An Environment for Developing Neural Network-based Intelligent Agents

Adriano Nascimento Miguel Franklin Mauro Oliveira

nascimen@hugo.int-evry.fr castro@hugo.int-evry.fr mauro@etfce.br

LAR - Multiinstitutional Laboratory of Computer Networks

CEFET/CE - Federal Center of Technological Education (Ceará)

LIA - Laboratory of Artificial Intelligence

UFC - Federal University of Ceará

Abstract

This work describes RENATA (Neural Networks Applied to the ATM Traffic Management), a tool that enables the development of intelligent agents to perform proactive management at an ATM network. The agents are based on Artificial Neural Networks, as their intelligent component. Due to their learning, generalization and adaptability capabilities, neural networks make possible anticipated detection of abnormal situations on an ATM network, by analyzing its status information. The developed agent provides information to the neural network, acts according to its outputs and monitors its behavior.

1 Introduction

Applications in computer network have demanded more bandwidth and more accurate requirements, such as minimum delay and low loss rates. The conventional technologies seem incapable of guarantee Quality of Service when they integrate at the same time services of voice, video and data. ATM (Asynchronous Transfer Mode) satisfies these requirements, so that it was chosen by the ITU-T to support the B-ISDN (Broadband Integrated Services Digital Networks).

Due to the complexity and flexibility inherent to ATM, an efficient management system is essential. Management solutions that were adequate to other technologies are insufficient to ATM. The diversity of services, the high-speed environment and the need for an integrated solution demand new requirements on the management system [1].

The management tools should anticipate possible problems before they happen and not only react. It is necessary to observe abnormal behavior in the network, collect its symptoms and diagnose a possible bigger problem. A proactive approach is therefore always desirable, specially in ATM environments. Techniques such as expert systems and neural networks have been applied for endowing management systems of knowledge making them able to behave proactively [2].

RENATA (Neural Networks Applied to the ATM Traffic Management) is a new proposal for the development of intelligent agents to perform proactive management at an ATM network.

The agents are based on Artificial Neural Networks. Due to their learning, generalization and adaptability capacities, neural networks can predict abnormal situations on an ATM network, by analyzing its status information. The RENATA agents will be able to act directly or just notify the network administrator about the problem.

The RENATA architecture treats since the acquirement of the necessary data to the neural network training, passing by its creation until the development of the agent. The agent uses the ATM management mechanisms (SNMP MIBs, ILMI and OAM cells) and interacts with the neural network in order to behave proactively. Besides the specification of the physical and functional architectures, a prototype was implemented integrating the necessary tools for the implementation of the agents.

This paper has the following organization. The section 2 presents RENATA, specifying its functional and physical architectures. The section 3 comments implementations details, describing the implemented tools and the developed agent. At last, section 4 presents some conclusions and suggestions for future works.

2 RENATA

Although SNMP became the *de facto* standard for network management, it has well-known limitations. The application of the intelligent agents technology can solve some of these problems, specially those concerning to scalability, complexity and cost [3]. If the intelligent agent presents proactive behavior, a greater optimization is possible. Among the techniques that enable this characteristic, the neural networks have been very researched. Also, various works apply neural networks on the resource management problem of ATM netwoks. For example, in [4] a neural-based controller is proposed for bandwidth allocation. Thus, the interest for neural networks applications in ATM proactive management has grown. Facilities for the development of intelligent agents based on neural networks must be created. In this context, RENATA was projected.

2.1 Functional Architecture

The figure 1 shows the Functional Architecture of RENATA, which has three modules: Training Module, Neural Module and Management Module. The architecture defines the functionality of each module and how they interact aiming the development of agents.

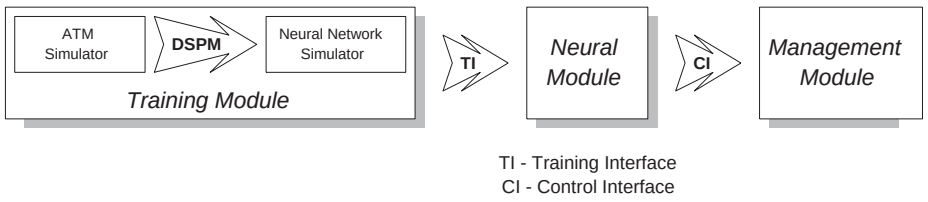

Figure 1: RENATA Functional Architecture

In the Training Module, the neural network is projected, trained and validated. To accomplish that, this module is divided in three: the ATM Simulator, the DSPM (Data Selection and Preparation Module) and the Neural Network Simulator. The Neural Module consists on

the neural network produced by the Training Module and information about its architecture and objective. The Management Module is responsible for the integration and activation of the neural network through the development of an agent that provides the input data to the neural network and, according to its output, takes the proper actions.

