# Automatic Configuration of PVCs in ATM Networks with Software Agents

*M. Cheikhrouhou, P. Conti, J. Labetoulle*
*Corporate Communications Department*
*Address: Institut Eurécom, BP 193 – 06904 Sophia-Antipolis Cédex – France*
*{Morsy.Cheikhrouhou, Pierre.Conti, Jacques.Labetoulle}@eurecom.fr*
*Tel: +33 4 93 00 26 48*

**Abstract**

This paper describes an agent-based approach to the automatic provision of Permanent Virtual Channels (PVC) in ATM networks. The agent framework described fosters flexibility and seamless evolution by the definition of capability skills. Skills are plugged into the agent's brain which is responsible for the coordination of the agent's behavior. The application of this agent framework to the problem of PVC configuration resulted in the definition of a set of skills, among which, only one skill is device-dependent. Therefore, the implemented application can easily support new types of ATM switches. Moreover, it is efficient in terms of performance and bandwidth saving.

**Keywords**

Distributed Network Management, Intelligent Agents, ATM Management.

## 1. Introduction

Currently, there is no standard protocol to automate the provision of Permanent Virtual Circuits (PVC) in ATM networks [1]. Moreover, each ATM vendor provides its own interface to manage PVCs. This makes the task of PVC creation and management particularly intricate in heterogeneous environments.

To the best of our knowledge, two other research projects tackled the problem of PVC provisioning. The Utopia Project [2] provides a Web-based interface to the

management of ATM cross connects. However, at the time this paper was written, the Utopia product did not offer a global view of end-to-end PVCs and only allowed to configure PVCs switch by switch.

The second project is described in [1] where an approach based on the concept of Mobile Agents (for example, see [3, 4, 5] for a description of the mobile agent concept) is described and compared to other classical approaches in network management. The implementation is experimented in a simulated testbed.

Alternatively, in our paper we present another approach based on the use of flexible software agents. Software agents [6, 7, 8] represent a new rapidly evolving paradigm for developing software applications [9]. There is little consensus on how an (a software) agent could be defined [6] or how it is designed [7].

We consider the agent concept as *a software entity that acts autonomously on behalf of a user and capable of social communication with other agents*. Within the DIANA (Distributed Intelligent Agents for Network Administration) project [1], we are developing an agent framework for Network Management (NM). Our architecture aims to foster agent flexibility and ease of development with a modular approach. The work presented in this paper was achieved during a case study that was implemented to test and improve the agent framework.

In the following section, we provide the reader with the necessary background in ATM PVC management. The problem scenario that is to be implemented is described in Section 3.. Section 4. describes the agent framework and its concepts. It also describes the steps to be followed to build an agent-based application. Section 5. presents how the case study of PVC configuration is designed using our agent framework. Implementation related results and result assessment is presented in section 6.. We show how our solution is efficient for example in terms of management traffic which is kept small. Also, our solution is easily portable towards new types of ATM fabrics. Finally, we conclude the paper with remarks and future directions.

## 2. ATM Background

There are two main types of end-to-end circuits or VCCs (Vitual Channel Circuits) in ATM networks: Switched Virtual Circuits and Permanent Virtual Channels. Switched Virtual Channels (SVC) are established automatically using ATM signalling protocols UNI and PNNI [10]; whereas Permanent Virtual Channels (PVC) are established by the human operator on a switch-by-switch basis. PVCs can be useful in many cases. For example, most of the existing ATM switches cannot offer SVCs with all the standardized traffic contracts (e.g. [11]). Currently, most of the ATM fabrics can only support SVCs with either UBR or CBR traffic. Moreover, PVCs are permanent connections. They can be established permanently between two end-hosts in order to have at any time, QoS-guaranteed and immediately usable connection. PVCs are more efficient for connections between hosts that communicate frequently since there is no

[1] The work achieved in the DIANA project was financially supported by SwissCom.

waiting time due to the establishment of the connection using the ATM signalling protocol. Finally, an ATM equipment may not support a complete implementation of the ATM signalling protocol. In fact, current implementations of signalling mechanisms are not completely compatible between devices from different providers. Furthermore, some ATM equipment providers implement proprietary signalling mechanisms such as SPANS [12]. Therefore, SVCs cannot be established in a heterogeneous ATM network. In this case, the only way to establish end-to-end connections is to use PVCs.

