A management architecture for active networks

A Management Architecture for Active Networks

A. Barone, P. Chirco
Dipartimento di Ingegneria Informatica
DINFO - Università degli Studi di Palermo
viale delle Scienze, 90128 Palermo, Italy

G. Di Fatta, G. Lo Re
CERE - CEntro di studio sulle Reti di Elaboratori
C.N.R. - Consiglio Nazionale delle Ricerche
viale delle Scienze, 90128 Palermo, Italy
{difatta, lore}@cere.pa.cnr.it

Abstract

In this paper we present an architecture for network and applications management, which is based on the Active Networks paradigm and shows the advantages of network programmability. The stimulus to develop this architecture arises from an actual need to manage a cluster of active nodes, where it is often required to redeploy network assets and modify nodes connectivity. In our architecture, a remote front-end of the managing entity allows the operator to design new network topologies, to check the status of the nodes and to configure them. Moreover, the proposed framework allows to explore an active network, to monitor the active applications, to query each node and to install programmable traps. In order to take advantage of the Active Networks technology, we introduce active SNMP-like MIBs and agents, which are dynamic and programmable. The programmable management agents make tracing distributed applications a feasible task. We propose a general framework that can interoperate with any active execution environment. In this framework, both the manager and the monitor front-ends communicate with an active node (the Active Network Access Point) through the XML language. A gateway service performs the translation of the queries from XML to an active packet language and injects the code in the network. We demonstrate the implementation of an active network gateway for PLAN (Packet Language for Active Networks) in a forty active nodes testbed. Finally, we discuss an application of the active management architecture to detect the causes of network failures by tracing network events in time.

1. Introduction

Traditionally, telecommunication networks have separated the management level from the communication level; different infrastructures exist for the delivery of the user data and the delivery of the information and commands exchanged between managing entities and managed objects. This is not the case with computer networks and in particular with todays Internet, where management and transport services share the communication infrastructure. In the Internet management architecture there are three main components: a managing entity, typically a centralized application which requires human intervention, the managed devices, such as network equipments and their software, and the Simple Network Management Protocol (SNMP) [18]. SNMP is the most widely used and deployed network management framework and it has been designed to be independent from vendor-specific products or networks. SNMP is an inherently centralized framework. An agent program embedded in a network device collects traffic statistics and stores configuration values in a Management Information Base (MIB). The managing entity can obtain the status information or modify the configuration of the device by polling the agent. Managing agents have a limited active role, i.e. they can only send an asynchronous trap message to the managing entity when few particular events occur, such as bootstrap and link failure. In the SNMP framework management agents are not able to manage faults locally or to coordinate other managing agents. The high number of network nodes in an Internet administrative domain suggests the adoption of a distributed approach to provide scalability. Moreover, a customizable management, i.e. the right control in the opportune place, can provide a better efficiency. Several distributed and programmable approaches to network management, which are based on paradigms such as Mobile Agents and Active Networks, have been proposed. The authors in [16] propose the Smart Packets architecture, which allows the remote execution of small code portions inside network objects to monitor and manage classical MIB objects. Smart Packets architecture, however, does not allow a dynamic addition of new monitorable objects on the nodes. The authors in [10] have an analogous goal even though they adopt the mobile agent technology as underlying layer. Recently, the authors in [1] proposed a general framework for the management of distributed mul-

timedia systems. This framework adopts hierarchical organization of agents that can be dynamically instantiated by means of customizable filters that are spread appropriately among all the network agents. For this reason, a heavy control activity is required to coordinate management actions for the realization of the required activities. Although Active Networks are a promising way to adopt a programmable agent approach in computer network management, they envision a new scenario and pose new problems.

While current data networks are static and impose the 'a priori' definition of the entities that have to be monitored, Active Networks involve a dynamic evolution of the network software. New network services and applications can be easily deployed in the network and skip the slow standardization process. In this case, network management has to deal with a highly dynamic environment, without the possibility of a previous standard definition of the monitorable data. Therefore, the current SNMP protocols do not appear suitable for the new scenarios; new management frameworks must be designed to deal with dynamic active environments. In this paper we propose an active networks management architecture, which adopts a distributed programmable agents strategy and a dynamic active MIB definition, in order to overcome the limitation of the SNMP framework and to enable the development and deployment of new management services.