The neural network training is done off-line so it may have a better performance in a high-speed environment during the operational phase. Because of that, the Training Module is separated from the Management Module. Next, all the components of the architecture are described.

2.1.1 Training Module - ATM Simulator

Usually, neural networks need great quantity of data in order to learn. In telecommunications real situations, their applicability depends on the availability of test patterns that characterize normal and abnormal operations. If there is not historic records, it becomes hard to obtain training data since it is not possible to stop a production network only to test it with problems. So, the use of an ATM Simulator is justified by this difficulty in obtaining data and by the flexibility to simulate an ATM network under various situations.

From the definition of the problem that will be diagnosed by the neural network, simulations must be done to obtain the necessary data for the neural network learning. First, the topology of the ATM network is described: switches, links and terminals with their physical characteristics. From the topology, applications are simulated according to their traffic load and type. It's possible to simulate link failure or heavy load over a node to observe the behavior of the network and so obtain data that characterizes these situations.

According to the problem, certain log options in the ATM Simulator must be set so that it generates the data which will be used in the training of the neural network. These data should correspond to those which will be collected by an agent in a real environment, during the operational phase.

2.1.2 Training Module - DSPM

Before being submitted to the neural network, data should be selected, divided and treated. These functions are done by the DSPM (Data Selection and Preparation Module).

According to [5], approximately 98% of the data should come from normal operation and 2% should characterize the situation that the neural network must detect. After that, data should be divided between training, test and validation data.

After the selection, data should be treated so they can be submitted to the neural network. Data must be passed from the ATM simulator log format to the Neural Network Simulator format. Besides that, some neural network models only accept binary inputs, while others accept real inputs in the interval 0 to 1 or -1 to 1. In this case, techniques like normalization, scaling and coding are applied. For example, the number of active connections passing through a switch is an integer that should be scaled to real in the interval 0 to 1 to be accepted as a valid input in certain neural network models.

Aspects like the type of the training should be considered during the data preparation. For example, if the training is supervised, the data must also contain the desired output for each input.

2.1.3 Training Module - Neural Network Simulator

After the problem definition and the data chosen and prepared, the neural network model must be determined. After this choice, the Neural Network Simulator is used to model the neural network and to control its training. General parameters, like the learning rate and the activation function, and other model-specific parameters are configured. These parameters set the training speed and generalization degree of the neural network, among other factors. The neural network performance must be monitored during the training to evaluate if the neural network is converging or if any parameter was badly chosen. In [5] and [6], considerations about neural networks training for various models are made. The goal is to optimize the neural network performance over the test patterns. For that, it's recommended to test its performance over the test data while the training is done. The Acceptance Criteria, the parameter that determines if the neural network is trained, depends on its model.

At last, the outcome of the Training Module is the trained and tested neural network. The Training Interface (TI) transmits the resulting data from the Neural Network Simulator (the matrix of connection weights) to form a Neural Module. These weights represent the knowledge acquired by the neural network after training.

2.1.4 Neural Module

The Neural Module consists on the resulting neural network of the Training Module and information about its architecture and objective. The trained neural network is an application module that must receive inputs (properly scaled and coded) to make a prediction about the ATM network status.

The neural network error should be continuously monitored. If the result is below the Acceptance Criteria, it's possible that the ATM network dynamics have changed: maybe because of changes in the topology or because of the introduction of new services. Although, small changes in the ATM networks should be absorbed by the generalization and adaptability abilities of the neural network.

The Control Interface (CI) passes to the Management Module the information related to a Neural Module. This information identifies the neural network inputs and the meaning of its outputs. The Management Module uses this information to create the proper agent.

2.1.5 Management Module

The Management Module is responsible for the integration and activation of the neural network through the development of an agent that provides the input data to the neural network and, according to its output, takes the proper actions, aiming a proactive behavior.

