QoS management in programmable networks through mobile agents

Aurelio La Corte(1), Antonio Puliafito(2), Orazio Tomarchio(1)

(1) Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, University of Catania
Viale Andrea Doria 6, 95125 Catania, Italy
{lacorte, toma}c@unitn.it

(2) Dipartimento di Matematica, University of Messina
Contrada Papardo, 98166 Messina, Italy
apulia@ingegneria.unime.it

Abstract—In delivering multimedia services, quality of service represents a crucial commitment to be satisfied. Very often it has been considered only from a theoretical point of view, leaving any implementation details out of the discussion, mainly for the lack of concrete possibilities to execute its control and management effectively. Recent technological developments in the networking and distributed programming fields are now opening new challenging scenarios towards the negotiation and guarantee of QoS in the delivery of multimedia services through the network. Active or programmable networks are becoming a reality, and the migration of software components among network nodes seems to be the direction pursued by most of network manufacturers. Mobile software agents represent a very attractive approach to the distributed control of computer networks and a valid alternative to the implementation of strategies for the management of QoS. In this paper we present our approach to QoS management through mobile agents. The potentiality of this approach is shown through two application examples. The first one focuses on resource reservation through RSVP in an int-serv scenario, while the second one shows how to provide QoS to aggregated traffic flowing through a virtual network.

Keywords—QoS, mobile agents, RSVP, int-serv and diff-serv model

I. INTRODUCTION

Recent technological developments in the networking field are rapidly converging towards the integration of data, voice and video traffic, the coexistence of ATM and IP, the integration of wired and wireless services, with the final target of delivering new and sophisticated broadband services to the users. This process is strongly due to the evolution of multimedia applications that pose new and challenging problems related to the prediction, guarantee and adaptation of quality of service (QoS). Such concept is not new, although only recently a concrete contribution in such direction has been done [1], [26]. Technology is nowadays mature enough to really pursue the objective to negotiate and guarantee QoS in the delivery of services through the network, eventually adapting to a different level, if unpredictable events occur. But a lot of challenging work for technicians and researchers has still to be done in order to reach interoperability among different protocols, capacity planning, service adaptability, QoS provision for different applications among different networks [26]. We do think that the new emerging technology of agent programming is a challenging opportunity for delivering more flexible services and dealing with network programmability. We think that distributing intelligence across the network allows the fast exploitation of more advanced services that can dynamically adapt to the user’s requirements. This also implies a strong integration of the user and the network perspective. User requirements have to be automatically translated into network requirements, and this implicitly assumes the possibility to interact with the network equipment. Programmable networks are then the direction to go, although the technology is still in its infancy, and a lot of technical problems have to be solved. Which are the drawbacks of opening network equipment to software coming from the network? Do network providers have any convenience in doing so? How to guarantee adequate security levels? These and many other typical questions that arise when discussing about the opportunity to have programmable networks [22], [21], [2]. Nevertheless, ad hoc software sent across the network may allow to use network resources more effectively, as they can be directly controlled at the application level. End-to-end QoS offers the possibility to involve the user in the process to manage the QoS, allowing with the possibility to adapt the management strategy to his/her requirements.

In this paper we tackle the problem of programmable networks, investigating on the definition of a network architecture based on a distributed programming environment that allows QoS management at the IP level. We adopt the agent programming paradigm [9], [23], [20] to develop such architecture and prove its flexibility in operating resource reservation in a flow-based situation (int-serv model [24]) and in aggregated traffic condition through virtual links (diff-serv model [25]). We assume the RSVP protocol [12] to be available at least in a subset of the network equipment, and investigate the following two scenarios: the first one is related with the problem of operating resource reservation in an int-serv model when some of the routers along the path to the network’s support the RSVP protocol; the second scenario focuses on the diff-serv model, and shows how to provide QoS to aggregated traffic flowing through a virtual network. In both case studies, we adopt the mobile agents paradigm to exploit network programmability and to implement the described services. We show the flexibility of such approach and its great potentialities that - we think - should push network equipment developers, network and service providers to open their systems to allow for the diffusion of more advanced services that can be deployed from the adoption of this new programming paradigm.

