Available online at www.sciencedirect.com
SCIENCE DIRECT®
Microprocessors and Microsystems 27 (2003) 73–83
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

Active middleware for Internet Video on Demand: the QoS-aware routing solution in ubiQoS

Paolo Bellavista*, Antonio Corradi

Dip. Elettronica, Informatica e Sistemistica, Università di Bologna, Viale Risorgimento 2, Bologna 40136, Italy

Received 24 November 2002; accepted 25 November 2002

Abstract

Several factors are forcing to address the issues of differentiated Quality of Service (QoS) and ubiquitous accessibility in Internet services, from growing user requirements to the increasing heterogeneity of access devices, from the competition of service providers to the severe constraints of resource availability in emerging wireless environments. The paper claims that the provision of services with negotiated and controlled QoS over best-effort networks is achievable via distributed support infrastructures that activate some of the nodes along the network path between clients and servers. The paper proposes mobile agents (MAs) as the activation technology to implement the needed active infrastructure and the MA-based design and implementation of the ubiQoS middleware for Video on Demand. At the starting of the service session, ubiQoS establishes an active path of intermediate nodes capable of tailoring multimedia flow QoS depending on profiles of user preferences and of device characteristics; at provision time, ubiQoS monitors the offered quality and promptly react to changes in resource availability by locally performing multimedia transcoding/downscaling and resource pre-emption. In particular, the paper focuses on how it is possible to determine dynamically a QoS-aware active path between clients and servers, discusses alternative solutions, and evaluates the performance results of the completely decentralized Peer-to-Peer implementation of the active path determination in ubiQoS.

Keywords: Quality of Service; Middleware; Video on Demand; Active Services; Mobile agents

1. Introduction

A constantly increasing number of users tend to access Internet services from ubiquitous points of attachment via a widening range of heterogeneous devices. Users tend to require differentiated and tailored Quality of Service (QoS), based on personal preferences and classes of usage, by considering also accounting aspects such as business/economic/free-of-charge QoS. The diffusion of wireless communications and of mobile access to the Web [1] widens further the heterogeneity of Internet client devices. Terminals span from traditional workstations and PCs, to laptops, personal assistants and smart phones, with continuous/intermittent ubiquitous connectivity.

Both service providers and network operators are calling for technologies, mechanisms, and tools to support Internet services with differentiated QoS, and to record, control and grant the QoS level provided at runtime. Several research efforts have recently investigated ad hoc protocols at the network layer [2,3]. These solutions achieved interesting results for limited networks, but tend to clash with the best-effort Internet model. In addition, they require that routers traversed by service flows implement specific ad hoc protocols. This constraint is likely to produce a long process of acceptance and diffusion. As a general consideration, network-layer solutions work at a level where it is difficult to embed some of the functions required in QoS-enabled service provisioning, such as application-specific adaptation, secure billing and non-repudiable accounting [4].

Some recent work has pointed out the suitability of distributed infrastructures where some intermediate nodes play an active role along the network path between clients and servers [5,6]. Service provision involves not only a coordinated set of server hosts and not only clients capable of proposing profile information (user preferences and device properties) and of enhancing service interactivity by offering local execution resources, as in the case of Java applets. Internet services should also activate intermediate nodes for QoS-enabled service provisioning by operating on

* Corresponding author. Tel.: +39-051-2093866; fax: +39-051-2093073.
E-mail address: pbellavista@deis.unibo.it (P. Bellavista).

0141-933/03/$ - see front matter © 2003 Elsevier Science B.V. All rights reserved.
PII: S0141-9331(02)00099-6

traversing data flows and reserving intermediate network resources. For instance, intermediate nodes should offer their storage resources to realize distributed caches of popular Video on Demand (VoD) contents for clients and intermediate nodes within their locality, thus permitting to decrease the generated network traffic and the client-perceived latency. In addition, the participation of intermediate nodes can achieve scalability and complete decentralization, crucial requirements for service provision and management in the open and global Internet environment [4]. Scalability imposes management decisions locally to the involved resources and autonomous adaptation/recovery operations on service components when and where there are changes in the availability of involved network resources.

