A HIGHLY DISTRIBUTED DYNAMIC IP MULTICAST ACCOUNTING AND MANAGEMENT FRAMEWORK

Hassen Sallay, Olivier Festor
The Madynes Research Team
LORIA-INRIA Lorraine
615, rue de Jardin Botanique
54602 Villers-lęs-Nancy, France
Tel: +33 (0) 383.592.000, Fax: +33 (0) 383.278.319
Hassen.Sallay@loria.fr, Olivier.Festor@loria.fr

Abstract: We present a highly distributed management architecture dedicated to IP multicast services. This architecture relies on a three level hierarchical model over which both management data and functions are distributed. We show how the architecture can be used to support an extended multicast accounting algorithm which adapts itself to the dynamics of a multicast tree and detail the implementation of the proposed framework using active network technology.

Keywords: Dynamic Accounting, Multicast Management, IP Accounting, Active Networks.

1 Introduction

As predicted in the last years [17], IP multicast is slowly moving from an optional feature in some networks to a primary service in the Internet. Multicast services gain more and more importance and become increasingly attractive. Multicast protocols are now deployed in almost all router products available on the market and they support a wide variety of applications (cooperative work, video-conferencing, tele-teaching, CDN update, . . .).

With the advent of protocols like PIM-SSM [1] or EXPRESS[2] [26, 13] which are scalable and which cover a wide number of applications, multicast deployment and usage will grow even faster. Unfortunately, this success curve is currently slowed by several factors. One of these limiting factors, is the lack of integrated management solutions able to cope with all components entering in the multicast service delivery chain [19].

Efficient management of multicast services is both a crucial requirement and a major challenge for multicast services. Building management solutions which can address dedicated security, accounting and fault management for multicast is a key to their successful deployment. But, due to its differences with unicast communications, namely the potentially huge number of participants together with its tree-based connectivity which dynamically evolves over time, the design and implementation of the

[1] Protocol Independent Multicast - Single Source Mode
[2] EXPlicitly Requested Single-Source Multicast

afore mentioned management services are much more complex to achieve. To be efficient in the multicast context, four factors need to be considered :

1 the multicast dynamics,

2 the scale factor,

3 the degree of specialisation for the management functions, and,

4 the ease of deployment and integration with the legacy.

Multicast services are dynamic by nature. This dynamic behaviour is mainly generated by join/leave operations of group members. The diffusion tree follows this dynamic behaviour, either expanding or reducing itself over time. In such an evolving environment, the signalling and management plane loops sometime share very close timing requirements, management being forced to follow near real-time constraints found in the control plane.

Multicast is in theory the solution which scales best. The problem of scalability is in itself not bound to the multicast technology used but rather to the implementation and the associated management solutions. For management solutions dedicated to multicast services, scalability is essentially a gradient of :

- the number of multicast groups to manage,
- the number of nodes of each managed multicast tree,
- the frequency of change in groups,
- the overhead of management and signalling.

Designing efficient management solutions that can scale up to thousands of groups and millions of group members, while maintaining a limited management overhead remains a very challenging task.

Management functions such as accounting and security have to integrate the specific nature of multicast in the definition of the corresponding management applications and related metrics. For example, it is not an easy task to maintain an up-to-date central knowledge of the number of participants together with their distribution in a multicast tree. Thus, if one of the accounting approach relies on considering the exact number of participants, these dynamic operations (join/leave) must be traced in a very precise way. Moreover new parameters like the number of active groups, the overall number of groups and members in a given network, the tree topology and the number of links, traffic volume and memory usage within routers must be considered in the cost allocation process[3] for multicast service.

For the security function, if a group has strong security requirements, cryptographic keys have to be generated and distributed each time a member joins or leaves a group to guarantee that members who left do no longer have access to the group's exchanged data and that members who join do not have access to data that was exchanged before they joined.

To be well integrated, any management solution must provide gateways and interfaces to standard protocols and frameworks. To be efficient, they also need to take

[3] i.e. the application of a set of strategies which enables a cost to be associated to each participant including the savings generated by the use of multicast.

advantage of the most recent technologies to facilitate their deployment and configuration. In the context of IP multicast management, this means that gateways with the SNMP world must exists and that the management platform must be sufficiently open and dynamic to adapt itself to changes and to be capable of supporting new types of management algorithms.