The rest of the paper is organized as follows. An analysis of QoS in the Internet environment is given in section 2. Section 3 describes the reference architecture for agent-based QoS management. Section 4 presents the results we have obtained in dealing with QoS provisioning when both int-serv and diff-serv models are assumed. Section 5 describes some implementation issues, and finally we conclude in section 6.
II. QoS and customized services by means of mobile agents

The current Internet technology is not able to guarantee a satisfactory level of quality for multimedia services. It is based on a protocol layer (IP) that allows a certain level of resource sharing and multiplexing, but implies best-effort quality. When network resources are widely used, variations of end-to-end delay and throughput may occur, which are not acceptable for multimedia services [1]. The requirements of service providers - who wish to exploit the best network resources in order to minimize costs - and the ones of the user - who wants to use multimedia applications effectively, are often in antithesis.

Reservation of resources and/or adaptation of applications are two possible strategies to be adopted in order to ensure adequate QoS levels. At the IP networking layer, the IETF standardization body has proposed the flow-based (state-aware) resource reservation approach (int-serv model [24]). Such approach introduces flexibility but experiences scalability problems. Differentiated service (diff-serv model [25]) classes do not rely on any flow-based state representation, but concerns remain about the scalability and complexity of the signalling procedure involved [29].

Considering that flow-based reservation schemes such as RSVP/intserv, or ATM for that matter, do not scale well in core networks, the issue is whether to develop QoS control mechanisms at the application level or to allow the customer to implement his/her own strategies for QoS management. For example, some customers, who could represent larger organizations, may wish to rely on guaranteed services provided by the intserv model (willing to pay for all implied resources), or may be satisfied with diffserv model, accepting occasional quality disturbances, or may even prefer to implement a type of service of their own that has not been standardized at all. [3], [2], [4].

In order to provide each customer with the opportunity of implementing his/her own policy, trading costs with service quality according to his/her own preferences and capacities, network service must be open, flexible and configurable upon demand. A first step towards the realization of a programmable network is that of using an agent-based approach, in order to obtain a faster time scale of service deployment.

A major incentive for an agent-based approach is that policies can be implemented dynamically, allowing for a resource-state-based admission and reservation strategy. Agents are used to discover about resources available inside the network and claim resources on behalf of customers according to some "figures of merit" [6] which represent trade-offs between bandwidth claimed and loss risk incurred due to high utilization.

Different customers may pay for resources in a different way, negotiating the costs for obtaining a certain "figure of merit". Agents are able to trigger adaptation of applications inside the network on behalf of customers. This allows for an immediate response to resource shortages, decreases the amount of useless data transported, and reduces signaling overhead.

Mobile agents [7], [9], [23] provide the highest possible degree of flexibility and can carry application specific knowledge into the network to locations where it is needed, following an approach similar to the one shown in some other programmable network projects [10], [11], [2].

III. The agent-based system architecture

For developing a distributed system for QoS management, a system architecture has to be defined, where mobile agents can communicate and cooperate (Figure 1). The architecture has been implemented using a software package for the development and management of mobile agents, called MAP\(^1\), which has been developed at the University of Catania. Further details on this platform are provided in [5], [28].

By using a distributed approach, some advantages can be obtained, such as less complexity for each agent, a better knowledge of the local situations, and thus a better discovery of the system’s problems, the possibility of taking different control actions according to the specific knowledge and capacity of the agents involved in the operation.

The architecture defines four types of agents, representing different levels of expertise: User Agent, System Agent, Network Agent, Service Agent. A user on the client system interacts with the QoS management component by means of an appropriate User Agent: such module is specific for each service (application) required by the user, and enables him/her to specify and negotiate the desired QoS parameters.