Mobile agents (MAs) emerge as a middleware technology suitable to develop and deploy active services [6]. MAs can exploit code mobility to reallocate on the nodes of the distribution paths, thus allowing the needed dynamic deployment of middleware components. MAs can monitor/control network resources locally and autonomously, and can perform prompt management operations at the dynamically determined critical points of the network infrastructure, e.g. where there is the need to overcome discontinuities in bandwidth due to either variations of connection technologies or congestion situations.

The paper describes the design and implementation of an MA-activated service infrastructure, called ubiQoS,[1] for the QoS tailoring, control and adaptation of VoD flows over standard best-effort networks. The name ubiQoS refers to the 2-fold ubiquity dimension of our middleware approach:

- *Ubiquitous accessibility*. ubiQoS allows the reception of VoD flows anywhere, by tailoring multimedia content to user preferences, client device characteristics and available network bandwidth at negotiation time. In addition, it can monitor the provided QoS levels at provision time to perform corrective flow adaptation in response to modifications in available resources.
- *Ubiquitous middleware*. ubiQoS tends to diffuse its components in the system. At negotiation time, middleware components autonomously distribute on the hosts along the paths from VoD receivers to VoD sources. When new path segments are needed at provision time, e.g. in case of fault recovery, ubiQoS components can migrate to the required locations without imposing any service restart.

In particular, the paper details how it is possible to determine dynamically a QoS-aware active path between clients and servers, even in case of multicast distribution of the same VoD flow with differentiated QoS levels, discusses possible solutions, and finally evaluates the performance results of the completely decentralized Peer-to-Peer (P2P) implementation of the active path determination in ubiQoS.

[1] The SOMA-based ubiQoS middleware is available for download at http://lia.deis.unibo.it/Research/ubiQoS/.

2. Mobile Agents for QoS-aware active service infrastructures

The development, deployment and management of Internet services should face the challenging issues related to the increased QoS requirements and to the wide heterogeneity of access devices in the global provision scenario. It is within this context that the traditional end-to-end model of interaction shows its limits, thus suggesting the proposal of alternative scenarios. The network infrastructure should play an active execution role: for instance, in programmable networks, intermediate nodes operate on transmitted data and can be programmed by dynamically injecting service/user-specific code [6]. Several research activities start to recognize the suitability of MAs in this activated scenario where active services exploit intermediate nodes typically programmed at the application layer [5,7–9]. MAs are autonomous entities with capacity of coordination, able to dynamically move (together with their code and the reached execution state) to where resources are located, and able to adapt to current system conditions in a completely asynchronous way with regard to their launching user. The MA adoption simplifies the achievement of active service properties, such as:

- *Control decentralization*. Cooperating MAs can migrate during service provision and take autonomous management decisions based on local resource state. MAs can modify dynamically service distribution paths, e.g. in case of link failures or by following possible movements of users and client devices. In addition, agent autonomy permits asynchronicity between user actions and MA-performed tasks. For instance, MAs can operate service negotiation and establish the active path also when users/access devices are temporarily disconnected.
- *Tailoring*. MAs provide an effective mechanism to tailor services to user requirements and resource availability at negotiation time. Dedicated agents can retrieve profile information, can propagate this information to current user access points and customize service flows, depending on the current access devices and the already admitted service sessions. For instance, for accesses ranging from a laptop to a light PDA, an active service can decide to include/discard attachments in downloading e-mail messages.
- *Adaptability*. MAs simplify the adaptation of services in response to modifications in the availability of network resources at provision time [4]. For instance, MAs can locally monitor network resources and

dynamically migrate where needed to obtain a global view of the system state. This awareness permits to trigger management operations to correct the achieved QoS (re-negotiation, additional communication channels, etc.) by exploiting locality to congested resources.