The establishment of a PVC is not a simple task. Firstly, a physical end-to-end route between the source and the destination must be selected. The route is a non-empty list of nodes. Each node is a triplet that identifies a switch and its selected input port and output port that are going to be used. There might be several physical routes and one of them must be chosen.

Once a route is identified, the next step will be to plan which Virtual Paths (VP) are going to be used to contain the PVC. The simplest way is to use the permanently established VP with identifier 0 (VP 0 for short). VP 0 is created and maintained on each port of any ATM hardware as soon as it is switched on. However, VP 0 cannot be used indefinitely since each VP has a limited capacity and can support only a limited range of VCI (Virtual Channel Identifier) values.

Instead, the approach used by ATM operators generally consists in creating Virtual Paths between each consecutive switches on the route. Though this shows to be far from being an optimal solution in terms of the number of supported VCCs, it presents the advantage of easily and quickly establishing PVCs.

The final step is to attribute VCIs on each switch and to create the VC route entry. On each switch, a VCI is needed for the input port and another for the output port. The output VCI of an intermediate switch must be the same as the input VCI of the next switch. In the case where local VPs are created on each switch to transport the PVC, the outgoing VPI of a switch must also be the same as the incoming VPI of the next switch. If one of these constraints is not satisfied, then data transmitted on this PVC will not reach its destination. These are the sources of difficult configuration errors that are hard to diagnose.

But what actually hardens the task of PVC creation even more is the problem of fabrics heterogeneity. Untill now, every ATM fabric provider elaborates its own management interface. In most of the cases, a telnet configuration and administration interface is offered. Moreover, even when SNMP is supported as a management protocol on most of the ATM switches, each provider uses a different MIB than the others. Therefore, the human network operator must know all these management interfaces to be able to appropriately create an end-to-end PVC.

Finally, each end-to-end ATM connection have a Usage Parameter Control (UPC) contract that specifies the characteristics of the traffic that is going to be transferred. For the same PVC, each switch that conveys this PVC must be configured to the same UPC traffic contract, i.e. using the same parameters. If a UPC parameter is wrongly configured at any switch, then the whole traffic on the PVC could be affected. Again, troubleshooting such abnormal behavior is particularly hard. At least, it requires to

check the UPC parameters on each switch that routes the PVC.

In summary, the establishment of a PVC between end systems needs to take into account a lot of parameters and has to satisfy some constraints that are hard to check especially in a heterogeneous environment. Therefore, a management application that allows to automate the provision of PVCs can really help ATM network operators in providing a more rapid service to their customers.

# 3. Problem Scenario

Let us suppose an ATM network operator is providing a Virtual Private Network (VPN) for a customer enterprise. Two distant users in the enterprise may want to establish a video-conference with high audio and video quality. In this case, a Constant Bit Rate (CBR) traffic contract is required with a Peak Cell Rate of about 500Kbit/s, a Cell Delay Variation Tolerance of 0.1ms and a maximum Cell Transfer Delay of 1 second [13]. If there are devices among the ATM equipment that do not support SVCs with CBR traffic, then there will be no Quality of Service guarantee during the video-conference. A possible alternative could be to establish two opposite PVCs with the desired QoS that allow the two users' hosts to communicate with a guaranteed QoS.

If no automatic PVC provision system is available, then one of the users should contact the network operator (by phone for example). The latter then checks whether it is possible to create the PVC, and later, contacts back the requester to inform him that the PVC is now ready to be used. He also has to provide him with the VPI:VCI values in order to configure the ATM software at his host in order to use the VCC with those values. The whole process takes at least several minutes to be completed. It could last even longer if there are more than one ATM network provider that have to cooperate together to establish the PVC.

Now, suppose that the user has a special software, a kind of User Assistant Agent, that automatically detects that he wants to use the video-conference software. The User Agent obtains the destination address and determines which UPC best fits the communication requirements. It then takes the initiative to contact another software on the network operator side and to request the establishment of the PVC with the other user equipment. In a matter of seconds, it shall receive an answer as to whether the PVC could be created and in that case, it will be supplied with the VPI:VCI parameters. The user agent can then automatically setup the ATM software layer on the user equipment and inform the user that he is ready to start his video-conference.