The remainder of the paper is structured as follows. Section 2 shows the potential benefits of active networks in network management. Section 3 presents the active networks management framework. In section 4 the application and service implementation and the basic gateway services are illustrated, and in section 5 a network event mining application is proposed. Finally, some concluding remarks are made in section 6.

2. Potential Benefits of Active Networks in Network Management

The programmability introduced by the Active Networks may represent the leap towards a pervasive network management. The use of network services is widespread in todays society. The importance of services requires that they are guaranteed by means of a continuous monitoring for immediate interventions in case of fault. The concept of Resilience in Communications Networks has become an indispensable feature of the network architecture design. Resilience is commonly defined as the set of mechanisms that are able to cope with network failures. In the field of network management, this is well known as proactive management. Proactive management operates on the ground of symptoms that predict negative events in order to avoid them.

In this scenario Active Networks may be a suitable instrument for the implementation of a complete management system. The actual benefits that management applications may obtain from the Active Networks adoption fall into the following categories:

- availability of information held by intermediate nodes;
- data processing capability along the path;
- adoption of distributed and autonomous strategies into the network.

The above features completely answer the network management requirements. Mobile agents can be encapsulated and transported in the active code of application capsules. They can retrieve and extract pieces of information held by intermediate nodes in a more effective way than through remote queries from the application itself. For instance, an agent could make use of an active code to look-up the MIB objects of an intermediate node and select some entries according to a given criterion. It can either send such extracted information back to the application, or it can use the information to take timely decisions autonomously from the application. More examples can be found in other network management issues, such as congestion control, error management, or traffic monitoring. A meaningful example is the customization of the routing function. A mobile agent could be devoted to the evaluation of the path for the application's data flow, according to the user's QoS specification. Each application could set up its own control policy or exploit a common service (the default per-hop forward function). Active networks applications can easily implement distributed strategies by spreading management mobile agents in the network. The introduction of network node programmability makes the network system one single knowledge base, which is also capable of producing new information. This happens, for instance, when new actions are generated deductively from the resolution of previously stored data with occurrences of particular events, thus allowing the inference of new events and the triggering of codified measures.

3. Active Networks Management Framework

In this section we present an architecture for Active Networks management and monitoring. Active Networks introduce programmability in the network; new software components can be dynamically injected in the network to be executed in intermediate nodes. Network management becomes even more important than in traditional networks and it can accomplish more complex tasks. Traditional network management framework and tools are aimed at the management of network devices and their software, which are not meant to change frequently. In the traditional SNMP framework it would be very difficult, if not even impossible, to

