Call Management using Software Agents

R. Marques and B. Marques and R. Teixeira de Oliveira

Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto
Portugal

In this paper, an agent-based platform to control and to manage the interaction between customer and different ATM service provider networks is addressed. The aim of the presented approach is to offer an ATM end-user Quality of Service (QoS) and costs information, enabling him to establish a proper routing policy when multiple switched-based ATM WANs are present. The system architecture described allows the retrieval of fundamental information by certified agents, potentially possessed by concurrent ATM service providers (ASPs), revealing a real market-based behavior. As a result, the attributes and metrics of each possible path will translate the QoS offered and the costs charged, allowing the end-user to choose the ASP that best suites his needs.

Keywords: Software Agents, ATM Call management, PNNI networks, SVCs, Resource allocation, QoS, Abstract Communication Languages, Content Languages, KQML.

1 Introduction

With the recent introduction of the concept *Global ATM* in the telecommunications market, several ATM carriers are willing to offer Quality of Service (QoS) connections to their corporate customers. If fact, when it becomes to connect private networks to ATM carrier services, a wide variety of options are nowadays available. ATM Service Providers (ASPs) usually offer a wide variety of access configurations, namely ATM circuits between corporate customers in the same local access and transport area or across long-haul circuits. In this latter case, it is frequent to find scenarios where corporate customers ask for the establishment of QoS connections to set up a communication channel with other corporate domains sited in a different country, possibly belonging to the same enterprise organisation.

Nevertheless, there are still some huge difficulties to overcome. On one hand, the customer has other truly competitive alternatives at a lower price such as Frame Relay or leased lines. On the other hand, interoperability between different ATM domains is still an open issue. A set of paradigms must coexist when a customer asks for end-to-end QoS, crossing different ATM networks. Different routing protocols and addressing plans, non support of switched virtual connections (SVCs), different upper-layer protocols and so on, are some of the incongruous characteristics found when analysing heterogeneous ATM networks.

Additionally, there are still other issues that need to be addressed when referring the role an ATM carrier must accomplish with respect to their customers [SvdWvBea99]. ATM carriers have to develop their own means to control and manage customers access in the set-up phase, possibly enforcing a connection admission policy. In fact, it is known that most equipment is limited to the Connection Admission Control (CAC) function. The CAC function controls the network resources, accepting or denying a new connection, depending only on the resources availability. The main problem is that CAC does not control how and by whom are being allocated those connections. In a real-market environment, ATM service providers must define a proper connection admission policy. Since corporate customers may give different income to the ATM service providers, different credits should be established to each of them. A wise policy will obviously overcome the problems related to an undesirable share of network resources.

Another main trend when referring *Global business* is, obviously, the ATM accounting. As stated before, it is up to the ATM carriers to develop means to control and manage the charging of customers access.

O. Festor and A. Pras (Eds.) : 12th International Worshop on Distributed Systems: Operations and Management
DSOM'2001 Nancy France, October 15-17, 2001.

Means for metering, charging and billing should be enforced. A charging and billing policy becomes especially imperative, although may attain complex contours, when ATM connections are established by means of long-haul circuits. In this case, one ASP may not be capable of providing the necessary end-to-end QoS connection, since the destination address can be out of its scope. When a connection is, therefore, established crossing more than one ATM carrier domain, the charging and billing process must be well outlined.

As consequence of the pointed scenario, the potential ATM customers are facing serious difficulties when trying to select the most convenient ATM service provider to route their traffic. When multiple switched-based ATM WANs are available, a call between two distant parties may be established selecting one path within a great number of available paths, with resort to different ASPs resources. The decision of whatever path to select is, obviously, intrinsically dependent on the routing policy applied by the customer himself. Different metrics and attributes should be assigned to each path, establishing a wise compromise between the QoS offered and the costs charged by each ASP.