The goal of the work undertaken in our group, is to build such a framework to manage IP multicast services. Based on the above requirements and on an in-depth study of multicast management, we propose a management framework that is highly distributed and extensible to fit both very flexible service level constraints and very dynamic network conditions. In addition to describing the core concepts of the architecture, we show how it was implemented using an active network framework. To illustrate the applicability of the framework, we propose an extension of a recently described cost allocation algorithm and show how the framework can cope with the added dynamics and show how it can be used in the context of fault management.

The paper is organised as follow. Section 2 provides a description of work related to multicast accounting and security management. The proposed architecture is presented in section 3. The technology choices made to host this architecture are described in section 4. Section 5 is dedicated to the description of the application of the architecture for cost allocation and accounting purpose followed by the implementation details (section 6). Some conclusions together with an outlook for future work are given in section 7.

2 Related work

Several approaches have been designed and proposed so far for the management of multicast communications. Each of them targets a set of specific management functions.

Within IETF, the AAA[4] model [23, 7] has been designed. This architecture interacts with the various services it applies to through specific modules. Diameter [6] is the communication protocol used among different entities of the AAA architecture. This protocol can be extended to meet the requirements of specific target applications. So far, no extension was proposed to embrace multicast services. Thus, using the architecture for multicast management is not feasible as is and its scalability has not been investigated in this context.

HERZOG et al. [12, 11] propose different strategies to allocate the cost of a multicast tree over its members. A cost allocation mechanism and a LPM[5] architecture defining components in charge of access control and accounting have also been proposed in this work. An example of the proposed strategies is ELSD[6]. This strategy divides equally the cost of a link over all members in the downstream of the multicast tree. This strategy, which is the most equal for a tree/source schema, has been implemented in the MultiCost prototype.
Unfortunately this strategy, together with the LPM architecture, do not take into account the dynamics of a multicast tree and consider the costs associated to a link to be static. LPM also implements access control but does not support any key distribu-

[4] Authentication Authorisation Accounting architecture
[5] Local Policy Modules
[6] Equal Link Split Downstream

tion mechanism nor does it support any fault management function. The use of a tree metric for charging multicast communications is described in [8].

Designed by HOLBROOK et al., EXPRESS[7] [26], is a multicast communication model dedicated to the single source multicast schema. Within EXPRESS, a protocol named ECMP[8] offers support for accounting tasks and integrates options for security features in addition to routing and membership management. ECMP assumes the accounting architecture is based on a centralised architecture (the source being the center) from where accounting campaigns on a well known attribute are initiated on demand. Data collected through this process are global data related to the multicast tree like number of members, number of branches, ...
Collecting more fine grained data for more precise accounting strategies, e.g. number of members beyond a given router, generates a huge wave of requests over all nodes of the multicast tree. Centralised management at the source is known for its bottleneck, its intolerance to any fault, and the overload generated on the network with often unnecessary management traffic. ECMP does not consider the dynamics of the multicast tree, nor does it implement any cost allocation strategy. The protocol remains dedicated to source specific multicast trees and no implementation of this part is known to us so far.

Solutions for securing multicast can be found in [24, 10, 16, 4]. Centralised key generation by a KDC[9] in charge of manually distributing these keys is proposed in [24]. This solution has clear scalability problems due to the off-line coordination effort that needs to be provided between the KDC and the members of the multicast service. Several decentralised and automated solutions are proposed in [10, 16, 4]. These architectures distribute key management functions among the involved entities and have a much better scalability.

In [1], an architecture for fault and quality management of a multicast link based on SNMP is proposed. Another architecture, based on the integration of existing management tools like MTrace and MRM[10] [2] is also proposed in a previous work of our group [18]. While these architectures offer a good level of integration and interesting functions like limited fault management, topology discovery and test generation and processing, they face scalability issues and do not address security nor accounting. A dynamic topology discovery approach for IP multicast is proposed in [20]. An alternative is proposed is [15]. A more complete monitoring environment called MCPM [14] exists but does not address accounting.

3 A highy distributed architecture

In this section we present the basics of the management architecture we designed to overcome the limitations of existing approaches and fit the requirements identified in the context of IP multicast.

3.1 Design choices

The main concept behind our architecture is to distribute as much as possible both management data and processing units to maintain a high degree of dynamics and

[7] EXPlicity REquested Single-Source
[8] Express Count Management Protocol
[9] Key Distribution Center
[10] Multicast Reachability Monitoring

ensure scalability. This distribution follows a pattern that enables composition and coupling of operations. Our architecture embraces this principle to meet the requirements identified in the introduction.