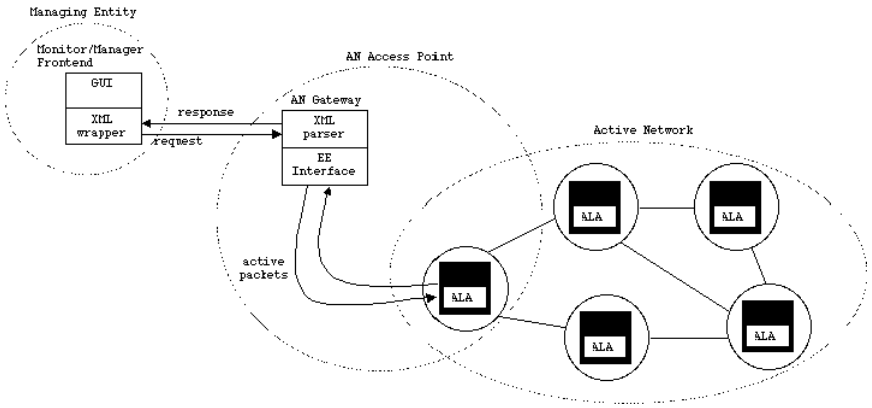

Figure 1. *Active Network Management Framework*

trace, debug, and monitor distributed applications dynamically deployed into the network. In the proposed architecture we introduce programmability features in the Management Information Base (MIB) and in the local agents to make the management application itself distributed, cooperative, and adaptive. Furthermore, we point out that the aim of an Active Networks Management framework has to take the management of active applications into account. Active applications management includes the deployment, integration, and coordination of the software components to monitor, test, poll, configure, analyze, evaluate, and control distributed network applications and their network resources. In the framework (figure 1) the Managing Entity (ME) operates by means of a graphical user interface. It sends queries and receives replies in Extensible Markup Language (XML) to and from an AN Access Point. An AN Access Point is an active node hosting a Gateway service. A Gateway service has two tasks: the translation of XML requests to the specific language adopted by the Execution Environment (EE) and the injection in the network of the appropriate active capsules that perform ME requests. This way, the Gateway acts as an interface for a particular active network and it is specific for the language supported by its EE. Several nodes in the active network can be configured to provide Gateway services. As illustrated in figure 1, a resident management service, the Active Local Agent (ALA), is installed in each active node. ME and ALA are the end points of the management communication. Namely, the ME can either query the Active Local Agents (polling) or deploy subtasks to them (programmable trapping). Local agents asynchronously perform subtasks in terms of Actions to be executed at local Events occurrences, as we will explain in the following. The architecture can be shared by different AN implementations and can provide a common management framework for different EEs. We have adopted XML to define a set of requests/replies for basic operational tasks, which are common to any Active Network environment. The Gateway basic services we have currently implemented are presented in 4.1. However, the set of Gateway services can be gradually enlarged. Once a new service is developed and tested, it can be provided for public use. Each basic service requires an EE-specific code fragment stored at the Gateway. This way, Gateway nodes provide transparent access to different Active Networks. For instance, the ME can use Gateway services to discover the network topology, explore the network nodes, find out which active applications are running and monitor their activities. Furthermore, the ME may include EE-specific code in XML requests to implement customized management services. A specific tag is provided to wrap user code in XML requests. User code causes safety and security problems that can be managed by means of authentication and authorization policy and limitation in the programming language. We are particularly interested in managing an Active Networks testbed where unusual management tasks are also required, such as defining the network topology to be deployed. For this aim the GUI provides a topology editor and the Gateway is able to manage the configuration and bootstrap phases of EEs in the active nodes.

3.1 Active MIB and Active Local Agent

In the design of our architecture we moved away from the traditional SNMP framework and introduced the advantages of programmability of the Active Networks paradigm. We redefined the role of the management agent and adopted a different model for the MIB. In the traditional SNMP framework a managed device is a network equipment