Since end-to-end service management across multi-vendor networks has received an enormous interest by ASPs, means to achieve interoperability between different management domains have to be deployed. In fact, the potential roles of software agents are gaining irrefutable importance in the telecommunications environment, namely in networks management (refer to [HB99]). Because the paradigm of network management relays essentially on its distributed nature, agents potentialities can be of great interest. The reduction of the communications overhead (derived by a higher level communication language established between agents), the improvement of flexibility and scalability (adding new agent roles and tasks as the customer claims better QoS), or the amelioration of ATM service provider response time with respect to the customer demands (by the retrieval of fundamental information in distant nodes of the network, for example), makes software agents especially suitable for our purposes.

In our opinion the pointed problems have to be solved by other means than ATM signalling, since the selection of a path (in response to a connection request) is not only dependent of the resources availability: it is also function of the costs charged. Furthermore, it is even possible that the pricing established by each ASP changes along the time. In this case, customers may want to reroute their traffic (that is being carried out by a set of connections) through a new ASP, the one that is offering lower prices. As soon as the decision of selecting a cheaper ASP is made, mechanisms to force traffic to be routed through the chosen ASP must be defined (rerouting the traffic from the existent connections to the new ones). Software agents are of great interest in this matter, since they can inform network nodes (namely the ingress switch, the one which is responsible of calculating the path to the destination address) to force the incoming calls to be routed in accordance with the customer selection.

Moreover, software agents can be of great usefulness to monitor circuits which cross different ATM domains. Agents can provide management information oriented to the customer needs in a much more meaningful way than the one provided by the indirect access to management variables made available by the ASPs. Usually the end-user just wants to be informed if the Service Level Agreement (SLA) is being satisfied or not. In the latter case, the customer should be notified in what terms the QoS is being affected in order to preview the impact in the service being used.

The aim to provide the user end-to-end quality service across network boundaries, instigate us for the development of an operation and management platform to use when multiple ATM domains are available. This integrated architecture allows an ATM end-user to select the best ATM service provider to route his traffic and guaranties that the establishment, maintenance and clearance of switched and soft permanent virtual connections (SVCs and soft PVCs) are successful across different ATM borders. The paradigm of the ATM call management, covering all the trends pointed before, is therefore the target of our agent-based architecture. It is our intent to achieve an integrated platform to deal with the establishment of QoS connections, possibly long-haul connections, across heterogeneous ATM WANs, while assuring that a corporate customer select a set of reliable ASPs to route his large amount of traffic.

The structure of this article is as follows. The next section describes the main trends and principles of software agents in telecommunication networks. Section 3 describes our agents architecture proposal for assuring ATM call management in the frame outlined above. Section 4 and 5 cover, respectively, the agent ATM tools and agent design and roles. Discussion and Conclusions are presented in section 6.

2 Software Agents in Network Management

In order to keep up the recent interest in software agents in the area of networks management, agent technology is, as already stated, also introduced.

Software agents can be seen as a three dimensional technology [dO98]: they exhibit characteristic properties, they communicate through a set of languages and they are designed following certain architectures. These three dimensions make the agents especially suited to substitute humans by pieces of software, which are based in agent technology. It is very common when referring to network control and management to find humans operators doing tasks that deal with users, or even companies, interaction.

The four most important agent properties - autonomy, pro-activeness, reactiveness and social-ability - are fundamental characteristics for applications working on behalf of the user (i.e. the customer) and the ATM service provider, in order to negotiate traffic contracts or find and filter management information in a complex distributed environment.

For software agents to interact and interoperate effectively, it is required three fundamental components: a common language, a common understanding of the knowledge exchanged and the ability to exchange whatever is defined within the context above [FLM97]. Therefore, in order to communicate, agents use different type of languages that allow them to exchange information matching the properties they have and the tasks they face. Agents use communication languages that let them to develop communicative acts of interrogative, declarative and permissive types, among others. The semantic of information passed in the communication acts are expressed through content languages. The base content languages can be extended depending on the particular ontologies associated to the content.