We consider the multicast dynamics as a two facets entity : one at the micro-dynamics level and another at the macro-dynamics one. Through the micro-dynamics level, the evolution of the group members (join/leave operations to a multicast group) can be monitored. The macro-dynamic level represents the evolution of the multicast tree through expansion/reduction operations (i.e. routers leaving/joining the multicast tree). As we will see below, differenciating these two facets, leads to the design of a scalable management solution especially in terms of the number of group members supported.

The overhead generated by the transport of management data related to a multicast service can also be reduced by setting the granularity of the data to be exchanged. Thus, only data that is mandatory for a given management task is sent over the network. Management data storage becomes necessary at the places where these data are produced and tasks can be delegated to the various network nodes that participate in the multicast service delivery chain. In this case, the processing distribution follows the data distribution providing the most efficient solution for large groups.

3.2 Global architecture

The proposed architecture provides a 3-level hierarchy : the source level, intermediate nodes level and the edge nodes level. At each level of the hierarchy a dedicated management agent is operational (see Figure 1).

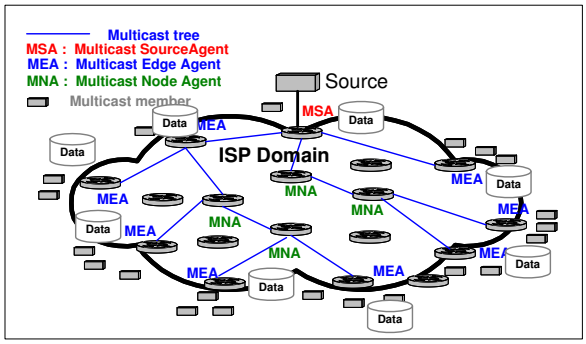

Figure 1. Global management architecture

At the multicast source level, a Multicast Source Agent (MSA) is in charge of the management activities. The instantiation and content of this agent can be part of the SLA/SLS that have been defined for the service between the service provided and the multicast service customer. The agent itself can be instantiated within the source which delivers traffic if a single administrative domain is considered. When multiple sources exist in one multicast session, the MSA agent is placed in the rendez-vous point node. In a multi-domain environment, we can extend our architecture by considering one administrative domain as a peer. The multicast service will be established by the cooperation of the different peer representing different domains. The source agent

can play the role of the peer and negociate the service SLA setup among different ISPs concerned by the service. In this paper we consider only the single domain scenario, and leave the multi-domain scenario for a future work.

The MSA hosts a data storage facility that has the entire data related to the service. The collection algorithm initiated to feed the database follows the dynamics of the multicast tree. The database is updated each time an edge node (router) joins or leaves the tree.

At the edge node level, Multicast Edge Agents (MEA) are deployed. These agents manage the micro-dynamic facet of the multicast management. They maintain a local view of session join and leave operations performed by end users. To this end, these agents interact with the membership management protocols like IGMPv3 [5] or MLDv2 [9] through dedicated interfaces and build a local database which holds very detailed data about each member. These agents push the data forward to the concerned source agents (MSAs) only after all members of a session have left.

Nodes of the intermediate level (not the source, nor the edges) hold specific agents called Multicast Node Agent (MNA). Deployment or activation of these agents is done dynamically according to the expansion/reduction of the multicast tree of the managed service. This dynamic deployment implements the macro-dynamics facet related to topology changes in the multicast tree. These agents interact with the local multicast routing protocol entities through a well defined interface. Through this interface, data related to the service can be collected (e.g. number of sent/lost packets, number of links per multicast node, ...). Each agent has a local view of the tree topology, maintains a link with the underlying agents (MNA or MEA) as well as a link upward towards the source MSA.

This model is more scalable than a full source driven polling approach. Moreover it enables source agents to build service level statistics which are specific to the service level management process in use for the service (e.g. checking the conformity of the delivered service to the agreed level).