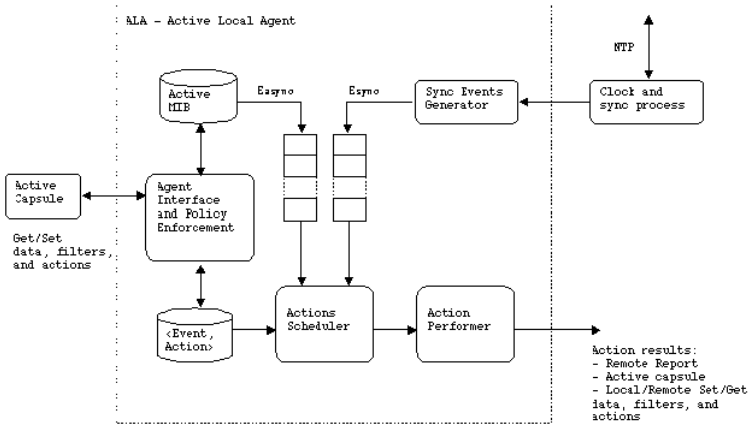

Figure 2. *Active Local Agent (ALA)*

which, in general, may contain several monitorable objects, either hardware or software (for instance network interface cards and routing protocols). In this protocol the Managing Entity can request the local SNMP agents basic operations on the MIB, i.e. 'Set' and 'Get' the value of the objects. Different from the SNMP scheme, in our model the ME can program the local agents behavior to accomplish independent tasks. Consequently, it becomes able to deploy distributed strategies. While the local SNMP agents have only preconfigured capabilities, the Active Local Agent is programmable. A degree of programmability is also added to the MIB objects with the adoption of the object oriented programming paradigm. In our framework, the managed objects are active applications, which are distributed applications whose software components run in both end nodes and intermediate nodes of the network. An application can subscribe the management/monitoring service in a node by registering a unique ID to the local agent. Once registered, it can store information into the local Active MIB (AMIB). An AMIB object is related to a component of the managed application in the active node. The AMIB object structure allows applications to store data and, eventually, a related code. Each object is not just a single variable storing a value, but it can also contain some code represented as a set of conditions called $filters$. In the ALA we implemented an Events-Actions model, as shown in figure 2. AMIB Filters are $< Test, Event >$ pairs, where the test is a Boolean expression built over basic predicates by means of logical operators (AND and OR). A filter test verification indicates that the given $Event$ has occurred. The filter test is executed on the data when a primitive *Set_data* call occurs, i.e. whenever the data change. If the test succeeds, the asynchronous event associated with the filter is raised and submitted to the Actions Scheduler. It should be noticed that, while the primitive *Set_data* events are independent from the application, the events produced by the filters follow the application semantic. Further events are generated by a synchronization process connected to the system clock. The system clock is synchronized with a global time from a reference time source by means of the Network Time Protocol. NTP provides accuracies typically within a millisecond in LANs and up to a few tens of milliseconds on WANs [13]. The local agent manages a list of $< Event, Action >$ pairs, which are the scheduled actions for the synchronous and asynchronous events. Applications can set actions to be executed by the ALA in correspondence to given events. Incoming active capsules can access the data and code in the AMIB and in the Action list through the services provided by the Agent Interface and Policy Enforcement module (Set/Get data, filters, actions). In general, actions are in the form of active capsules, which are released to perform the required task. For instance, as a consequence of a queue overflow event, a report trap can be simply sent back to the ME through a capsule containing the warning message, or a proactive action could be taken by means of a more complex code in the capsule. In general, AMIB filters can be injected by either the application itself (the data owner) or by a different application, e.g. the managing entity. In order to limit the effects of malicious or malfunctioning codes, filters cannot access network primitives nor make recursive calls and Actions cannot trigger local Events. The restriction on filters and actions can be tuned by a proper security policy based on authentication and authorization. In a public network infrastructure a security policy in the management of AMIB Data, filters and actions has to be enforced by the ALA. However, this is beyond the scope of this paper. The

management framework is now considered active because, first, the MIBs contain both data and code, and secondly, each local management agent (ALA) plays an active role in the network management by independently performing programmed tasks.

4. Application and Service Implementation

The framework described in this paper originates from a practical need of managing an active network. We have realized an experimental testbed of forty active nodes with the aim to investigate the novel paradigm of active networking. The testbed is currently used for the development and testing of innovative protocols and applications: [3], [8], [6], [7]. The experimental laboratory is constituted by a fully connected network of active nodes, which allows us to clip specific topologies according to the research needs. Each node is implemented by a Linux workstation equipped with four 100 Mbps Ethernet network cards to emulate a four ports active router. The active nodes are equipped with the main software packages for active networking such as PLAN [12], ANTS [22], ASP [14]. The management framework described in the previous section allows monitoring and management of the network services and applications. Moreover, the management framework assumes a relevant role in the development phase of network applications. A network application is fully responsible for selecting its own internal data that may be observed by other applications (e.g. the ME). The application can decide which data in the active MIB can be monitored. Variable monitoring is very useful during the debugging activity. Furthermore, the active MIBs allow the realization of a communication system between network applications when AMIB objects are used as shared memory variables. Inter-applications communication makes new network services possible: applications can take advantage from data collected by other applications. Several simple functionalities of the active network Gateway have been developed and tested as active services in the PLAN execution environment (see 4.1). For instance, we developed a service, called $Delivery\_MST$, in order to optimally visit all the nodes of the network. Given a task, the service delivers and executes the task in each node. The service builds a temporary minimum spanning tree along which the results will be progressively merged and finally delivered to the source. The service is optimal in the bandwidth consumption and it avoids messages implosion at the source. Navigation patterns are strategies to move from node to node in the network as part of program execution. Simple navigation patterns are described in [9]; the $Delivery\_MST$ service can be considered an advanced navigation pattern. Furthermore, synchronous events allow an easy and simple implementation of a 'snapshot' service. By means of this service it is possible to capture all the values assumed by the investigated AMIB in a precise temporal instant, in all the network nodes. Other services we implemented include topology discovery, monitoring the path of active packets, monitoring the routing tables, etc.. External applications can use the basic Gateway Services to obtain information on the active services available at a node and finding fault situations.

4.1 Basic Gateway Services

An active network can have many Access Points which provide the Gateway services. From any Access Point it is possible to query active nodes. Gateway services are relative to either the network or application levels, and fall into two categories: public and administrative services. The public services do not modify the active network configuration and their multiple and concurrent executions are allowed. On the contrary, in order to avoid conflicts, a mutual exclusion technique is adopted to guarantee that only one administrator has been authenticated and is executing administrative operations. In the following we briefly present the services we have implemented in our testbed within the PLAN EE.