# 4. DIANA agent architecture

Our agent architecture aims at providing flexible and dynamic software agents. Agents are wanted to be able to acquire new capabilities and skills seamlessly without interrupting their operation. This is essential for network management purposes where network elements may need to be upgraded frequently and therefore, the management

application needs to be easily adaptable. For these reasons, our agent architecture is based on two major component types: the Brain and the skills. Skills provide the agent with capabilities and behaviors, while the Brain is the "headmaster" that accepts and manages agent skills.

## 4.1 The Agent's Brain

The Brain (Figure 1) offers basic and innate facilities necessary for the agent operation. These facilities are either local facilities, i.e. for the agent's local operation; or inter-agent facilities, i.e. responsible for communications and social interactions with the other agents.

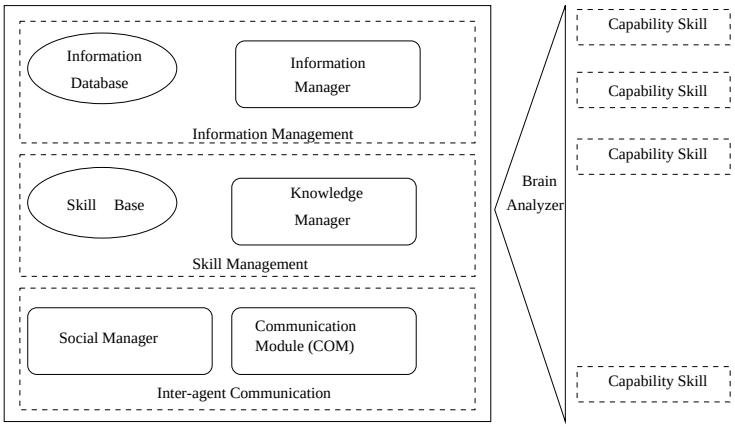


Figure 1: DIANA agent architecture

### Local Facilities

Locally, the Brain is responsible for maintaining the agent's information database. The information database holds network management information as well as information about the other agents and about the agent itself. The agent's information can be accessed concurrently during its operation. Therefore, the Brain includes an *information manager* that ensures the coherent access to the information database and maintains its integrity. The low-level term "information" is used purposely because there is no commitment in our agent architecture to any particular structure of this information.

As a general rule, a skill that creates a certain information in the information database is marked as the owner of this information. Whenever another skill tries to update this information, the brain is reponsible for notifying the owner skill. In this way, the owner skill is able to check whether the update is allowed or not, thus ensuring the coherence of the information database.

The Brain is also responsible for the management of the agent's skills. Skills can be downloaded on-the-fly and integrated into the agent inside its *skill base*. If a loaded skill is of no more use to the agent operation, it can be disposed off so that to keep the agent as small as possible in size. Newly loaded skills may require pre-requisite skills, and the Brain is responsible for checking whether or not these skills are available and are already loaded into the agent. If a skill is necessary for another one, and is not yet loaded, the Brain is responsible for searching for it either locally or via the help of other agents.

When a skill becomes active, it makes use of the agent's database by creating, updating or deleting pieces of information. A skill operation may depend on the information maintained or generated by other skills, and the Brain is therefore in charge of dispatching asynchronously this information and its updates to the interested skills in a transparent way. These facilities are provided by the *knowledge manager* which holds the necessary information about the skills in the *skill base*.

These three functions are governed by the *brain analyzer* which is responsible for the parsing of the messages that the Brain receives, either from the skills or from the inter-agent communication.

#### Inter-agent Facilities

The Brain offers also inter-agent communication facilities that allow skills from different agents to interact in a transparent way. A communication module inside the agent is responsible for sending to and receiving requests from the other agents. Another module, the *social manager*, holds information about the other agents, such as the host on which they run as well as its address. Therefore, skills only deal with the symbolic names of the distant agents they want to interact with, and they are not aware of distribution-related details in the agent system.

In our current implementation, the communication module can use HTTP, raw TCP, UDP and SMTP to exchange messages between the agents. Future versions will provide support to distributed computational environments such as CORBA and Java RMI.

### 4.2 Skills

An agent skill is a piece of software specialized in a network management area and can be plugged-in dynamically into the agent to enrich it with a new capability. It offers new services and more elaborated pieces of information to other skills. For this, it may use the information and the services offered by other skills that are supposed to operate at a lower level.