A System Agent resides on each client and provides all the information about the client’s system state. Relying on the services provided by the operating system of the host machine, it has the mechanisms to control the QoS on the client. This agent is the reference module within the client: both the User Agent and the Network Agents, which will be described below, rely on it for various operations. From an implementation point of view, the logical functionalities exploited by the System Agent may be seen as independent processes (or agents) concurrently executing on the node. It supports mapping of QoS parameters from one level to another, reservation of hw/sw resources inside the client, and the negotiation of QoS parameters. In addition, it deals with the communication and co-operation with the other agents of the system. Attached to the network are Network Agents, whose goal is to control the network; such agents provide the possibility of specifying a set of QoS parameters for a given connection, reserving suitable network resources, monitoring QoS parameters during existence of a connection, starting the corrective activities of adaptation in case some QoS parameters are not respected, and interaction with the network management system in order to obtain new resources and/or optimize

\(^1\) MAP is available at http://sun195.iit.unict.it/MAP
the use of the existing ones. A Service Agent is available on each application server node, and has to monitor whether the QoS parameters (to be provided for each service required by the clients) are respected, permit the use of a service to the optimal number of users by organizing an efficient use of the available resources, keep track of the users logged in and their occupied resources, in order to optimize the sharing of the network resources.

User Agents play a particular role. They are considered greedy to a certain extent as to get the service quality which suits them best. Therefore, user agents are designed to compete with the other type of agents. Opposed to this, the agents of other types are supposed to cooperate with each other supportively. But note that once negotiation has been completed, all agents must respect their own commitments for the whole process, in order to work well. A key point of our architecture is built on the possibility of downloading (from the server) the User Agent related to the application to be run. QoS parameters at the user level depend on the type of application (and thus on the service required). The provider can therefore include (in an agent) the information related with the most appropriate combinations of QoS parameters for the service required. This way, the server can meet the users’ requirements better, by optimizing, at the same time, the use of network resources.

A. Cooperation and communication among agents

The use of an agent-based infrastructure for the management of QoS is a good choice for implementing several adaptation strategies in which available resources need to be re-arranged in such a way that the user is still satisfied. Such operations, in general, imply an intense (and sometimes complex) exchange of messages among the parties involved. Agents can therefore adequately support such functions and permit more flexibility in doing such operations.

As we can see in Figure 2, our architecture offers two kinds of interactions among agents: (1) horizontal communication between agents of corresponding levels, (2) vertical communication between agents of adjacent levels. Horizontal communication involves negotiation of QoS parameters specific to the corresponding level. For example, the User Agent and the Service Agent negotiate application level QoS parameters, while System Agents operate on network QoS parameters. Requests for setting-up a connection with a given QoS occur among agents of adjacent levels, thus involving vertical communications. The agent of each level is in charge of translating its parameters into the ones corresponding to the adjacent level. At the user’s level, typical parameters to specify the required QoS are related to the description of the reproduction quality at the device level, for example sample size (resulting from height, width and color specification for a video stream) and the media sample rate. The user generally has no specific knowledge of the underlying technological constraints. The User Agent will therefore make a preliminary conversion, sending a set of values to the System Agent, representing the level of QoS desired by the user. After starting the application and establishing the connection via the network, System Agents together with Network Agents have to monitor the connection state, in order to assure that the negotiated parameters are respected. If some of these parameters do not match with the negotiated values, and if an adaptation phase cannot establish suitable QoS parameters, a renegotiation phase is started.

As shown in Figure 2, a System Agent always instantiates a Network Interface Agent that deals with the negotiation of QoS parameters. Through this agent, other monitoring and control activities on the network are started: in fact, this agent communicates with Network Agents on the network nodes. Network Agents refer to the Network Interface Agent when they need to notify critical situations or problems on the network part that affects the node in question.

IV. QoS MANAGEMENT IN IP NETWORKS THROUGH MOBILE AGENTS

In this section we propose two case studies where mobile software agents have been used to tackle the problem of QoS management in IP networks. The first application, which has been detailed in [27], shows how RSVP functionalities can be augmented through the use of mobile agents technology to manage the unavoidable situation where such protocol suite is not installed in some routers along a given path. The second case study, which reports the main results shown in [6], goes further the management of the single flow of information (as RSVP implicitly does), and describes how to benefit from mobile agents technology to manage QoS in aggregated traffic flows. Such case studies are not mutually exclusive. Actually, they complement each other in current network scenarios where Intranet environments are interconnected through the Internet. Int-serv model (which operates on a per-flow basis) can then be used inside the Intranet, while diff-serv model (which assumes aggregated flows) can be adopted in transferring data through the Internet.

