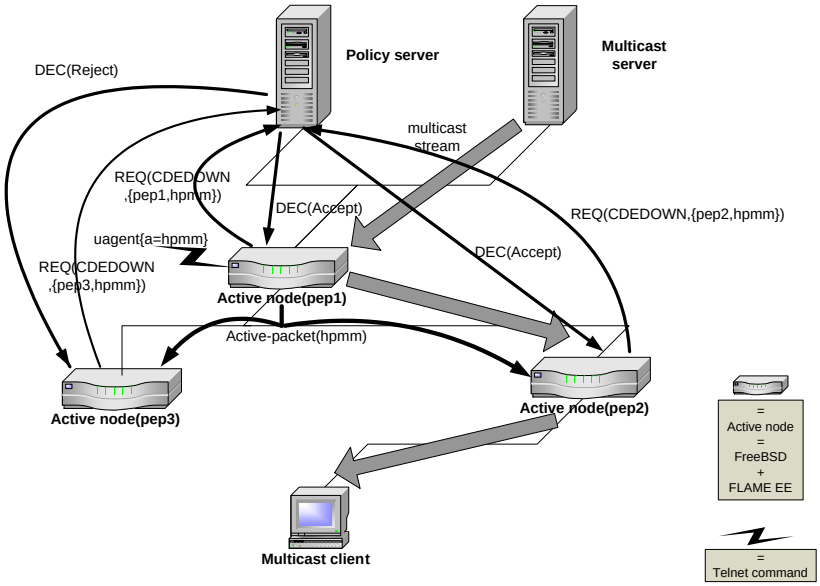


Figure 7: The testbed

In this section we present the platform we are deploying to test the COPS-ANEL protocol. This platform, illustrated in Figure 7, is composed of active nodes running the FLAME Execution Environment. They are embedding the COPS-ANEL client and a policy server supporting the COPS-ANEL protocol. In addition to the COPS-ANEL components, sev-

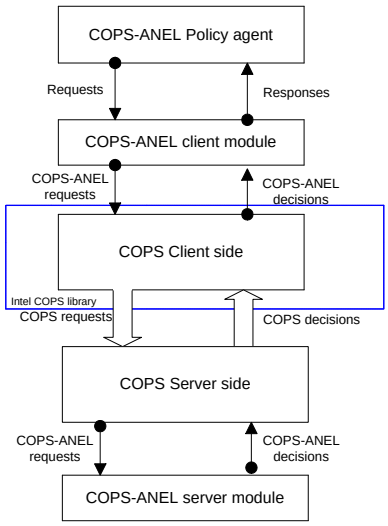


Figure 8: COPS-ANEL layers

eral other elements have been introduced. These elements consists of a multicast server, a multicast client and an active application (HPMM) that has in charge the monitoring of the multicast traffic. Our aim is to dynamically deploy the HPMM AA on pep1, pep2 and to forbid its deployment on pep3. The pep1 and pep2 have the permission to download the code of the HPMM AA, however the pep3 does not own this permission. A customer or an administrator will try to launch the HPMM AA on the pep1 to monitor the multicast traffic coming from the multicast server. The active node pep1 sends a COPS request and gets an Accept response. Thus the HPMM AA is loaded on the pep1 node. The HPMM AA running on pep1 broadcasts active packets to activate the HPMM protocol on the downstream nodes. The pep2 node succesfully loads the HPMM AA after receiving an Accept decision from the PDP. The pep3 will try to load the corresponding code but before that it sends a COPS request to the server asking the permission. The server responds with a Reject decision and the code loading is failed.

The client side implementation of COPS-ANEL is based on the Intel COPS SDK [8]. As depicted in Figure 8, the different layers of the implementation are divided in a client and a service side. The client side is plugged into the FLAME Execution Environment.

It is is composed of: the COPS-ANEL policy agent, the COPS-ANEL client module and the COPS client. The server side is a COPS based server that contains an additional module that handles COPS-ANEL messages. The Execution Environement policy manager creates Active Application request messages, either for outsourcing or for provisioning. These messages are delivered to the COPS-ANEL layer, which represents the client specific layer for the COPS protocol. This layer encapsulates the messages in the COPS requests and sends them to the policy server using the Intel COPS-SDK library. The intel COPS-SDK library uses the callback mechanism to notify the COPS-ANEL policy manager of the arrival of decision messages from the policy server.

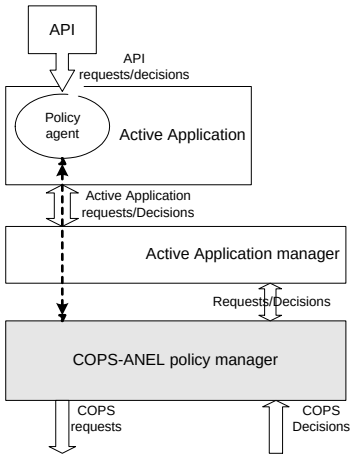


Figure 9: Policy agent architecture

Furthermore, each Active Application can have its own policy agent (Figure 9) that acts with the COPS-ANEL policy manager plugged into the EE. This policy agent has the role to check permissions for the Active Application. Each permission specifies a permitted access to a particular ressource, such as APIs and system functions. The Active Application policy agent provides a set of `checkXXX()` methods. Examples are `checkNETCAP` (for the netcap API), `checkCONNECT` (for the connect system function). Permissions will be provided by the COPS-ANEL policy agent from the PDP server when requesting the basic configuration of the AA. Thus permissions will be maintained by the COPS-ANEL to check Active

Application privileges in the AA policy agent. When permissions of an Active Application are updated on the policy repository the PDP sends an unsollicited message to the COPS-ANEL policy agent to update those permissions for the corresponding AA. During the Active Application startup procedure, the policy agent provides the Active Application with its specific configuration.