The skill has an interface that communicates initialization information to the Brain. This information declares, using a proprietary lisp-like language, the pre-requisite skills it needs for its operation, the set of services it offers to higher-level skills. For each service, the required type of information, the requested services from other skills

and the produced information are declared. During skill initialization, this information allows the Brain to determine whether all the necessary pre-requisite skills are available or not. During the skill operation, the Brain can determine which skill is concerned by a service request and automatically forward it to that skill. Furthermore, it is able to determine, according to which services are currently requested from that skill, which information should be dispatched to the skill when this information is created, updated or deleted.

Agent skills can be related in many possible ways according to the characteristics of the management application to be developed. One possible way to organize skills is to have an abstraction hierarchy in which the lower-level skills provide abstraction means to higher-level skills. For example, a high-level skill can provide a uniform view to access and manage any ATM switch. This skill will be based on low-level skills that are dedicated to specific ATM switches. The low-level skill produces information elements that are close to the proprietary information model provided by the switch's management interface. The brain is therefore responsible for notifying the high-level skill of any change that occurs in the low-level skill. In this way, the high-level skill is able to build its uniform view of the ATM switch and keep this view constantly updated. Conversely, the higher-level skill can be asked by other skills to perform management operation on an ATM switch. These management operations are therefore translated into calls to the low-level skill. Interestingly, the low-level skill and the high-level skill can be deployed on two separate agents in a transparent way. This is due to the inter-agent facilities and to the transparent communication mechanism offered by the brain.

## 4.3 Developing Agents

This section provides an overview of the agent development process suggested for our skill-based agent architecture. Mainly, the process is divided in three phases: macro-design, micro-design and agent deployment.

1. **Macro-design**

   In this phase, the developer tries to identify the major agent roles needed for the system to be developed. In general, agents are conceived as peer-to-peer entities but with different roles: Each agent is specialized in some aspect of the overall system. Let us insist on the fact that the roles of an agent can be changed dynamically during their lifetime according to the skills that an agent has at a moment.

   Once the major roles are identified, the behavior of the agent system can be described using interaction scenarios between the agents.

2. **Micro-design**

   This phase is a refinement of the Macro-design. Its focus is on the skills that implement the determined roles instead of on the agents themselves. A set of

skills must be designed to ensure each of the roles identified in the first phase. The developer decides which lower-level skills are going to be used to build the new ones upon.

A skill is specified when the pre-requisite skills, the services it requests from them, the services it offers to the other skills and the information it uses are known. This allows to refine the scenarios described in the first phase to go down to the details of the interaction between the skills themselves.

3. **Implementation and Deployment**

   The skills can be written without paying attention to distribution-related issues (thanks to the Brain inter-agent facilities). The problem of agent attribution to hosts can be handled just after the skills are ready. At this stage, parameters such as CPU load balancing, response time and bandwidth usage can be optimized by placing the agents appropriately throughout the network.

# 5. Agent Design for PVC Provision

In this section, we describe how our agent-based solution is designed. At a first stage, we identify the different agent roles and address the distribution issues. Agent roles describe the responsibility of each agent in terms of abstract high-level tasks. Distribution issues include agent location assignment and global agent communication requirements. However, this is only a conceptual distribution and depends only on the agent roles. It does not take into consideration the actual implementation of the ATM network.

At a second stage, we focus on the micro-design for agents with the same roles. This will identify the different skills needed for each agent to assume a particular role and establish the interactions between the skills.

## 5.1 Macro Design

The problem scenario described in section 3. suggests that the user initiates a request to establish an end-to-end PVC from his site to another remote site. Therefore, the role of a *User Agent* (UA) is introduced. In a general view, the UA is in charge of perceiving the user requirement for a connection to a remote site with a determined Quality of Service. It may also capture the user constraints such as the price range he prefers or imposes for the connection billing. The UA may then negotiate these requirements with the ATM network provider.

The network operator is represented by a set of agents with a different role. The agents on the ATM network side accept PVC requests from different UAs and do their best to establish them. Obviously, the best place where these agents can be located is directly on the ATM equipment (or at least as close as possible). This will reduce network management traffic since PVC configuration operations require the exchange

of a lot of management messages with the ATM network equipment. Therefore, an agent is affected to each ATM switch, and are thus called *Switch Agents* (SA).

When a switch agent receives a connection demand from a UA, it will be the responsible for the establishment of the corresponding PVC. Relatively to a particular PVC request, we call such SA the *Master Agent*. Indeed, it will be up to it to globally coordinate with the other SAs the action sequences that should be taken for the PVC creation. These other SAs are then called *Slave Agents*. A Switch Agent may therefore have different roles according to whether it is the responsible for a PVC creation or not.