The internal agent architecture and the external multiagent architecture are usually based in three approaches: reactive, deliberative or hybrid. In any case, they must be adapted to the problem domain, especially in what concerns the reactive part which is responsible for the interface between the agents and the real environment.

In our case study, agent technology overcomes the trends of ATM heterogeneous management scenarios by providing means of establishing communication between nodes with different grades of knowledge; reducing operation and management data by the exchange of precise pieces of information; or detecting and reacting to changes in the state of the ATM networks [CCM99].

2.1 Communication and Content Languages

The utilisation of an agent communication language, such as Knowledge Query Manipulation Language (KQML) [LF97], requires a content language to represent the knowledge carried out by performatives (communicative acts). This content language can be, for instance, KIF [GF92] or another language that satisfies the requirement of ability for modelling knowledge.

The KQML agent communication language implicitly supposes a multiagent architecture based on regular and facilitator agents. Regular agents serve the problem domain, and the facilitator provide the infrastructure for regular agents to build easily the multiagent system. For example, facilitator agents accept the registration of regular agents, as well as information about the regular agents abilities and interests. Therefore, facilitators are paramount for regular agents getting easily integrated in the multiagent system.

Agent communication and content languages are an huge improvement in the way management applications communicate. When compared to protocols such SNMP, agent communication languages provide much more powerful communicative acts than the poor SNMP *get*, *getnext* and *set* primitives. Content languages, on their turn, provide for knowledge modelling, which supports the possibility for management applications to exchange information about networking services, instead of exchanging huge amounts of raw data (SNMP MIB variables). Furthermore, SNMP MIB variables usually need lot of processing before getting ready for human understanding.

3 Dynamic Call Management Plataform

When a corporate customer has multiple ATM transport services available, each offered by one ATM operator, a challenger decision is imposed: to choose the ATM carrier which submits a more profitable traffic

contract. This implies that the ATM customer is compelled to select the ATM carrier that offers best QoS connections (i.e. that meets or exceeds a set of specified QoS parameters), while assuring lower costs. As soon as the choice is made, the customer expects that the chosen ATM carrier routes his traffic, possibly to other corporate customer, accordingly to his selection. To succeed in this task, the ATM transport operator must establish a connection with the customers (the calling and called parties), through their local ATM operators.

The local ATM operator usually connects a set of private corporate networks. Additionally, it also connects these private corporate networks to a set of ATM transport carriers, that usually offer transport services by means of long-haul circuits. Still, it is also possible that the local ASP offers transport services too. In this case, although the called party may be connected to other local ATM operator (maybe sited in a different country), the local ASP of the calling party may submit a traffic proposal to route all of his traffic. To accomplish this job, the local ASP of the calling party must subcontract some ATM services to other ATM carrier, since the called party is out of his domain. Interoperability between different ATM networks is, therefore, always needed.

In our case, it is assumed that ATM service providers aim to offer their customers QoS end-to-end connections based on a flexible and scaleable routing architecture. In fact, PNNI [For96] is a routing and signalling protocol specified by the ATM Forum, which offers a hierarchical architecture approach allied with distribution of reachability information (i.e. topology discovery) and support of QoS parameters. PNNI was first introduced for interoperability within private ATM switches but it has been also receiving an increased interest by ATM public operators. More information can be found in [HHS98], [Bla98] and [Gin99].

3.1 Agents Architecture

This section describes the top-level design of the proposed agent plataform for control and management system. Our integrated approach relays on a two-layer architecture: the user agent and the ATM service provider agent.