Before the development of agents, it's necessary to configure in the Management Module the management environment of the ATM network: devices that will be managed and their respective management mechanisms (SNMP MIBs, ILMI and OAM cells).

When developing an agent, the information about the neural network provided by the Control Interface is processed. For each neural network input, a source is defined. The agent will consult this source before activating the neural network. For each neural network output, it's defined what is the proper action to be taken. In its actions, the agent can use the ATM management mechanisms previously configured. After that mapping, the agent is generated. Then, it must be installed and activated.

The Management Module is also responsible for monitoring the agents, besides allowing direct management of the ATM network. The actions taken by the agent should be recorded for later analysis. The agent should also alert the Management module when the neural network is below its Acceptance Criteria.

2.2 Physical Architecture

The figure 2 presents the Physical Architecture of RENATA, their modules and how the information is exchanged. It introduces a new element in the Training Module: the Neural Network Documentator (DocNN). This tool is responsible for the creation of the file that contains information about the neural network. This file is present in a Neural Module.

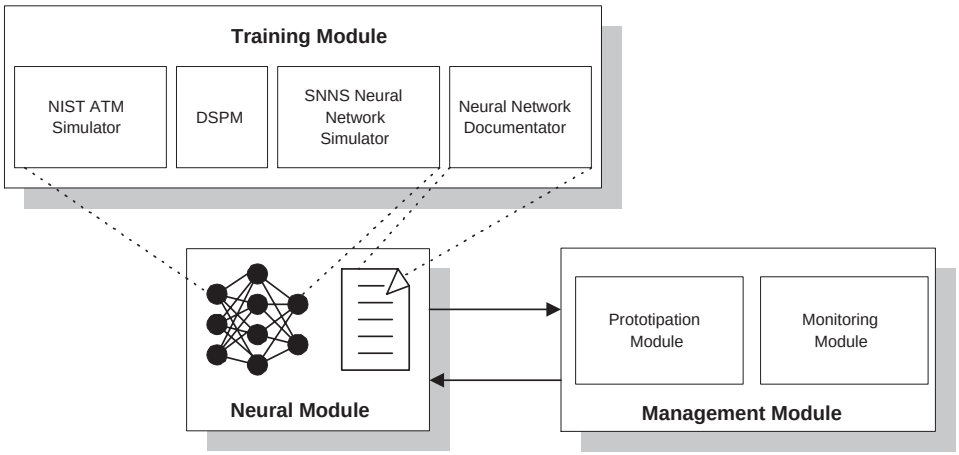

Figure 2: RENATA Physical Architecture

Physically, the Management Module is divided in two: the Prototipation Module and the Monitoring Module. The first one is responsible for the agent generation based on the neural network information and on the ATM network configuration. The other module monitors the developed agents and the ATM network.

The most complex part of the development of an agent in RENATA takes place in the Training Module: the modeling and training of a neural network that predicts certain situation in order to perform proactive management. From the created Neural Module, the Management Module offers facilities to develop intelligent agents. The resulting agent consists basically on ATM management mechanisms and the on knowledge to predict determined situation.

Next, the chosen options to the implementation of each RENATA module are presented. In the case of the ATM Simulator and the Neural Network Simulator, due to their complexity, existant products were adopted. These products are integrated aiming the training of a neural network. For the other modules, the following tools had to be implemented: the DSPM, the Neural Network Documentator and the Management Module.

2.2.1 Adopted Products

ATM Simulator

The *NIST ATM Simulator* was developed in the National Institute of Standards and Technology (NIST) to provide an environment of study and determination of the performance of

ATM networks. In a graphic and interactive environment, the user can create various network topologies and set the operation parameters of each component. While the simulation is running, various performance measures can be displayed on screen or saved to files for subsequent analysis [7].

Neural Network Simulator

The *SNNS* (Stuttgart Neural Network Simulator) has been developed in the Stuttgart University since 1989. The goal of the project is the creation of a flexible and efficient environment for research on and applications of neural networks [6]. The simulator can be used for create, model, train, test, analyse and visualize neural networks.

Neural Module

The Neural Module consists on the neural network produced by the Training Module and on its documentation generated by the DocNN. From the neural network trained in the SNNS, a tool from the SNNS package is used. The *snns2c* generates C code representing the neural network.