## 5.2 Micro Design

### The User Agent Skills

As we have previously seen in the description of its role, the UA behavior can be implemented using two skills. The *Contract Negotiation Skill* is responsible for sending PVC requests to the SA and negotiating the service contract as well as the price. The *User Interface Skill* is responsible for capturing user requests for a connection establishment and to formulate them for the Contract Negotiation Skill.

The current implementation of this case study provides a simplified version of these two skills. The User Interface Skill is only composed of a Graphical User Interface that allows the user to specify the destination that he wishes to communicate with and the desired QoS contract. This request is then forwarded to the Contract Negotiation Skill which, in its turn, delegates the task of PVC establishment to the SA which manages the ATM switch to which the user is connected.

### The Switch Agent Skills

The switch agent role is ensured by four skills: the *Switch Skill*, the *Slave Skill*, the *Master Skill* and the *Topology Skill*.

- **The Switch Skill**

  A switch agent is responsible for PVC configuration operations of the switch it is affected to. The *Switch Skill* is therefore designed to provide services to create and delete VPs and VCs using the SNMP management protocol. There is a switch skill for each different ATM equipment family. For example, the current implementation runs on FORE ATM switches, and therefore, a *FORE ATM Switch Skill* is developed. Actually, the switch skill is the unique part of the whole system that should be adapted for each product family. However, this should be a temporary situation untill standard ATM management MIBs (e.g. [14]) are deployed in the future ATM devices.

  In addition, the switch skill provides information related to the status of the current VPs and PVCs existing on the ATM switch. This information is used by

the *Slave Skill* to decide on the local parameters for the establishment of a new PVC.

- **The Slave Skill**

  The *Slave Skill* is responsible for the local configuration operations that create or delete a PVC fragment on the switch managed by the corresponding SA. For example, it is up to the slave skill to decide whether to create a new local VP in order to convey the PVC within, or to use an already existing VP with sufficient bandwidth available. Also, it determines which VPI/VCI couples are to be assigned to the newly created VPs and PVCs.

- **The Master Skill**

  The *Master Skill* is responsible for the global supervision of the PVC establishment. Once a physical end-to-end route is found between the source and destination end-systems, the master skill contacts the slave switch agents on that route in order to ask them to perform the necessary operations to create the PVC. It is also responsible for handling creation errors that might occur on switches.

- **The Topology skill**

  Finally, the *Topology Skill* helps the master skill to identify a physical route between the source and the destination. The physical route identifies which switches must be traversed by the PVC. For each switch, it determines the input and the output ports that shall be used.

  Finding a physical route is of a minor concern for us since we are more concerned with the PVC configuration than with the routing issues. Yet the topology of the network is hard-coded inside the topology skill source, since the experimental network on which we run the application is not large.

## 5.3 PVC Creation Steps

The creation of an end-to-end PVC requires three major steps that are coordinated by the master skill. These steps are detailed in Figure 2 that shows the interactions between all the skills of a user agent (on the left) and two switch agents (symobolically identified by “baltazar” and “douchka”).

1. **Finding a physical route.** The master skill queries the topology skill for a physical route that links the source to the destination. The topology skills indicates which are the set of switches to be traversed and which input and output ports to be used on each switch on the route. The interaction between the master and topology skills is performed through the second and the third messages in Figure 2.

2. **PVC reservation.** In this phase, the master skill asks the slave skills on the switch agents (including its own agent) to reserve the PVC (messages 4 and 5).