$Active\ Network\ Level - Public\ Services$

Get_Topology
This service retrieves the topology of the current active network and information on the status of active nodes.

Ping
This service uses the UNIX ping command to test if a node is reachable at the network layer.

Active_Ping
This active version of the ping command uses active messages to test if a node is reachable at the active network layer, i.e. at the Execution Environment layer.

Get_Routing_Table
This service provides the routing table of an active node.

$Active\ Network\ Level - Administrative\ Services$

Set_Topology
In experimental test-bed this service defines the topology of the active network. It generates and deploys the configuration required for each active node.

Bootstrap_AN
This service spawns the EE processes in all active nodes. The service is available only if a topology configuration has been deployed by the Set_Topology service.

Shutdown_AN
This service stops the EE processes of all the active nodes.

Set_Static_Route
This service sets a static route between a source and a destination.

Delivery_Path
This service generates an active capsule, which visits the active node of a given source-destination path to execute a given task. The task is a code fragment, which is provided by the user in the language supported by the EE.
Delivery_Path&Return
As Delivery_Path, but in this case the path is traversed in the two directions, forwards and backwards. Each node is visited twice. In the backwards path results will be progressively merged and finally delivered to the source of the request.
Delivery_MST
This service generates an active capsule, which optimally visits the active network to execute a given task in each active node. The service builds a temporary minimum spanning tree along which the results will be progressively merged and finally delivered to the source of the request.
Delivery_Tree
As Delivery_MST, but in this case the nodes to be visited are a subset of the network nodes.

Active Application Level – Public Services
Get_Applications_List
This service retrieves the list of the active applications which have stored data in the AMIB of a given active node.
Get_Data_List
This service retrieves the list of the AMIB object IDs of a given application in an active node.
Get_Value
This service retrieves the current value of a particular AMIB data of a given application in an active node.
Get_Filters_List
This service retrieves the active filters associated to a particular AMIB data of a given application in an active node.
Get_Events_List
This service retrieves the Events of a given application in an active node. Application Events are defined in filters and actions.
Get_Actions_List
This service retrieves the Actions list of a given application in an active node.
Get_Action_Code
This service retrieves the Action code fragment of a given application in an active node.

Active Application Level – Administrative Services
Set_Value
This service sets the value of a particular AMIB object of a given application in an active node.
Set_Filters
This service modifies the active filters associated to a particular AMIB data of a given application in an active node. Setting an empty list of filters is used in order to remove filters.
Set/Remove_Action
This service modifies the active actions of a given application in an active node.

It should be noticed that an active application can always set and remove its own data, filters and actions through the ALA services of an active node. This process does not involve the above Gateway services, which are provided to managing/monitoring entities. Finally, the XML message prototypes of the request and response of the 'Get_Topology' service are given as example.

$$< request\ command = \text{“}get\_topology\text{”} >$$
$$< /request >$$

$$< response\ command = \text{“}get\_topology\text{”} >$$
$$\quad < status >< /status >$$
$$\quad < net\ nodenum = \text{“ ”} maxdegree = \text{“ ”} >$$
$$\quad\quad < node\ name = \text{“ ”} degree = \text{“ ”} >$$
$$\quad\quad\quad < coord\ x = \text{“ ”} y = \text{“ ”} >< /coord >$$
$$\quad\quad\quad < adjac\ name = \text{“ ”} cost = \text{“ ”} >< /adiac >$$
$$\quad\quad\quad < addr >< /addr >$$
$$\quad\quad\quad < port >< /port >$$
$$\quad\quad < /node >$$
$$\quad < /net >$$
$$< /response >$$

In the next section we discuss an application of the active management architecture to detect the causes of network failures by tracing in time network events.

5. Network Events Mining

An application, which can take advantage of the architecture proposed in this paper, is related to the knowledge management of network events. For different management aims, it is often necessary to acquire information from the 'log' files of many network devices. However, it may result extremely expensive to store all the events that occur in all the network devices in a long period of time. It would be more efficient to be able to distinguish which events must be stored and which do not add further knowledge to the network history. Network Events Mining (NEM) deals with large archives of events obtained from the network systems, and extracts useful information from these data. Goals of NEM systems are:

- to diagnose root causes of network faults and performance degradations by establishing relationships between network events;

- to filter event (alarm) flood by correlating several events into a single conceptual event.