The first step while defining an agent architecture, after the problem domain and the agency architecture analysis, is the definition of the agent roles and communication protocols. A role defines the characteristics or expected behaviour of an agent. In our case it is foreseen three different main classes of roles, one for each type of agent: the user agent (UA), the ATM Service Provider agent (ASPA) and the facilitator agent (FA). The UA is responsible for creating a friendly user interface to the corporate customer. It is its task to establish a traffic contract with all available ASPAs, and to choose the one that best suites the customer's needs. The ASP agents, on their turn, are responsible for ensuring interoperability between heterogeneous networks while collecting relevant management data (e.g. metering information) or monitoring circuits. There should be one ASPA for each ATM service domain available. The ASPAs must announce their capabilities to the facilitator agent, by means of an agent communication language. The FAs build their knowledge about the community via the KQML *register* and *advertise* performatives. An agent uses the *register* performative to announce its presence to the facilitator. In order to build the FA knowledge base, the ASPAs must use the *advertise* performative to announce what kind of ATM service requests they are willing to receive.

When a ASP is selected, a new communication channel to the destination address should be established. Since there may be different paths to the destination address, each of them possibly owned by different ASPs, the ingress switch (the one that receives the connection requests from that UA) must be told that the path to the destination address, which crosses the selected ASP borders, must be chosen. This is achieved attaining a higher priority to the path. The priority value is always considered whenever the Connection Admission Control (CAC) is launched. The CAC algorithm is responsible for accommodating new connections into the available network resources. In this case, if there is more than one path to the same destination address, the path that has a higher priority (and that has enough available bandwidth to satisfy the QoS specifications) is always chosen. This path should be the one owned by the selected ASP.

If the destination address is exterior reachable, i.e. the reachability advertisement within the PNNI domain includes information about accessibility of both exterior addresses and transit networks, the MIB objects *pnniRouteTnsTable* and *pnniMetricTable* should be accessed. These objects give some important information about the metrics and attributes assigned to transit networks. Since the path to the destina-

tion address, owned by the selected ASP, should have a higher level of desirability, a heavy administrative weight should be assigned to it. If the destination address is interior reachable, the PNNI MIB objects that need to be analysed are the *pnniRouteAddrTable* and the *pnniMetricTable*.

If the selected ASP does not want to receive further connection requests to that destination address (or to that destination address prefix) under the same profitable conditions, mechanisms to filter ATM addresses have to be deployed. These filtering mechanisms need also to be present if the ASP does not want to receive connections to that destination address from other source addresses, i.e. from other UAs, under the same profitable conditions. These pointed situations often happen if the ASP establishes different pricing lists to their clients, possibly depending on the income they give to the ATM operator. If the ASP establishes a nice price for one connection to one client, it does not mean that that price is still valid for other clients' connections. In order to surpass this situation, a filter could be used. ATM address filtering provides a way to control call set-ups thought SVCs. Some source or destinations addresses may be admitted or denied access to a port or set of ports in the ingress switch according to the ASP incoming connections policy.

3.2 Connection Request

Suppose, within the context of Figure 1, that two end-users, possibly sited in different countries, want to establish a communication channel through the creation of a SVC. This SVC is established in response of a request of the calling domain UA. Suppose also that each end-user is connected to his local ATM service provider, and that the local ASPs are connected to other ASP carriers (possibly international carriers). Due to the lack of signalling directives, there is no way to provide the calling party information about the path that would lead to the lower cost, except for the case where agents exist to provide this information. In our architecture, agents can overcome the pointed situation by giving precise management information to the ASP customer, in this case information related to the charged price-list.