2.2.2 Implemented Tools

DSPM

The DSPM is the module responsible for the selection and preparation of the data that will be submitted to the neural network. The DSPM translates the file in the NIST format to the SNNS format, selecting the relevant data to the neural network learning. Besides, the DSPM also does the necessary treatment on the brute data so that it can be accepted by the neural network.

Neural Network Documentator

In order to the agent may provide the input data to the neural network and interpret its output, it's necessary for the Management Module obtain this information when creating the proper agent. These data are generated by the Neural Network Documentator (DocNN).

In DocNN, information about the objective of the neural network, about its architecture (inputs and outputs), how it should be activated and evaluated and some training parameters among other are provided. The information about the inputs defines what data the neural network must receive in order to predict and how these data should be treated before being submitted to the neural network. For each output, it should be provided the interpretation of such prediction and a suggestion of the action to be taken. The DocNN was implemented in Java, mainly because its portability features.

Management Module

The Management Module functionalities are divided between its sub-modules. The Prototipation Module is responsible by the code generation of the agent according to the information generated by the DocNN and to configuration and to the ATM management information; *i.e.* the configuration of the sources for the neural network inputs and the instruction of how the

agent should act according to the neural network prediction. The Monitoring Module offers access to the ATM management, besides being responsible for the management of the agents generated by the Prototipation Module.

Both were implemented in Java. The Java SNMP API developed by the Advent Network Management, Inc. was used. This API allows the development of Java applets and applications that use SNMP to communicate with the managed nodes. It was used the API version 1.3.1 for SNMPv2C, since the RENATA developed agent can consult SNMP MIBs for retrieve information necessary to the neural network prediction, besides also allow itself monitoring by Monitoring Module.

The Prototipation Module generates Java code for the agent. The neural network is accessed by native method calls. The agent code must be compiled and then installed in the pre-established device that must have the Java virtual machine.

2.3 Using RENATA

Access to ATM Management Mechanisms

After an initial configuration phase (using the RenataSetup tool), where all management mechanisms in the network in question are incorporated by RENATA, the agents and the user through the Monitoring Module can consult SNMP MIBs with a MIB Browser or start OAM cells flows. This way RENATA allies the ATM management mechanisms with the neural networks intelligence in order to endow the developed agents with a proactive attitude.

Besides offering direct access to the ATM management mechanisms (SNMP MIBs, ILMI and OAM cells), RENATA allows through these mechanisms the selection of the information that will be monitored by the agent. By its turn, the agent passes these information, after the proper treatment to the neural network, that will make a prediction.

In the case of prediction of abnormal situations, the action that should be taken by the agent must also be configured between those available in the ATM management mechanisms. For example, the agent can act directly over the device (via SNMP, ILMI or OAM) or only notify the user.

User Interface

RENATA has two types of user: the *developer*, that uses RENATA for creating neural network applications for ATM management; and the *network administrator*, that configures these applications for running on his network, through agents prototipation.

The developer carries about all process, mainly about the modeling and training of a neural network for detection of abnormal situations in an ATM network or for resource management applications. The administrator deals with the configuration, installation and monitoring of the agents. Each user has its own interface, different in their tools and purposes. The figure 3 shows the prototype with its interfaces.

The *I_DES* (Developer Interface) allows access to the Training Module facilities: the ATM simulator, the DSPM, the Neural Network Simulator and the Neural Network Documentator. From the I_DES, the developer creates neural networks that later will be activated by agents configured by the administrator. The *I_ADM* (Administrator Interface) integrates the ATM management mechanisms and tools for agent prototipation, *i.e.* the Management Module and the Prototipation Module. The common element between the two interfaces is a Neural Module.

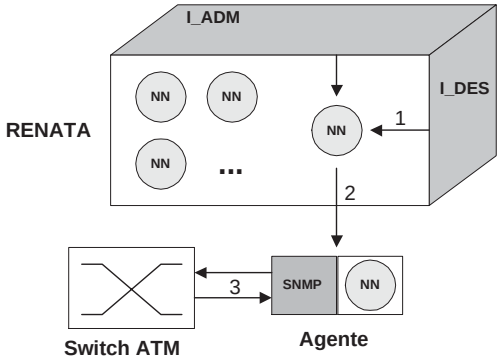

Figure 3: RENATA Prototype

The figure 3 also shows the prototype general operation, divided in three big phases. In phase 1, the developer creates a Neural Module, using the Training Module tools. In phase 2, the administrator configures an agent that will encapsulate the created neural network. The phase 3 exemplifies the developed agent operation, communicating with an ATM switch through SNMP.