8 Illustrative examples for COPS-ANEL client

This section details an example for loading an Active Application on a node, and showing the running model for this AA.

```
PEP --> PDP REQ:= <Handle H>
                  <Context Outsourced AA Request, Code Downloading>
                  <ClientSI:
                        AA Identifier     :Ping,1
                        Signature         :XXXXX, MD5
                        CType             :User Agent
                        IPSource          :152.81.7.45
                        URL               :AARepository.loria.fr
                  >
```

The PDP accepts the request:

```
PDP --> PEP DEC:= <Handle H>
                  <Context Outsourced AA Request, Code Downloading>
                  <Decision : Command = Accept>
```

The EE will download the code from the repository. Then it will send a A Basic Configuration Request to the PDP.

```
PEP --> PDP REQ := <Handle H>
                    < Context Configuration Request,
                      Basic Configuration AA
```

```
        >
        <ClientSI:
            AA Identifier : Ping,1
        >
```

The PDP will send the Configuration Objects for this AA.

```
PDP--> PEP DEC := <Handle H>
        <Context Configuration Request,
         Basic Configuration AA
        >
        <Decision : Command = Install>
        <Named Decision Data:
            AA Identifier          : Ping,1
            Language ID            : C
            Permissions            : PERMAPI, logger
                                     PERMNET, socket
                                     PERMFILE,"/tmp", r/w
                                     PERMRUNTIME,thread
            Transport protocol ID  : UDP
            Allowed Ressource      : cpu = 0.1 mem = 0.3
            Failure                : 0
        >
```

The EE now instanciates the code for this AA. To this end, it sends an Outsourcing request to the PDP.

```
PEP --> PDP REQ:= <Handle H>
        <Context Outsourced AA Request,
```

```
 Code Instanciation
>
<ClientSI:
        AA Identifier      :Ping,1
        Failures           :0
        CType              :User Agent
>
```

The PDP accepts the request. Now the EE may instanciate the AA code. After instanciation, the Active Application is operational. If the AA crashes or if it uses to much ressources, the AA will be stopped and the EE will send a Report state message to the PDP.

```
PEP --> PDP RPT:= <Handle H>
                  <Report Type : Failure>
                  <ClientSI:
                        AA Identifier :Ping,1
                        Failures      :1," Send Active Packet Function"
                  >
```

9 Conclusion

In this document, a policy based management architecture for managing active network elements has been proposed. This approach covers three management functional areas: (Configuration, Performance, Security). This architecture uses the COPS protocol, which allows us to define a new client type named `COPS-ANEL`. The architecture is a centralized one, but it can be extended to be distributed by using of caches that will be maintained by each policy agent on an execution environment. Our future work will concentrate on the definition of a PIB for our client. This PIB will contains both information for the

outsourcing and the provisionning models. Additionally, a further investigation has to be done concerning the implementation of a prototype of a Policy Server supporting our client.

References

[1] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry. **COPS usage for RSVP**. Request of Comments 2749, Internet Engineering Task Force, January 2000.

[2] K. Calvert, ed. **Architectural Framework for Active Networks**. AN draft, AN Architecture Working Group, 1998.

[3] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith. **COPS Usage for Policy Provisionning (COPS-PR)**. Request of Comments 3084, Internet Engineering Task Force, 2001.

[4] Stephan D'alu and Olivier Festor. **Flame, A dedicated Active Plate-Forme for Internet services supervision**. *Proceedings of CFIP2002, Montreal Canada*, May 2002.

[5] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. **The COPS Common Open Policy Service Protocol**. Request of Comments 2748, Internet Engineering Task Force, January 2000.

[6] V. Galtier, K. Mills, Y. Carlinet, and A. Bush, S .and Kulkarni. **Predicting resource demand in heterogeneous active networks**. *Proceedings of MILCOM 2001*, October 2001.

[7] K. McCloghrie, M. Fine, J. Seligson, K. chan, S. Hahn, R. Sahita, A. Smith, and F. Reichmeyer. **Structure of Policy Provisioning Information SPPI**. Internet draft, work in progress, Internet Engineering Task Force, May 2001.

[8] Intel Labs Research and Developpement. **Intel COPS Client Software Development Kit**. http://www.intel.com/labs/manage/cops/.

[9] R. Yavatkar and al et. **A Framework for Policy-based Admission Control**. Request OfComments 2753, Internet Engineering Task Force, January 2000.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803