4 AMAM : an active network-based support of the architecture

The previously presented architecture can be implemented in several ways. This can be done with feature and protocol extensions of specific multicast approaches or through a standard management framework like SNMP with dedicated Management Information Models and associated MIBs together with usage scenarios. We chose an alternative to the above solutions, namely to exploit the benefit of active network technology in terms of flexibility and extensibility, to host the various components of the architecture [21, 22, 3]. We use this technology to dynamically download and operate both edge and node agents (MEAs and MNAs), enabling a seamless evolution that follows the topology changes of the managed multicast tree. The availability of a dynamic code distribution facility and the presence of execution environments on all nodes enables rapid and online cost calculation strategies change, update of security components, activation of tests for fault management purpose or more generally to push management functions where they are needed.

The resulting implementation called AMAM (Active-based Management Architecture for IP Multicast) is illustrated in Figure 2. Within AMAM, multicast management functions are designed as a set of dedicated active applications called plug-ins (accounting plug-in, security plug-in, test sender plug-in, ...). Plug-ins are stored in a repository within the source agent context. Once installed on the source agent,

these components can be downloaded by either MEA or MNA agents where they are executed. Plug-ins uninstall themselves when they are no longer in use.

The FLAME active network framework is the execution environment we used to build AMAM. This execution environment enables both dynamic installation and removal of active applications but also APIs and associated libraries enabling the execution environment to dynamically extend the interface it offers to active applications (e.g. providing a new packet capture service). All AMAM plug-gins are defined as FLAME applications.

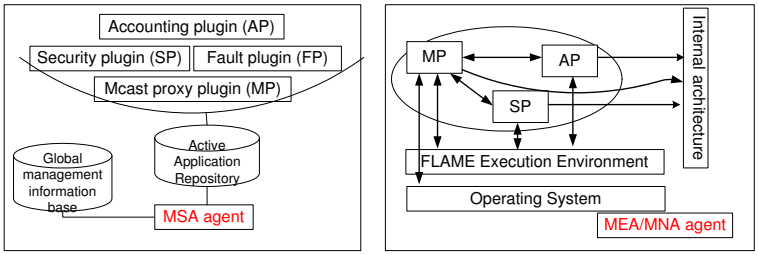

Figure 2. AMAM architecture

The repository contains by default four plug-ins, each of them having its own internal architecture. These plug-ins are :

- the **Mcast proxy plug-in (MP)**. The role of this plug-in is to provide transparency to the underlying multicast technology used. The plug-in interacts with the routing as well as with the membership management protocols through specific interfaces to collect information concerning members and delivered traffic. Providing such a protocol independence facilitates deployment and integration into legacy approaches. This plug-in is loaded in all agents.

  In the edge agents, MP builds a multicast table that contains information for each member on this edge. This generic table is built using proxy-lets which provide the link with the two currently supported group membership protocols namely IGMP for IPv4 and MLD for IPv6 (see Figure 3).

  In the node agents (MNAs), proxy-lets that communicate with the routing protocol are used. Other plug-ins use these proxy-lets and the data they collected to perform their management task (see Figure 2);

- the **Accounting plug-in (AP)** specialised in accounting tasks. It owns an interface with the local MP, with external APs deployed in the other nodes as well as with the group members that joined a session. Using the routing proxy-lets, this agent collects information related to the multicast traffic and feeds the local database. Based on this information and the member table built by the MP agent, this plug-in allocates the cost and computes the amount that will be assigned to each member. This information can then be used directly in the charging process and published directly towards the users through the *Client publishing* interface. The agent also maintains a copy of charging units in the local database (see Figure 3).

- the ***Security plug-in (SP)*** executes the security functions. Combined with a policy server which can be located at any place in the network, even in the

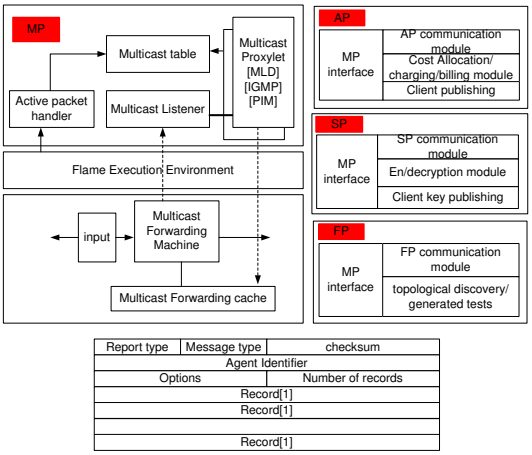

Figure 3. Internal architecture of the plug-ins

multicast source domain, it ensures access control and user authentication. The SP generates local keys and ensures their distribution to all connected members through its *key publishing* interface. This plug-in also executes the algorithms that encrypt/decrypt the multicast traffic that needs to be delivered (see figure 3).