A. Complementing the RSVP through mobile agents

RSVP is a framework for resource reservation and QoS provisioning mechanisms within the Internet. The RSVP protocol defines the exchange of control messages to build up and maintain a shared knowledge of the reservation state [12]. In order to allow the correct operation of the protocol, all the routers along the path between source and destination have to be RSVP-capable routers\(^2\). In fact, the RSVP protocol simply ignores

\(^2\)We will refer to RSVP-capable routers and non-RSVP-capable routers as RSVP routers and non-RSVP routers, respectively, in the sequel.
non-RSVP routers. RSVP messages are forwarded through a cloud of non-RSVP routers and reservation is merely performed outside the cloud. However, since non-RSVP routers may degrade the perceived QoS in an uncontrolled way, no end-to-end guarantee can be given. Although the guarantee of having only RSVP routers along a given path can be hardly met, at least in the near or mid future, as confirmed in [12]. It is still believed that the application of RSVP on the remaining part of the path is beneficial, because the performance of the non-RSVP routers may be sufficient to fulfill the demanded level of QoS. We have defined the set of routers included in the sub-path within a cloud of non-RSVP routers to be a tunnel; a single router in the tunnel is referred to as a tunnel router. We are going to improve the end-to-end quality by monitoring the tunnel and providing feedback for an enhancement of the reservation scheme. For such purpose, distributed mechanisms are needed for implementing cooperative monitoring techniques, in order to:

- monitor the behavior of the system in terms of the performance level it provides,
- use information on the status of the system for establishing a QoS negotiation phase and for proper distribution of application requirements on system resources,
- interact with the reservation entities within the RSVP routers.

The basic idea of the approach consists in exploiting the interaction among properly defined software entities continuously running inside the system in order to implement QoS monitoring, reservation, and adaptation functionalities [14]. Their primary purpose is monitoring tunnel properties in order to interact with the RSVP routers for assuring the end-to-end QoS to the user. The reference scenario is shown in Figure 3.

![Fig. 3. Tunnels](image)

Two particular network agents, in the following referred to as the tunnel agents, are associated to each tunnel. More specifically, tunnel agents are located on the RSVP nodes at the edges of each tunnel. The RSVP nodes at the edges of a tunnel will be referred to as edge routers in the sequel. We assume that each RSVP node is able to execute our agents. Furthermore, all nodes - including non-RSVP nodes - are supposed to support standard network management protocols like SNMP [15] to provide some means for monitoring. In an extreme scenario, none of the routers supports RSVP, and thus the tunnel spans over the entire path. In this case, the tunnel agents are located on the application hosts, and provide the application with useful performance monitoring data to allow for adaptation of traffic volume and encoding. The presence of tunnel agents running on edge routers is not punishing, although the edge routers are already performing RSVP functionalities. The events managed by tunnel agents are less frequent than the continuous packets forwarding activity performed by a router.

The code of the agents is maintained in a generic node that plays the role of a server of QoS applications. Once a tunnel is identified, the user agent located on the receiving application node sends a request to the nearest QoS application server to upload the tunnel agent code, which is then dynamically executed (for further details on these mechanisms, see [27]). The main advantage of such approach consists of avoiding installation of the code on each node: the location of a tunnel depends on dynamic factors like the existence and location of the session path, as well as on more static properties like the RSVP capability of the nodes. It may even happen that a node will never be at the edge of a tunnel. A tunnel agent installed on such a node would not be useful, and would only consume node resources. The existence of simple mechanisms for code retrieval and downloading is thus a useful feature for our specific problem. Below we focus on the description of how tunnel agents migrate as a consequence of a re-routing in the Internet. Such circumstance shows how to take advantage from the mobility capability of software agents between routers. We refer to [27] for details about the setup procedures and the interaction strategy among tunnels and RSVP.

![Fig. 4. Agent migration after route change](image)
A. Routers. To specify the role of the edge routers better, we refer to traffic, namely to establish virtual links among different edges. Communication at the Internet level requires to play with aggregated resources, and also the number of users is under control. The community, as depicted in figure 5, assumes a set of routers that allow to establish virtual links (VLs) among different customers. A VL groups physical connections between users belonging to different customers. A set of VLs is what is nowadays referred to as a virtual network.