The table 1 details in steps the phases for the development of an agent in RENATA, relating the tools and interfaces used.

I_DES	
Step 1:	Problem Definition
Step 2:	ATM Network Simulation Tool: *NIST Simulator*
Step 3:	Preparation and Selection of the Data Generated by the ATM Simulator Tool: *MSPD*
Step 4:	Neural Network Project, Training and Validation Tool: *SNNS Simulator*
Step 5:	Neural Network Documentation Tool: *Neural Network Documentator*
I_ADM	
Step 6:	Configuration of ATM Management Mechanisms Tool: *RENATA_SETUP*
Step 7:	Agent Prototipation Tool: *Prototipation Module*
Step 8:	Agent Activation and Installation
Step 9:	Agent Monitoring Tool: *Monitoring Module*

Table 1: Steps and Tools of a RENATA Agent Development

3 Implementation

3.1 Prototype Description

One of the objectives of this work is the creation of a generic development environment of neural network-based agents. All the tools were implemented in Java, with the JDK (Java Development Kit) 1.1.7 version provided by Sun Mycrosystems. This guarantees that RENATA will runs over any platform that supports the Java virtual machine. For the visual interface, it was used the Borland JBuilder version 1.2. The interface code generated by JBuilder was modified so that it wouldn't be dependent on any Borland class.

Since it is practically impossible the creation of a generic DSPM, this module wasn't implemented because it's intrinsically specific to each neural network application. So, the developer must program the DSPM suitable for its application. The following tools were implemented the DocNN, the RenataSetup, the Monitoring Module and the Prototipation Module.

DocNN

DocNN is the tool used by the developer for describing the characteristics and purpose of the neural network created in the Training Module.

Important parameters like the Prediction Interval (time that the agent must wait for activating the neural network, in milliseconds), the Activation Mode of the agent (always active, when is requested or depending on some network parameter) and the Acceptance Criteria (how the neural network error must be evaluated) are configured in the DocNN. Some training parameters are stored in case of a possible retraining. Such parameter values could be changed or used again.

In the architectural configuration, the developer provides initially the number of inputs and outputs. For each input (figure 4), the developer must provide its description, the data original type and the input type allowed by the neural network (float or binary). The data original type can be float or integer (simple, an array element or a computed attribute), string or data. The combination between the original type and the input type determines how is going to be done the data preparation during the operational phase. For each output, the developer describes its meaning and may give a suggestion of action, in case of its activation.

RenataSetup

RenataSetup is the tool where the administrator informs what are the machines that will be managed by RENATA and which are the management mechanisms presents in each one.

For each device that RENATA may manage, the administrator must provide its name and type (switch, workstation or router), besides its IP and ATM addresses. Next, the ATM management mechanisms (SNMP MIBs, ILMI and OAM cells) available on that machine. For each MIB, the administrator must inform the read and write communities, besides the port number of the SNMP agent for that MIB. In the ILMI case, the administrator must configure the community of each IME (local and remote), besides its SNMP agent port. Concerning to OAM cells (F4 and F5 flows), the administrator must inform if these are allowed in the device; if so, what is the driver for this mechanism.