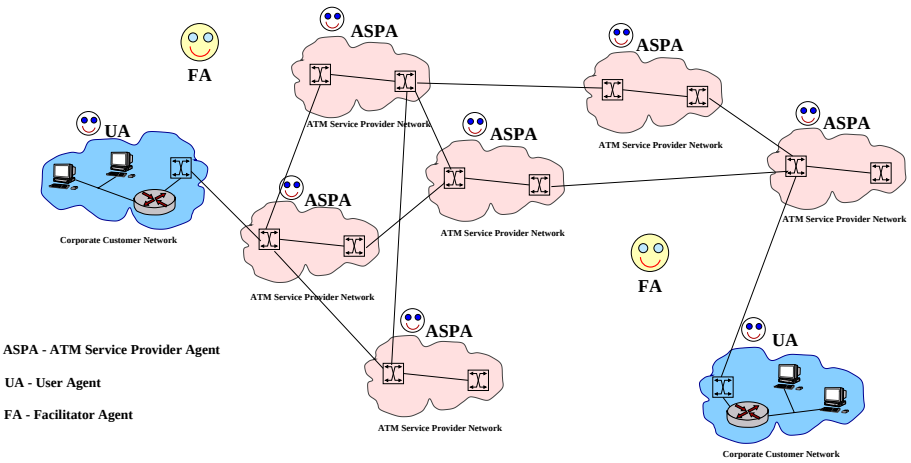

Fig. 1: Software agent roles in our ATM call management platform.

When a calling user intends to make a call to other distant user (the called party), he expects to obtain a complete list of ASPs that are available to establish the communication channel, as well as a set of details concerning the QoS parameters and prices charged for routing his traffic along the channel. In our case, the calling party is represented by the User Agent (UA), while the ATM carriers are represented by their ASP agents (ASPA). The UA agent is responsible for contacting the available ASPAs (representative of the local and non-local ASPs) in order to get a set of traffic contract proposals. The UA gets acquaintance with the available ASPAs by means of the facilitator agent (FA). All ASPAs that aim to offer ATM services must register themselves in the FA knowledge base. The ASPA, in his turn, is responsible for outlining a traffic proposal, after collecting networks information. In a first step , the ASPA must assure that the called party is accessible by the ASP means. Afterwards, the ASPA must investigate if the required resources are available

to meet or exceed the specified QoS parameters. And finally, depending on the results of the previous tasks, costs information must be collected. If the ASPA finds that there are no means to establish a connection channel, a denial message must be forwarded to the UA. If the ASPA finds that means are available, a traffic contract proposal is outlined. The local ASPA must, therefore, provide the adequate routing policy for the UA domain, so that the SVC is established using the selected ASP resources.

3.3 Release Request Caused by a Pricing Variation

Although our agent architecture can be of great interest in the situation pointed above (i.e. in a connection request scenario), other possible situations can be easily described, such as a release request scenario caused by a pricing variation.

Whenever a connection is in an open state, the ASP client may want to know if there are other ASPs that may route his traffic under more profitable conditions. These new ASPs may charge a lower price than the one it is being charged for the already existent connection. In this case, the user is interested in deviating his traffic, without interrupting the traffic flow, from a more expensive ASP to a cheaper ASP.

In the presence of the above scenario, the UA should contact the Facilitator Agent (FA) to get acquaintance with one or more ASPs that are able to offer cheaper conditions for the establishment of the connection. This connection should be established to the same destination address and with the same QoS parameters defined previously.

When the new path is already established, traffic should to be deviated from the older path to this new one. This procedure involves both the source and destination UAs because it is necessary to assign new VPI/VCI pairs (one pair at each extreme of the connection), while enabling the release of the older ones. This means that whenever a SVC is established, the source and the destination UAs must associate their new VPI/VCI pairs to their respective addresses, bringing down and up their interfaces.

Occasionally, during this set-up phase some cells may be discarded somewhere in the network if some buffers, existent in the older path, observe congestion. This situation can be solved with the retransmission of cells requested by the Transport Layer.

3.4 Monitoring Circuits Request

In case of a monitoring circuits request, the customer may want to know how, where and why his traffic flow is being affected in presence of variations in network resources availability. If the customer finds that some of the minimum QoS parameters are not being satisfied by the ASP which committed the traffic contract, mechanisms could be initiated to force a path calculation (initiating a connection request to other ASPs followed by a resources release request to free the old connection). Furthermore, it is also possible that a customer may want to be notified when and how a connection was released, possibly due to a network error. In fact, customers should know if a connection was terminated due to ATM service providers errors (e.g. congestion), in order to be charged less for that connection.