- the ***Fault plug-in (FP)*** notifies the source of changes in the topology (i.e. add/remove of a branch in the multicast tree) that were recorded locally by either MNAs or MEAs. Based on the knowledge acquired through the reception of these notifications, the source agent (MSA) builds a topological view of the distribution tree. The plug-in configures itself through the source agent to generate test cases based on the information model defined in [18]. The routing proxy-let is also used through the MP interface to measure link quality over the distribution tree.

In addition to the plug-ins, a management data exchange protocol has been defined to enable communication among plug-ins either in different entities or in the same system. The protocol defines a generic data format (see Figure 3). This message format can be specialised for each plug-in. For example, the AP plugin defines a message to transport the data necessary to allocate the costs and another to transport the cost vector to be allocated. Each message contains the following elements :

- report plug-in type : an octet that specifies the type of the plug-in that sends the report. For example, if the *report type* of an MP agent is set to 1, then all messages received by FLAME whose type is 1 are forwarded to the local MP plug-in;
- message type : one octet describing the message type. Each plug-in defines its own message name-space;
- checksum : checksum over the entire packet. It is checked at each node which processes the message;

- agent identifier : a two octet identifier uniquely identifying the agent that initiated the message;
- options : used to specify a set of options required by plug-ins;
- number of records : number of data records contained in the message;
- record : contains the data which is specific to the message type.

The sequencing of messages is defined by each plugin.

5 Using AMAM for cost allocation and accounting management

Cost allocation is done through the application of a set of strategies which enable the assignment of a given cost to every participant in the downstream. In the multicast case, gain can be obtained by sharing costs. For example, the ELSD (Equal Link Split Downstream) strategy divides equally the cost of a link to all participants behind the link. The cost is computed over several parameters like the volume of traffic, the used bandwidth, congestion state of the link, ... To this transport cost, one has to add the content cost. The ELSD strategy has been implemented in [12] but without taking into account the dynamic nature of multicast trees.

Our architecture enables the support for an extended ELSD approach that we propose. This extension, called D-ELSD explicitly considers the dynamics of a multicast tree as a fundamental parameter of the strategy as opposed to ELSD.

Lets consider following definitions:

- An allocation session allocates costs among leaving participants. This session is started each time an edge node looses its last participant. The node that initiates the allocation session is represented as $init$ in the remainder;
- Δ_t denotes the duration between the arrival of the first member on an MEA and the departure of the last member from the same MEA. Over this period, the MEA records all arrivals and departures of its local members;
- $\Delta_t = \sum_{m=0}^{k} \delta_m$ and $N_{loc}(i) = Vect_{m=0}^{k}[n_m]$ where n_m is the number of active members in δ_m at node i;
- for each intermediate node, the MNA agent maintains the evolution of the number of branches towards downstream nodes. Let $\Delta_t = \sum_{h=0}^{r} \theta_h$ and $B_{loc}(i) = Vect_{h=0}^{r}[b_h]$ such that b_h is the number of branches in θ_h[11];
- let $\Delta_t = \sum_{j=0}^{p} \tau_j$ and $N_{downstream}(i) = Vect_{j=0}^{p}[n_j]$ such that n_j is the number of members downstream of node i within τ_j;
- let $N_{merge}(i)$ be the resulting vector of the following classification (δ_m, τ_j) from $N_{downsteam}(i)$ out of every downstream branch and N_{loc} over Δ_t for node i. Thus, if we have only one branch downstream : $N_{merge}(i) = Vect_{s=0}^{s=k+p}[n_s]$ over $\Delta_t = \sum_{s=0}^{k+p} \alpha_s$ such that $n_s = n_i + nj$ for $\alpha_s = \min(\delta_m, \tau_j)$. Thus, we also have $N_{merge}^{1/N}(i)$: the notation of the vector whose elements are $1/n_s$;

[11] Note that one MEA can serve its members locally and have downstream branches to other MEAs or MNAs. Thus, an MEA can maintain both vectors of numbers for members and branches.