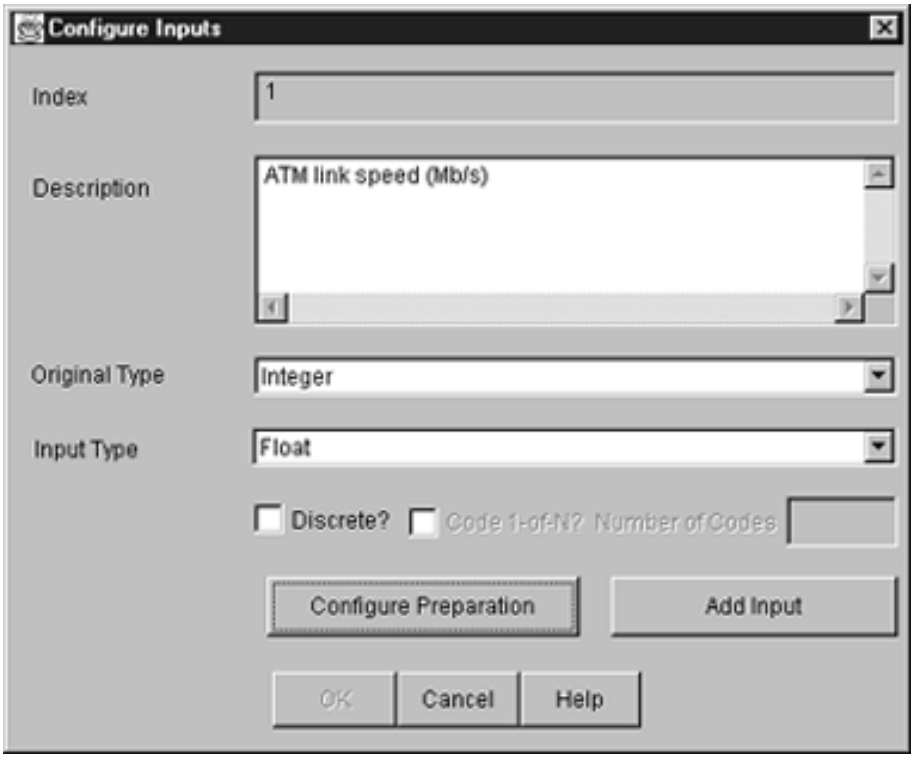

Figure 4: Neural Network Input Configuration at DocNN

Monitoring Module

The Monitoring Module is the tool that provides direct access to ATM management mechanisms and information, besides of allowing the monitoring of the agents developed in RENATA.

For each device that was configured in RenataSetup, it is possible access its SNMP MIBs though a MIB Browser. Adapted versions of this browser are used for access to the ILMI MIB (local and remote IME) and for allowing the monitoring of RENATA agents that are installed on that device, through request to a specific MIB developed for this purpose.

Prototipation Module

The Prototipation Module is the tool that generates the agents from the information about the management mechanisms of the ATM network configured on RenataSetup and from the information about the neural network generated by DocNN. After loading the file generated by DocNN, for each neural network input, the administrator configures where the agent should collect this information (in a SNMP MIB or in ILMI MIB, via SNMP SET in its own MIB, by OAM cells or it's a fixed and predetermined value).

According to the meaning of each output, the administrator configures what should be the action of agent: none, a SNMP SET, send a trap to a manager or send a e-mail for the administrator, communicating the problem.

3.2 Agent Description

In this section, the general operation of the agent is explained. In the agent initialization, two threads are activated: the PredictionThread (responsible for the integration with the neural network) and the ReceiverThread (responsible for responding external information requests). Depending on the Activation Mode and the Prediction Interval, the PredictionThread activates other threads that are responsible for collecting information for the neural network inputs. Parallely, these threads actualize the input vector that will be submitted to the neural network. In the case of the source for the input to be a SNMP GET, a good optimization was achieved because the socket opening and PDU fulfilling are done only in the thread initialization. During predictions, the time was practically reduced to almost the round-trip time between the two devices.

The neural network makes its prediction, updating the output vector. After the activated output is decided, the corresponding action is taken after the activation of other thread, the ActionThread. Next, the action is registered and after, it's evaluated. This way, RENATA uses a lot the Java thread mechanism, which allows a good optimization in the agent operation, that is necessary in a high-speed environment.

In order to allow the agent monitoring, including by other management systems, one MIB was developed according to SMIv2 (Structure of Management Information) with information related to the RENATA agent and to the neural network. The figure 5 shows the structure of RENATA MIB, that corresponds to the information that was provided before and to those related to the operation of the agent.