Although the monitoring circuits request functionally is not yet implemented in our agent-based architecture, it is expected to be deployed in the near future.

4 Agent ATM tools

The low-level implementation of our agent-based architecture is specified accordingly to the current status of the API for ATM-related system services under Linux [Alm96], based in a BDS-style socket interface. The API for ATM services under Linux supports the establishment of QoS SVCs, through the use of a *daemon* process, responsible for dealing with the necessary signalisation procedures. In fact, accordingly to [Alm96], the establishment of SVCs comprises four distinct phases: connection preparation, connection set-up, data exchange and, finally, connection teardown. SVCs are established to resort to SVC sockets.

Since the ASPA is intended to establish a SVC, after processing an up coming request from the UA, a user-space process should be launched in order to deal with the opening of that SVC. This can be achieved with another *daemon* process that controls the SVCs status, openings and teardowns, holding the socket file descriptor. In fact, whilst all opened VCs must have some process in the user-space holding them, the *daemon* is responsible for passing, afterwards, the file descriptor over a Unix domain socket to another

program. The deployed *daemon* is based in a client-server architecture. The *daemon* listens all the UA requests and invokes, afterwards, a set of procedures that aim to suit that same request: open, delete and list some SVC properties (VPI/VCI pair and QoS parameters). Two kinds of connections can be therefore identified: data and control links. Control links, established between different *daemons*, deal with the UA requests. Data Links are the result of control links.

When one UA demands for a new SVC, the *daemon* services, running in that same machine (calling *daemon*), are invoked. This calling *daemon* must contact the called *daemon*, running in the called UA machine, in order to test if the data link connection can be opened, following the pointed QoS parameters. If it does, the data link is established. Otherwise, the calling UA is notified with a denial of service.

In order to test connectivity parameters, two kind of functionalities were deployed. Conceptually identical to the Unix tool `ping`, two kind of atm pings were defined. The first tool returns the elapsed time from the SVC opening request to the effective SVC opening. The second one sends a set of requests through the SVC and waits for the respective answers, emitted by the called *daemon*. It is possible to specify a count and a waiting time (requests interval) parameter.

When the UA asks to deviate the ATM traffic from one SVC to another SVC, this one possible being established to resort to cheaper resources from another ATM Service Provider, the IP over ATM service must be invoked. Since the API for ATM-related system services under Linux supports IP over ATM, it is possible to assign an IP address to an opened SVC, forcing a new entry in the ATMARP table. When the new SVC is established, it is assigned the same IP address of the older SVC. As the traffic is always routed following the ATMARP table entries criteria (last entry, first choice), all traffic that was being carried out by the older SVC may now be transported by the newer one. The older SVC is, afterwards, cleared.

In order to establish a newer SVC, for the same ATM destiny address but with a different path, the PNNI administrative weight metric for the link that transverses the cheaper ATM service provider should be altered. Because the administrative weight is the fundamental metric in the PNNI routing protocol, links with a lower administrative weight are first selected. It is possible therefore to have two distinct SVCs for the same end system, each one having a distinct path. The administrative weight is set in the ingress switch by the UA application, to resort to the SVC *daemon*, by means of SNMP commands.

Since there is a clear lack of applications and network stacks that make use of ATM natively, one of the greatest strength of this control platform is that it can be used in any test case environment. In fact, although the above description reports to our agents model, these tools can be applied to other architectures.

5 Agents Interaction

This section describes some scenarios that explain the behaviour of each agent in our architecture, and clarify how software agents can add flexibility to the call management functionality in multi-operator ATM environments.