Useful NEM systems should provide:

- correctness: the root causes inferred by NEM should be entailed by the detected events with a high likelihood, i.e. the root causes have really occurred in the network;
- optimality: the NEM should infer as small a set of root causes that can explain all the detected events.

A Managing Entity with NEM functionalities can adopt a dynamic model of the network and a logic engine to generate a distributed knowledge base, where only the essential information to describe the system history is stored. The word '$history$' here refers to a limited period in the past, of which the beginning is continuously advanced with the flowing of the current time. The knowledge of network events constitutes the necessary information to answer questions about the causes of network failures. Fault management is based on the definition of the normal operating conditions, and on the abnormalities detection. In general, alarms are generated in the network when abnormalities are detected. In alarm-based fault detection systems, a single fault will often cause a large number of alarms. Moreover, several faults may coexist causing a cascade of alarms. NEM is able to correlate alarms to pinpoint their root causes in order to efficiently handle them.

Currently, we are developing a management entity ME, which is able to trace the real causes of network failures back. The ME engine is based on the situation calculus and specifically on the Golog language [15]. Situation Calculus is a dialect of the first order logic and it allows the modeling of dynamic systems. This calculus captures the dynamism of a system, since it allows the definition of actions that move the system from a given state to another one. All changes to the world are the results of named actions. A possible world history is simply a sequence of actions, and it is represented by a first order term, called a 'situation'. While the Situation Calculus allows the representation of simple actions, Golog is a novel logic programming language, which allows the modeling of complex behaviors. Particular attention must be paid to the ontological aspects of the knowledge base. An effective structured representation of the network events and actions can greatly improve the logical deductive process. This Managing Entity tries to resolve the occurrence of a given event within its knowledge base and, when possible, deduces a conclusion. If no conclusion is deducible from the current knowledge base, the ME is able to trigger further investigations with the aim to acquire new information, until the uncertainty is solved. New information is inserted in the knowledge base only if it is not effectively deducible. In terms of the first order logic this means that it is not a logical consequence of the asserted facts. This way, the knowledge base is kept in its minimal form. Moreover, the situation calculus is suitable for the coding of proactive behaviors of the ME. In a logic programming language, it results easy to implement rules to recognize situations that are prologue of incipient problems, e.g. failures of the network or hacker attacks, and to codify the opportune contra measures. Both the capabilities of obtaining new information and of implementing recovery and safety policies rely on the programmability introduced by active networks. The programmability of the active nodes allows the customization of particular tasks to investigate unknown aspects of the network history, to perform proactive actions which avoid failures, and to resist to attacks. For the sake of simplicity we defined an event as an instantaneous occurrence at a time point. The time is assumed to be discretized. Each event is associated with the AMIB object where it occurs. AMIB Objects can be seen as tuples of attributes, and the events as a change in the attribute state. Events can be classified as primitive or composite. Primitive events are generated at the change of the object attributes, while composite events are non directly observable. For instance, a router failure can be inferred from a persistent connection failure. We consider two forms of event correlations as in [11]: the causal and the temporal correlations. The causal correlation expresses the relationship of causality, i.e. a cause-effect dependence. For instance, this kind of correlation can be adopted to establish the root causes of faults. The causal correlation can be easily expressed as a first order logic theory in which are assiomatized simple properties such as the reflexivity, transitivity and union. Furthermore, we can establish even stronger correlations between events based on their temporal occurrences. Some examples are relationships such as: $e1$ **followed_by** $e2$, **first** $e1$ **since_the_recent** $e2$. In [11] it is shown that the temporal correlation is more powerful than the causal one. The causal events correlation does not allow to cover all the possible events relationships, and the temporal correlation is necessary to express the dependency from persistent events. Current efforts are aimed to draw event correlations in order to infer composite events from the knowledge of previous events (either primitive or composite) for different management aims.

6. Summary and Conclusions

This paper presents an active network management framework which exploits network programmability to enable agents based distributed strategies. The proposed architecture is capable of managing both the active network Execution Environments and the Active Applications running on them. Its main feature consist in the extension of the classical MIB objects into more powerful entities where

user customizable code is associated to network data. The $Filters - Events - Actions$ framework improves management efficiency and enables new management services. Finally, a network events mining strategy is proposed to extract only the useful information from the network data for different management aims.