- let $chem(i,j)$ be the path from a node i to a downstream node j belonging to the multicast tree and finally, let $cost(i,i+1)$ be the cost of a link between a node i and its first downstream node $chem(i,j)$ towards node j;

D-ELSD cost allocation is done in two phases : (1) a preparation phase during which the data necessary for the application of a strategy is collected. The necessary data is the number of members downstream (in the $chem(source,init)$ path)(2) a processing phase where the cost for all members which receive their traffic through a path that crosses $chem(source,init)$. Thus :

- for the duration Δ_t, the node that initiates the cost allocation session (node i), builds his $N_{merge}(i)$ vector. Once built, this vector includes the entire dynamics of the downstream multicast subtree. The initiating node sends this vector to the first upstream node $(i-1)$ on the path to the source ($chem(source,i)$).

  This vector will then be used as the $N_{downstream}(i-1)$ vector. Node $i-1$ then builds its fusion vector and sends it upwards. This process continues until all vectors have reached the source;

- once the MSA received the vector $Vect[n_i]_{i=0}^{i=l}$ for the duration $\Delta_t = \sum_{i=0}^{l} \beta_i$, it builds the following allocation vector :

$$Vect_{toSend}(source) = cost(source, downstream\_node) * [N_{merge}^{1/N}(source)] \tag{5.1}$$

  and sends it to the first node downstream. l represents all join/leave events that have been registered for the Δ_t duration for the subtree containing *chem-(source,init)*;

- every node i in $chem(source,init)$ that did receive an allocation vector from the upstream node, builds the cost vector for its members according to the following equation :

$$Vect_{cost}(i) = Vect_{toSend}(i-1) * I(l,l) * (M(l,k) * N_{loc}) \tag{5.2}$$

  where $I(l,l)$ is the identity matrix of size l and $M(l,k)$ the transformation matrix of the vector N_{loc} from size k to a size l vector.

$$M(l,k) = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 1 & 0 & \dots & \vdots \\ 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \vdots & 1 & \vdots & 1 \\ \vdots & 0 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix} \tag{5.3}$$

  where the number of 1's per column is the number of δ_l in δ_k

- Node i sends a notification to its upstream nodes that it assigns the following cost vector :

$$Vect_{toSend}(i) = Vect_{toSend}(i-1) + cost(i, i+1) * N^{1/N}_{merge}(i) \qquad (5.4)$$

  Every node on the path $chem(source, init)$ does the same processing until the vector reaches the node that initiated the session.

As soon as the cost allocation vector has been received (see equation 5.2), each MEA agent that is in the subtree on the $chem(source, init)$ path ensures allocation for all its local members. This is done, based on the detailed knowledge of the behaviour of each member, behaviour that has been recorded in the local database. A member m_i being connected during a $\subset \Delta_t$ period of time will get a cost equal to the sum of elements from $Vect_{cost}$ for this duration. These costs can be communicated to the members which can thus estimate their cost in near real time.

Note that the link cost is computed while respecting the strategies of load computation. Load computation can rely on several parameters like the volume of received data, the duration of the connection to the service, the used bandwidth, . . .). The cost vector depends only on the cost allocation strategy. In our framework, we used an extended version of ELSD but other strategies can be implemented as well.

6 Implementation issues

In this section we will discuss some implementation issues in the AMAM architecture. Figure 4 represents the deployment senario of the management agents. We assume that in each edge router, a MEA is installed and that the MP plugin is downloaded by default. The MP plugin serves to interact with the multicast membership and routing protocols as mentionned before. When MP detects the arrival of the first member joining the multicast session, the MEA initiates the deployment of the different MNAs in all the routers which are in the route to the first MNA/MEA belonging to the multicast distribution tree. These MNAs take their proper deployment configuration from a policy sever provided by the FLAME environment. Once installed, the MNAs records the routing information that will be used in the accounting process (like number of branches. . .). The MEA records in its turn all the arrival and departure of its local members. When The MEA detects the departure of the last member, it initiates a cost allocation session according to the D-ELSD strategy. The source computes the cost allocation vector and sends it to the concerned downstream routers. Each MEA, based on this cost allocation vector and its local information computes the charge of its local members and sends them a real time invoice. Finally, before desinstalling itself, the MNA notifies the policy server. This interaction with the policy server is also used to construct and update the topologic view of the multicast tree distribution at the source. The MEA, before desinstalling its downloaded plugins (except MP plugin), sends the local management data to the source for a final storage purposes (see Figure 4).