5.1 ATM Service Providers Offers

In a multi-operator environment, it is of paramount importance to discover which operator is in better position to provide ATM services. Consequently, the agents operating on behalf of the different ASPs must announce their presence to the closest FA. While announcing its presence, each ASPA must also inform the FA about the types of ATM service they will be able to serve. This is achieved by advertising the FA with one or more embedded messages templates. These messages will be of great interest in a near future to the UAs. In fact, during the phase where it is asked, by the UA, if it is possible of open a virtual circuit (like a SVC) to a certain destination, satisfying a number of features (written down in a service specification using an appropriate content language), a prior advertisement of the ASPAs is a key point. The advertisement phase is done by means of the performative *advertise(ask-if(ServiceReq))*, where *ServiceReq* is the service specification. If more freedom is used in the service specifications (e.g. message templates, using variable content languages), more possibilities will be allowed in UA service specifications. The UA service specifications are done accordingly to customers QoS and costs requirements. The advertisement performative is presented, along with other performatives, in figure 2.

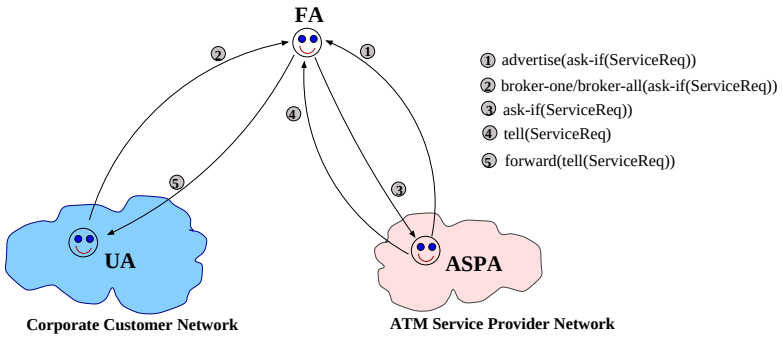

Fig. 2: Facilitator agent, user agent and ASP agents interaction.

Whenever the UA, representing a customer network, needs to open a virtual circuit to a remote network, in order to satisfy local users, a selection of a set of operator networks, through which the circuit can be opened, must be made. Selecting a set of operators means the selection of an unique ASPA, that is responsible for the circuit control end-to-end, possibly resorting to other ASPAs subcontracts. Since the UA does not have to know all the existent ASPAs, it simply sends a *broker-one* (or *broker-all*) performative to the FA, which will try to find if the embedded message in the *broker-one* matches any embedded message sent previously by an ASPA (inside an *advertise* performative). If The FA finds a matching message, the ASPA will be inquired. The ASPA's answer will be, afterwards, forwarded to the UA by the FA. By that time the UA becomes aware of one ASPA than is able to serve him, according to the service specification present in the answer. This service specification is not mandatory to be equal to the specification originally sent. Figure 2 shows the sequence of all the performatives exchanged by the agents comprised during these steps.

5.2 Connection Request Contract

As described earlier, in presence of all the ASPAs proposals, the UA must select which are those which are offering the best services. As soon as this decision is made, the UA must establish a contract with the selected ASPA.

Since the list of ASPA discovered in the scenario described in 5.1 might have been altered, the UA should confirm if the selected ASPA has still conditions to satisfy its requisites. This step is achieved by sending directly to the selected ASPA an *ask-if* performative. In case of a positive answer, the UA sends a request to the ASPA in order to establish a connection request contract. This is done to resort to an *achieve* performative together with a Service Level Agreement specification. The ASPA answers back with a *tell* performative with the same SLA embedded, which means the SLA has been accepted. At that time the UA can open a circuit using ATM signalling. It is the selected ASPA responsibility to guarantee that the ATM route uses network resources belonging to the ASPA itself.

When the UA no longer needs the virtual circuit, the ASPA is informed through an *unachieve* performative and the circuit is closed by means of ATM signalling mechanisms. The ASPA returns an *untell* performative stating that the SLA is no longer valid. The complete life cycle of a connection request contract is illustrated in figure 3.