B. Using Mobile Agents to deal with Differentiated Services

One of the main drawbacks of the RSVP protocol is its limited scalability due to the per-flow reservation strategy adopted. However, resources are usually claimed on a much coarser scale than flow-based resource reservations would do. Therefore, we are inclined to talk about virtual links and virtual networks that are customized on behalf of customers. Granularity of time and space for resource reservation through agents are then performed on a much coarser scale than with flows. Our proposal is based on the use of a set of specialized agents that dynamically decide the amount of resources to be assigned to each virtual link and use the RSVP signaling mechanisms for allocating the requested bandwidth [6].

The reference scenario, as depicted in figure 5, assumes a set of customers each consisting of users wishing to exchange data. A customer can be a company, a university campus, a public or private entity that groups various users. Each customer accesses the core network through an edge router, which represents the access point (for ingress and egress traffic) to the external world. The core network is made of a set of routers that allow to establish virtual links (VLs) among different customers. A VL groups physical connections between users belonging to different customers. A set of VLs is what is nowadays referred to as a virtual programmable network.

At the Intranet level, bandwidth reservation may be done through RSVP: the extension of the network is limited enough and also the number of users is under control. The communication at the Internet level requires to play with aggregated traffic, namely to establish virtual links among different edges routers. To specify the role of the edge routers better, we refer to Figure 6, which shows the software module to be executed on them: we assume router A as the originator of a request for VL creation, and router B as the receiver of the request. The edge router A hosts the coordinator and the collector agents. The former receives the opening requests of a new VL from the customer and creates a collector agent for each request, that manages the request itself. The latter is the manager of the VL, and, as soon as it is created, it sends a message of opening request to the remote edge router (B), and waits for the arrival of the collector agent.

The router B hosts the coordinator, collector and collector agents. As soon as the coordinator receives a 'request for VL creation', a new collector agent is created in charge of managing the VL which contains the identifier of the VL and the identifier (unique) of the agent associated with the same VL but executing on the remote edge router (A). The collector also periodically creates a collector agents with the task of traveling through all the routers between the two edge routers (A and B) to collect useful information on the state of the network.
take place only once, but is periodically executed. Besides, for avoiding problems of inconsistency, the collector agents are programmed to kill their execution if an open (but not reserved) VL already exists in a router to which they migrate. This way, we avoid to collect potentially unreliable information as a consequence that the existing VL is still being set up and the data related to the status are being modified.

The algorithm executed by the clconnect agent in order to implement the reservation policy consists of two distinct phases. In the first phase (for example according to [12], or according to the quality a customer wishes to assure to its users) the clconnect agent first tries to reserve all the bandwidth originally required by the application. If there are not enough resources available, it checks whether an under-reservation can be executed according to some figures of merit that will depend on the network parameters (speed, latency, technologies) and also on some other parameters such as a user’s willingness to pay for obtaining a high and guaranteed QoS. More specifically, the idea is to check the feasibility of providing a lower amount of reserved bandwidth than the one originally needed, as long as the performance (e.g. average loss ratio) is maintained within acceptable values.

If the resources of the network are still not adequate for meeting the request of the under-reserved connection, the second phase of the algorithm is started. The clconnect agent requires the clconnect agents managing other VL to give up a part of the resources used. This requires a strong coordination action with the clconnect agents managing the VL already established. In fact, the clconnect agent builds (according to the information communicated by the collector agent) a list of the routers that do not have the resources needed for creating the VL. Each router of this list is then associated to a record that stores all the reserved VL that crosses the router itself. For each router of the list, the clconnect agent sends a resource request message to the clconnect agents related with the reserved VL crossing the router, asking to reduce the amount of bandwidth used. If it succeeds in collecting the necessary bandwidth, the router is removed from the list and the same procedure is repeated on the subsequent router, if any. If the list is empty, it indicates that the agent has already obtained all the necessary resources, and may create the under-reserved VL. Conversely, if even a single router on the path has no VL to which the clconnect could require resources, or the sum of those given up by the other agents is not sufficient, the VL creation is refused and the QoS required by the customer has to be renegotiated.

Once more, the problem examined shows how agents are paving the way through the concrete exploitation of programmable networks. Agents mobility represents an extra feature that introduces more flexibility in specifying and changing the algorithms executed on the network nodes. Network nodes are very likely to become pure hardware equipment that will execute software code according to the needs of the user, namely a company, a university campus, a public or private entity that groups various users. We are aware of evident difficulties in proceeding through such direction (one of the more important is the maintenance of adequate security levels), but we do think that the benefits deriving from opening the networks will be equally divided among service, network providers and final users.