References

[1] Al Shaer E., "Active Management Framework for Distributed Multimedia Systems", Journal of Networks and Systems Management, vol. 8 n. 1, 2000, pp.49-72.

[2] Bhattacharjee, S., Calvert, K.L., Zegura, E.W.: "Active Networking and End-to-End Arguments", IEEE Network Special Issue on Active and Controllable Networks, vol. 12, n. 3, May-June 1998

[3] A. Chella, G. Di Fatta, G. Favarò, M.D. Guarino, G. Lo Re, "A Reconfigurable Neural Environment on Active Networks", Proc. of IEEE IJCNN2000, Como, Italy, July 2000

[4] Chen, T. M., "Evolution to the Programmable Internet", IEEE Communications Magazine, vol.38, n. 3, March 2000.

[5] L. Delgrossi, G. Di Fatta, D. Ferrari, G. Lo Re, "Interference and Communications among Active Network Applications", Lecture Notes in Computer Science vol. 1653, Springer-Verlag, pagg. 97-108, Proc. of IWAN'99, Berlin, Germany, 30 June - 2 July 1999.

[6] Di Fatta G., Gaglio S., Lo Re G., Ortolani M., "Adaptive Routing in Active Networks", IEEE Openarch 2000, Tel Aviv Israel 23-24 March 2000.

[7] G. Di Fatta, G. Lo Re, "Active Networks: an Evolution of the Internet", Proc. of AICA2001 - 39th Annual Conference, Cernobbio, Italy, 19-22 Sept. 2001.

[8] G. Di Fatta, S. Gaglio, G. Lo Re, M. Ortolani," Artificial Ants for Active Routing", IAS6, the 6th International Conference on Intelligent Autonomous Systems. Venice,July 2000.

[9] Kawamura R., Stadler R., "Active Distributed Management for IP Networks", IEEE Communicatiions Magazine, vol. 38, N 4. April 2000, pp. 114-121.

[10] Goldszmidt G., Yemini Y, "Delegated Agents for Network Management", IEEE Communicatiions Magazine, vol. 36, N. 3 March 1998, pp. 66-71.

[11] Hasan, M., Sugla, B., Viswanathan, R.: A Conceptual Framework for Network Management Event Correlation and Filtering Systems. Proc. of Sixth IFIP/IEEE International Symposyum on Integrated Management, May 1999

[12] Hick, M., et al, "PLAN: A Packet Language for Active Networks", Proc. of 3rd ACM SIGPLAN International Conference on Functional Programming, pages 86-93. ACM, September 1998.

[13] Mills D. L., "On the accuracy of Clocks Synchronized by the Network Time Protocol in the Internet System", ACM Computer Communication Review, vol. 20, no. 1, pp. 65-75, Jan. 1990.

[14] G. Phillips, B. Braden, J. Kann, and B. Lindell. "Writing an Active Application for the ASP Execution Environment" (Release 1.3)

[15] Reiter Raymond, "Knowledge in action: Logical Foundations for specifying and implementing Dynamical Systems" The MIT Press Cambridge Massachusetts 2001

[16] Schwartz B. Y., et al, "Smart Packets: applying Active Networks to Network Management", ACM Transaction on computer Systems, Vol. 18, N. 1, February 2000, pp 67-88.

[17] Smith, J. M., Calvert, K.L., Murphy, S. L., Orman, H. K., Peterson, L.L.: Activating Networks: A Progress Report". IEEE Computer, Vol. 32 N. 4, April 1999, 32 - 41

[18] Stallings W., "SNMP and SNMPv2: The Infrastructure for Network Management", IEEE Communicatiions Magazine, vol. 36, N. 3 March 1998, pp. 37-45.

[19] Tennenhouse, D. L., Smith, J.M., Sincoskie, W.D., Wetherall D.J., Minde, G.J.: "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, January 1997, 8

[20] Tennenhouse, D. L., Wetherall, D.J.: "Towards an Active Network Architecture", Computer Communication Review, Vol. 26, No. 2, April 1996

[21] Wetherall, D.J., Legedza, U., Guttag, J.: "Introducing New Internet Services: Why and How", IEEE Network Magazine Special Issue on Active and Programmable Networks, vol. 12, n.3, May-June 1998

[22] Wetherall, D.J., Guttag, J., Tennenhouse, D.L.: "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, CA, April 1998