Note that D-ELSD is computed only when a topology change in the multicast distribution tree occured and not as long as there is a join/leave of members. Consquently, the scalability of management functions in the core is the same as the one of the multicast routing protocol since these management nodes only communicate and process data on a tree topology change only.

The open interfaces with the membership and routing protocols can be obtained by extending the major open source IGMP, PIM-SM and MLDv2 implementations.

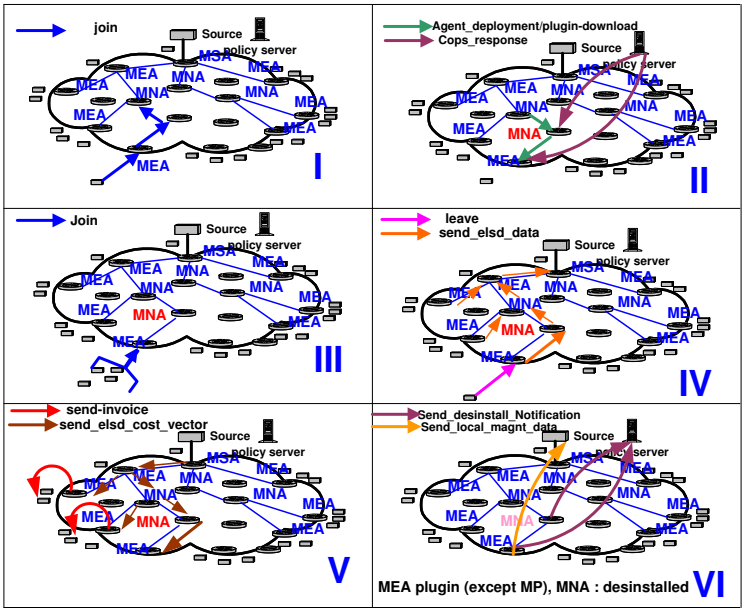

Figure 4. AMAM Agents deployment scenario

In our research group, we have extended the IGMPv3 interface for some accounting purposes. For instance, the connection duration for each member is computed with some timer variable used by the IGMP proxylet to fixe the time of arrival and departure of each member.

To evaluate how the performance could be affected when extra functions are deployed in the agents, FLAME provides some integrated functionnalities for perfermance evaluation that we will use for our real implementation of AMAM. To improve the scalability gain in the case of large scale of group members a simulation work will be done for more performance analysis.

7 Conclusion and future work

In this paper, the need for developing management architectures dedicated to multicast services was addressed. The dynamics of these services, the very specific requirements towards the standard management functions especially the needs for scalability and ease of deployment and maintenance have been identified as the major points, a management solution for multicast services should master.

Based on these requirements, a management architecture was proposed. This architecture is based on a three level hierarchy over which both management data and functions are distributed. It enables the fusion of both signalling and management planes for several multicast management functions like security and accounting. The FLAME execution environment has been selected to host the architecture. The resulting environment is called AMAM. Within AMAM, management functions are defined

in terms of active applications which can be downloaded on demand and which use a dedicated protocol to exchange management messages. Integration with the control planes (membership management and routing) is done through proxy-lets. New management functions can be dynamically added.

Two management functions and their implementation within AMAM have been presented. Through the proposal of an extended ELSD strategy, we have shown that the architecture can support accounting functions that support the dynamics of multicast group members. Other functions can be implemented in the framework in a similar way. For example, in the security management process and especially in the key management function, AMAM implements efficiently the solution based on the principle of a single global key per source and several local keys managed by the edge routers. The source agent generates one global key and ensures its distribution to all nodes of the multicast tree. This key is used to encrypt the traffic that will be delivered over the secured group. Each edge agent decrypts the traffic with the global key and generates a local key, ensures its distribution to its local members and encrypts the traffic with this local key. Local update of cryptographic keys makes the solution more scalable avoiding a complete update over the entire tree each time a member joins or leaves. Furthermore, the architecture can deploy approaches like MRM through configuration of MNAs and MEAs. Session control can be done based on the information model proposed in [18]. Another multicast monitoring service was also implemented in this architecture. This is the Hierarchical Passive Multicast Monitoring (HPMM) framework [25] which does fault correlation over the multicast tree.

Pushing forward the implementation of the architecture within the FLAME active network constitutes our first goal in the near future. This mainly requires some refinement in the specification of the management plug-ins. A second goal is to complete the study of the behaviour of the framework in the context of SSM multicast services. Finally, the architecture need to be extended to be deployed in a multi-domain (multi-ISP) environment.