V. IMPLEMENTATION ISSUES

In this section, some implementation details will be provided with regard to the Java-based agent system used for the applications described so far. The mobile agent system MAP [5] provides all the basic tools for the creation and the management of agents, and for their migration and communication. In fact, the platform enables to create, run, suspend, wake up, deactivate and reactivate agents, to stop their execution, and to make them communicate and migrate through the network. The MAP is also equipped with a simple graphical interface that facilitates the access to the above mentioned management functions. In order to reach interoperability, in its latest release the MAP platform has been made compliant with the MASIF specification [19], [17]; this way, each MAP platform can accept agents coming from different platforms (also complying with MASIF), and can make them run, allowing them to access the methods needed for their management. Besides, the same way, a MAP agent is allowed to migrate to other platforms that can support it, and can run there. According to the MASIF specifications, the different agent servers are grouped in “regions”, where an appropriate entity (Region Registrar) maintains the information concerning the agents and the agent systems present. Figure 7 shows the logical organization of the different MAP servers.

We can easily map these regions with the different administrative domains that are present when working in a different model, enabling each network administrator to use his/her preferred agent execution environment, provided its compatibility with the MASIF specification.

Security is the other very important aspect - present in MAP - for the development of useful agent-based applications. Nobody would accept a foreign agent, allowing it to set or get network parameters without a strong security infrastructure that could assure who sent the agent and the kind of action it is allowed to do. An adequate security model has been implemented in MAP [18], which considers the issues of authentication and authorization, in order to protect - on one hand - the hosts from the agents, and - on the other hand - the agents from the other agents and from the attacks coming from the network. Security policies can be defined for specifying the conditions according to which the agents can access host resources. The techniques used are based on public key encryption, in which each user has a pair of public and private keys, with which he/she can prove his/her identity.

A. Agent Execution Environment

In this section, we explicitly focus on the software environment that has to be present on a router to allow agents execution. For the sake of clarity, we also provide a full description of the reference architecture of our routers. Figure 8 shows the
logical organization of a router’s main functional entities. With Routing Functionalities we indicate the basic mechanisms for packets routing, which depend on the router itself. They can be implemented directly in hardware or emulated through adequate software. In the current state of implementation, routing functionalities are executed through a software approach; we use simple PCs equipped with multiple network cards configured as routers, running Unix FreeBSD.

An SNMP agent is required to monitor and retrieve information on the status of the router and on the different connections that go through the router. The RSVP module provides the router with the capability of eventually using this protocol to communicate with other routers for resource reservation purposes. RSVP version 1, as implemented by the ISI-rsvpd Release 4.2a1, and the packet scheduler mechanism, provided by AL TQ version 0.4.1, are currently used. Finally, an Agent Server is also instantiated on a router that represents the execution environment for the agents.

![Software modules needed on a router](image)

**VI. CONCLUSIONS**

In this paper we have focused on the management of QoS through the use of mobile software agents. We have presented a reference architecture that allows for the distributed control of a networking environment where network components and the user interact each other in order to pursue predefined levels of quality of service. We have used the MAP system (available at [http://sun195.iit.unict.it/MAP](http://sun195.iit.unict.it/MAP)) as a mobile software platform; on its top we have developed two applications. The first one focuses on resource reservation through RSVP in an int-serv scenario, while the second one shows how to provide QoS to aggregated traffic flowing through a virtual network. Both examples show a concrete way to execute QoS management through software agents, and highlight some requirements to be met in order to make such approach really feasible. Agent mobility is an extra feature that allows for more flexibility and for the development of more sophisticated and powerful management applications. Security guarantee (although not covered in this paper) is a crucial issue that has to be provided by the agent execution environment. Network programmability (that implies the openness of network equipment) should be strongly encouraged, although network and service providers might have some reasons to obstacle such process: both of them can really benefit from opening their systems to allow for the diffusion of more advanced services which can be accessed in a very flexible way according to the user requirements.

**REFERENCES**


---

**Fig. 8. Software modules needed on a router**