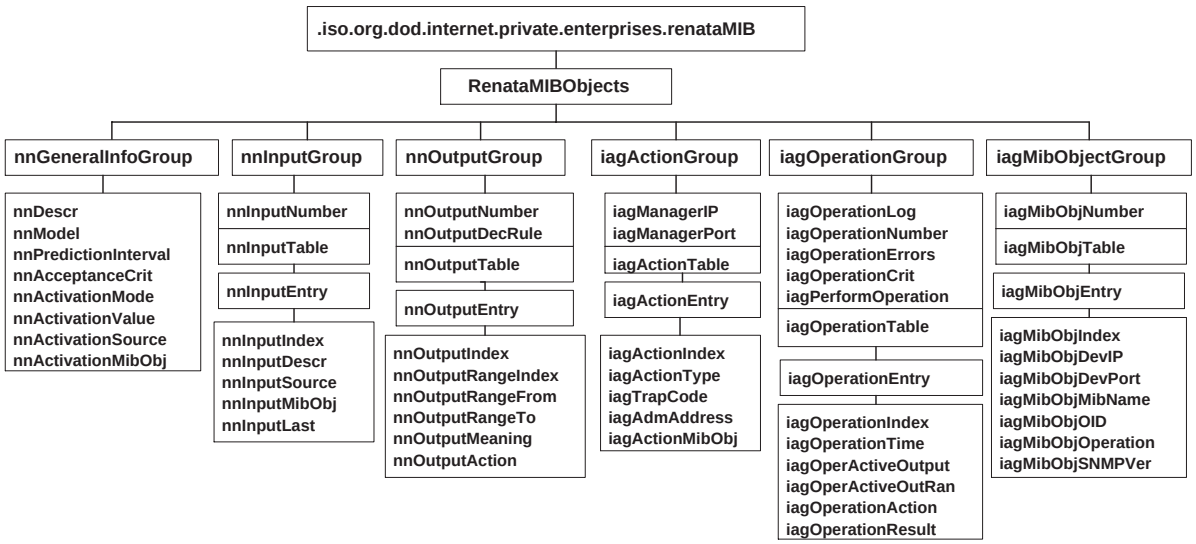

Figure 5: RENATA MIB

4 Conclusion

RENATA is a flexible experimentation environment of the use of neural networks in ATM management, allowing the development of intelligent agents. In other approaches, the user would have to worry about all details of implementation of the agents. RENATA provides

facilities to the integration of a neural network in an ATM network, also allowing software test of neural solutions.

Several MIBs have been analyzed and it was observed that there is few Managed Objects that model aspects of traffic management and network congestion in ATM [8]. This limitation has reflexs in this first version of RENATA. However, the extensions of RMON MIB for ATM and the use of low-level routines may soothe these shortcomings.

Aspects of inter-agent communication are possible extensions of this work. A first idea is to use an inter-agent communication language, like KQML (Knowledge and Query Manipulation Language) or though CORBA (*Commom Object Request Broker Architecture*), that could be used to provide access transparency and interoperability. However, for some application the overhead due to CORBA must be evaluated, since it could compromise the efficiency of the solution, that intends to be proactive. Among other possible extensions, the most natural is the development of new agents from this environment. RENATA makes possible the development of a new class of ATM management applications that use neural networks.

References

[1] M. Oliveira, M. Franklin, A. Nascimento, and M. Vasconcelos, *Introdução à Gerência de Redes ATM*. Editora CEFET-CE, segunda ed., 1998.

[2] M. A. da Rocha and C. B. Westphall, "Proactive Management of Computer Networks using Artificial Intelligence Agents and Techniques," in *Fifth International IFIP/IEEE International Symposium on Integrated Network Management*, May 1997.

[3] M. Cheikhrouhou, P. Conti, and J. Labetoulle, "Intelligent Agents in Network Management, a State-of-the-Art," *Networking and Information Systems Journal*, vol. 1, no. 1, pp. 9–38, 1998.

[4] S. A. Youssef, I. W. Habib, and T. N. Saadawi, "A Neurocomputing Controller for Bandwidth Allocation in ATM Networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 191–199, February 1997.

[5] J. P. Bigus, *Data Mining with Neural Networks*. McGraw-Hill, 1996.

[6] U. of Stuttgart, *SNNS - Stuttgart Neural Network Simulator - User Manual, Version 4.1*, 1995.

[7] N. Golmie, A. Koenig, and D. Su, *The NIST ATM Network Simulator - Operation and Programming, Version 1.0*, August 1995.

[8] L. M. R. Tarouco and M. de Fátima Webber do Prado Lima, "Análise do Gerenciamento dos Mecanismos de Policiamento de Tráfego em Redes ATM Através de Objetos Gerenciados," in *XVI Simpósio Brasileiro de Redes de Computadores*, pp. 319–338, May 1998.