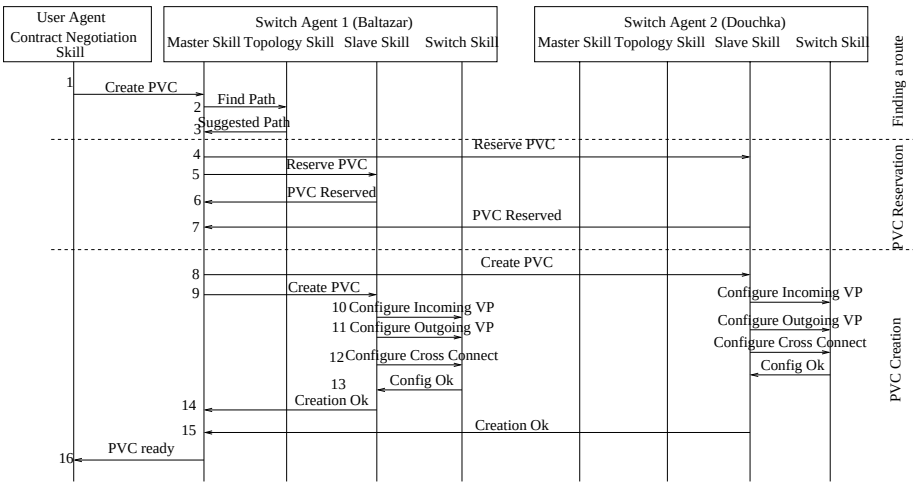


Figure 2: PVC Creation Scenario

Each slave agent then determines whether it is possible or not to accept the PVC (messages 6 and 7). If it can be accepted, then all the parameters (e.g. VC and VP identifiers) of the PVC are determined at this phase.

3. **PVC creation.** If the SA that is responsible for the global creation of the PVC receives positive acknowledgments from the other SAs, then the PVC can be effectively created. Again, the master skill sends creation commitments to the switch agents where the slave skills execute the commitment using the services of the switch skill.

   Finally, when all the creation positive acknowledgments are received, the master skill can send back a message to the UA indicating that the PVC is now created and can be used to communicate with the remote host.

# 6. Implementation Results

The agent system code is entirely written in Java. Therefore, it is completely portable, and the agents actually can be run on Sun/Solaris machines as well as on PC/Windows NT machines.

We have experimented the developed agent system on an experimental ATM network. The experimental ATM equipment is composed of two FORE switches (FORE Runner LE and FORE ASX 200 switches). The network scheme is represented in Figure 3. We have attributed the switch agent *Baltazar* to the FORE Runner LE, and switch agent *Douchka* to the FORE ASX 200. Actually, these two agents run on proxy machines other than the ATM switches themselves. This is because the avail-

able switches do not support the Java Virtual Machine yet. In the near future, network equipment will be Java-enabled and it will be possible to run agents directly on them. Also, we have launched two user agents, *Snoopy* on host “lys” and *Shiva* on host “violette”.

Using the simple GUI that the UAs *Snoopy* and *Shiva* provide, we could establish PVCs with determined traffic contracts between lys (resp. violette) and the other hosts. The time taken for the total establishment of the PVC ranges in general between 1 and 3 seconds, and never exceeds 4 or 5 seconds. This response time should however be even more interesting if the switch agents are run directly on the switches.

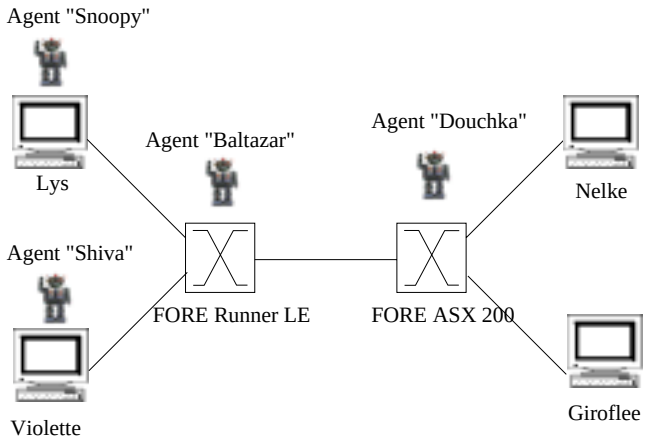


Figure 3: Experimental ATM network and Agent distribution

The experiment showed the gain we obtain on the cost of management traffic compared to a centralized approach. If a centralized approach was used, all the SNMP requests and responses would be transmitted on the network, between the management station and the ATM switches. In our implementation, only high-level agent communications are exchanged via the network. All the many SNMP primitives are performed locally on the switch.

This makes our agent-based solution highly scalable. Since every switch is supposed to run its own agent, there is no potential bottleneck for the processing capacity as in centralized approaches. Also, there is no global central control of the agent system. Therefore, the management traffic is also distributed throughout the ATM network. Here also, there is no potential bottleneck like that in the centralized approach were management traffic is high around the central management station.

The only part that is switch-dependent inside the agent is the switch skill. It offers a common logical view to manage PVCs which is then translated to switch-specific management operations. These operations can be carried out via any management interface offered by the switch. In our case, we developed a unique switch skill for

both FORE switches. This skill uses SNMP to access FORE ATM MIB that allows to create and delete PVCs.