References

[1] E. Al-Shaer and Y. Tang. Smrm: Snmp-based multicast reachability monitoring. Proc. NOMS'2002, p. 467-482, 8th IEEE/IFIP Network Operations and Management Symposium : Management Solutions for the New Communications World, R. Stadler and M. Ulema, Editors, ISBN 0-7803-7382-0, April 2002.

[2] K. Almeroth, L. Wei, and D. Frainacci. Multicast reachability monitor (mrm), April 1999.

[3] L. Andrey, I. Chrisment, O. Festor, and E. Fleury. *Les réseaux multi-média*, chapter Les réseaux actifs. collection IC2 chez Hermès, 2000.

[4] A. Ballardie. Scalable multicast key distribution, May 1996. IETF RFC 1949, Experimental.

[5] B. Cain, S. Deering, B. Fenner, I. Kouvelas, and A. Thyagarajan. Internet group management protocol, version 3, March 2001. RFC 1075.

[6] P. Calhoun, J. Arkko, E. Guttman, G. Zorn, and J. Loughney. Diameter base protocol, june 2002. <draft-ietf-aaa-diameter-11.txt>.

[7] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. Generic AAA Architecture, August 2000. RFC 2903.

[8] H.J. Einsiedler, P. Hurley, B. Stiller, and T. Braun. Charging multicast communications based on a tree metric, May 1999. 1st Workshop on Multicast Protokolle und Anwendungen, Braunschweig, Germany.

[9] B. Haberman and R. Worzella. Ip version 6 management information base for the multicast listener discovery protocol, January 2001. RFC 3019, Standards Track.

[10] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) architecture, July 1997. IETF RFC 2094, Experimental.

[11] S. Herzog, S. Shenker, and D. Estrin. Sharing the "cost" of multicast trees: an axiomatic analysis. *IEEE/ACM Transactions on Networking*, 5(6):847–860, 1997.

[12] Shai Herzog. *Accounting and Access Control for Multicast Distributions : Models and Mechanisms*. PhD thesis, USC, august 1996.

[13] H. Holbrook and B. Cain. Source-specific multicast for ip. november 2000.

[14] A. Kanwar, K. Almeroth, S. Bhattacharya, and M. Davy. Enabling end-user network monitoring via the multicast consolidated proxy monitor. In *SPIE ITCom Conference on Scalability and Traffic Control in IP Networks, Denver, Colorado, USA*, 2001.

[15] Jangwon Lee and Gustavo de Veciana. Resource and topology discovery for IP multicast using a fan-out decrement mechanism. In *INFOCOM*, pages 1627–1635, 2001.

[16] R. Oppliger and A. Albanese. Distributed registration and key distribution (dirk), May 1996.

[17] Bob Quinn and Kevin Almeroth. Ip multicast applications : Challenges and solutions, septembre 2001. RFC 3170, Informational.

[18] H. Sallay, R. State, and O. Festor. A distributed management platform for integrated multicast monitoring. Proc. NOMS'2002, p. 483-496, 8th IEEE/IFIP Network Operations and Management Symposium : Management Solutions for the New Communications World, R. Stadler and M. Ulema, Editors, ISBN 0-7803-7382-0, April 2002.

[19] K. Sarac and K. Almeroth. Supporting multicast deployment efforts: A survey of tools for multicast monitoring.

[20] K. Sarac and K. Almeroth. Scalable techniques for discovering multicast tree topology, 2001.

[21] J. Schoenwaelder. Emerging internet management technologies. IEEE IM'99 (Tutorial), October 1999.

[22] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, , and G.J. Minden. A survey of active network research. *IEEE Communications Magazine, Vol. 35, No. 1, pp80-86*, January 1997.

[23] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. AAA Authorization Framework, August 2000. RFC 2904 Informational.

[24] D. Wallner, E. Harder, and R. Agee. Key management for multicast : Issues and architecture. Internet RFC 2627, june 1999.

[25] J. Walz. Multicast Monitoring - Current Usage and a New Hierarchical Pr otocol . Master's thesis, Dept. of Computer Science, University of Massachusetts, February 2001.

[26] Hugh W.Holbrook and David R. Cheriton. Ip multicast channels : Express support for large-scale single-source applications. 29(4), october 1999.