6 Conclusions

In this article a multi-agent architecture for the management of connections crossing different ATM networks is addressed. This call management platform aims to contour some existent limitations due to the lack of signalling directives in the frame of heterogeneous ATM service providers networks, namely in what concerns pricing.

Although we are still on a designing and experimenting phase, it is foreseen that software agents potentialities are very adequate to solve some call management paradigms, while providing means for scalability.

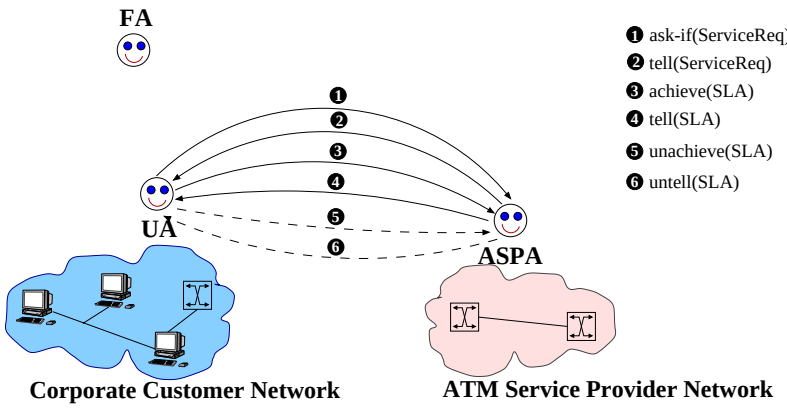

Fig. 3: Connection request contract.

Since it is considered both the ATM service provider and the customer interests, concerning a large set of management information issues, we expect to get really interesting results.

Our future work will concentrate on the enhancement of the agents content language, adding new skills as new factors emerge. Additionally, a further investigation has to be done concerning messaging and computational efficiency in the context of our distributed management system.

References

[Alm96] Werner Almesberger. Linux ATM API, Julho 1996. ftp://lrcftp.epfl.ch/pub/linux/atm/api.

[Bla98] Uyless Black. *Signalling in Broadband Networks*, volume II. Prentice-Hall, 1998.

[CCM99] Morsy M. Cheikrouhou, Pierre Conti, and Karina Markus. Software Agents for Network Management: Case Studies and Experienced-Gained. Technical report, Institut Eurécom, 1999.

[dO98] Raúl Teixeira de Oliveira. *Managing Awareness in Networks Throught Software Agents*. PhD thesis, Ecole Nationale Superieure De Telecomunications, 1998.

[FLM97] Tim Finin, Yannis Labrou, and James Mayfield. KQML as an Agent Communication Language. In *Software Agents*, chapter 14, pages 291–316. AAAI Press/The MIT Press, 1997.

[For96] ATM Forum. The ATM Forum Private Network-Network Interface Specification Version 1.0 (PNNI 1.0), Março 1996. af-pnni-0055.000.

[GF92] Michael R. Genesereth and Richard E. Fikes. *Knowledge Interchange Format, Version 3.0, Reference Manual*. Logic Group, Computer Science Department,, Stanford University, January 1992.

[Gin99] David Ginsburg. *ATM Solutions for Enterprise Internetworking*. Addison-Wesley, 1999.

[HB99] Alex L.G. Hayzelden and John Bigham. Agent Technology in Communications Systems: An overview. *Knowldege Engineering Review*, 14(4), 1999.

[HHS98] Rainer Handel, Manfred Huber, and Stefan Schroder. *ATM Networks: Concepts, Protocols and Applications*. Addison-Wesley, 1998.

[LF97] Yannis Labrou and Tim Finin. A Proposal for a new KQML Specification. Technical report, Computer Science and Electrical Engineering Department(CSEE), University of Maryland Baltimore County(UMBC), February 1997.

[SvdWvBea99] R. Sprenkels, B. van der Waaij, and B.J. van Beijnum et al. The Feasibility of Introducing ATM SVCs. In *Proceedings of the European Conference on Networks and Optical Communications*, 1999.