All the other skills are independent from the ATM switch. This makes the application easily portable and extensible to any new kind of ATM device. Moreover, the agent is able to dynamically integrate new skills or new versions of skills seamlessly during its operation. Therefore, the network operator need not stop the agent system in the case of a management software upgrade in ATM switches.

Finally, our approach has an interesting property of graceful degradation. The agent system can operate even though one of the switch agents crashes for some reason or another. Obiously, no more PVCs can be established via the switch that has been running the crashed switch agent; However, if another route avoiding that switch can be found, the PVC still can be established.

# 7. Conclusion

The paper described an agent approach to implement a system for the automatic provision of PVCs in ATM networks. Our agent framework is based on two main components: the agent's brain and the skills. Skills can be dynamically integrated into a running agent to provide it with new capabilities. They can be fetched on-demand from any location in the agent system. Moreover, the whole agent framework is written in Java. These features made our agents very flexible and easily deployable in heterogeneous and dynamic environments.

The implementation of the PVC provision system lead to the definition of two main agent roles: the User Agent and the Switch Agent. The User Agents are deployed on the end-user machines, whereas the Switch Agents are deployed inside the switches and ATM fabrics in the network. A few set of concise skills were enough to implement both agent roles.

Using the implemented system, we could create and delete end-to-end PVCs easily, by simply specifying the source, the destination and the traffic contract. The generated configuration traffic is kept small in size since the switch agents perform management operations directly on the switches. Only high-level agent communication is transported on the network.

Thanks to the modularity offered by the concept of skill, the application was easily and dynamically portable to any new kind of ATM switches. Only one skill should be tailored for this reason, while all the others are device-independent.

# References

[1] Bernard Pagurek, Yanrong Li, Andrzej Bieszczad, and Gatot Susilo. Network configuration management in heterogeneous ATM environments. In Sahin Albayrak and Francisco J. Garijo, editors, *Intelligent Agents for Telecommunication*

*Applications - IATA'98*, number 1437 in Lecture Notes in Artificial Intelligence, Paris, France, July 1998.

[2] Utopia user manual version 4.1. http://wwwsnmp.cs.utwente.nl/nm/research/projects/utopia/release4.1/manual4.1.html, 1998.

[3] T. Magedanz, K. Rothermel, and S. Krause. Intelligent agents: An emerging technology for next generation telecommunications? In *INFOCOM'96*, pages 464–472, USA, March 24-28 1996. IEEE.

[4] Mario Baldi and Gian Pietro Picco. Evaluating the tradeoffs of mobile code design paradigms in network management applications. In R. Kemmerer and K. Futatsugi, editors, *20th International Conference on Software Engineering (ICSE'97)*, Kyoto (Japan), 1997.

[5] Mobile agent facility specification, June 1997. OMG TC Document cf/xx-x-xx.

[6] Hyacinth S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(3):205–244, October/November 1996. Available at http://www.cs.umbc.edu/agents/introduction/ao/.

[7] Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[8] Morsy Cheikhrouhou, Pierre Conti, Raúl Texeira Oliveria, and Jacques Labetoulle. Intelligent agents in network management, a state of the art. *Networking and Information Systems*, 1(1):9–38, 1998. http://www.eurecom.fr/~cheikhro/docs/StateOfTheArt.ps.gz.

[9] Nicholas R. Jennings and Michael J. Wooldridge. *Agent Technology: Foundations, Applications and Markets*, chapter Applications of Intelligent Agents. Springer-Verlag, February 1998.

[10] Uyless Black. *Signaling in broadband networks*, volume ATM: Volume II. Prentice-Hall, 1998.

[11] ForeRunner ATM switch configuration manual . http://www.fore.com/products/manuals.htm, March 1997.

[12] SPANS: Simple protocol for ATM network signaling. http://www.mi.infn.it/INFN/atm/articles/spansfore.ps, 1998.

[13] Anthony Phlipponneau and Jean-François Milhomme. ATM to VPN performance management. Second term project, Institut Eurécom, November 1997. ACTS Project AC052: PROSPECT.

[14] M. Ahmed and K. Tesink. Definitions of managed objects for ATM management version 8.0 using SMIv2. RFC1695, August 1994. ftp://ftp.nus.sg/pub/docs/rfc/rfc1695.txt.