In addition, MA solutions tend to address novel requirements and to provide infrastructure properties that significantly enhance the effectiveness of active services, thus facilitating their acceptance and diffusion. The most relevant property in this context is location awareness. MAs tend to maintain full visibility of the location of underlying system resources and to propagate this visibility to the service level. Location awareness is crucial to optimize resource usage within a locality [10,11]. For instance, MAs can decide to switch to another VoD server if the current one is overloaded and another one is currently available for a better service either in the same locality or in a near one.

In addition, MA-based middleware solutions facilitate the achievement of security and interoperability. On the one hand, MA systems not only introduce specific security mechanisms and policies to deal with untrusted incoming code, but also easily integrate standard solutions for secure services at the application level. For instance, MA operations can be controlled depending on permissions associated with authenticated principals and their role; based on these security mechanisms, any operation can be allowed, recorded and accounted to responsible users [12]. On the other hand, many MA systems achieve interoperability via compliance with general specifications, such as CORBA, and more MA-specific standards, such as the MA Systems Interoperability Facility and the Foundation for Intelligent Physical Agents specification [13,14].

Apart from the above properties that MAs can grant, one may argue that active services ask only for code mobility and that they do not require the full state migration typical of MAs: active services usually can take advantage of single-hop mobility patterns and not of multihop migrations. This consideration applies only to very simple services and commonly proposed case studies. The opportunity of state migration emerges in more complex and connection-oriented active services that require maintaining and moving sessions. This is evident in mobile computing scenarios where MA-based active nodes work as proxy of possibly disconnected users/devices [10]. A reasonable conclusion is "while none of the individual advantages of MAs is overwhelmingly strong, we believe that the aggregate advantages of MAs is overwhelmingly strong", as stated in Ref. [15].

Finally, since the beginning, MAs are considered a suitable technology for network and system management because of the possibility of moving management entities locally to administered resources [16]. For this reason, not so tied to the property of full mobility, many MA platforms give agents the possibility to access network and system properties, i.e. to have a certain degree of QoS awareness. In particular, most MA-based prototypes can interrogate network elements via standard management protocols such as SNMP and RMON [17,18]. When used for higher-level service management functions, MAs should also have visibility of system/application-specific indicators, such as the list of the current threads of an application and, for each of them, the CPU effective time and the allocated memory. This requirement is hard to grant because most MA platforms are implemented in Java and the Java Virtual Machine (JVM) tends to hide kernel-level system properties. However, some work has recently achieved interesting results in extending the monitoring visibility of Java MAs, with/without modifying the standard JVM [19,20].

2.1. MA-based QoS tailoring and adaptation of VoD flows

Two different phases for QoS management operations can be distinguished, QoS tailoring at negotiation time, and QoS adaptation at provision time.

The first phase precedes any real service flow and is necessary to negotiate the initial suitable QoS level. Its main goal is to determine the optimal engagement of resources, on the basis of the user preferences, of the characteristics of her current access device and of the currently available network resources. Ad hoc MAs could retrieve user/ terminal profiles and transport this information where needed. Then, the active service middleware could choose the VoD server capable of providing the requested content best satisfying the QoS requirements depending on the profiles. Once identified the server, the middleware could establish an active server-to-client network path. MAs can dynamically install along this path, to negotiate from there the QoS level that any path segment has to maintain and to decide for any required multimedia scaling operation. The VoD flow distribution is tailored also depending on already admitted service flows and current resource availability: MAs are in charge of application-level admission control and reservation of local resources.

The second phase is necessary during service provisioning and can impose strict constraints on middleware reaction times (RTs). Any deviation from conformity makes the service ineffective and should be avoided because it clashes with the initially negotiated QoS level. In fact, over best-effort networks, the quality of the provided VoD flows can change depending on the state of system/network resources along the distribution paths. Therefore, QoS should be controlled dynamically and possible modifications in available resources should promptly trigger adaptation operations. Adaptation operations include transformations on served VoD flows (from transcoding to frame resizing, from merging/splitting multilayered tracks to reducing frame resolution and

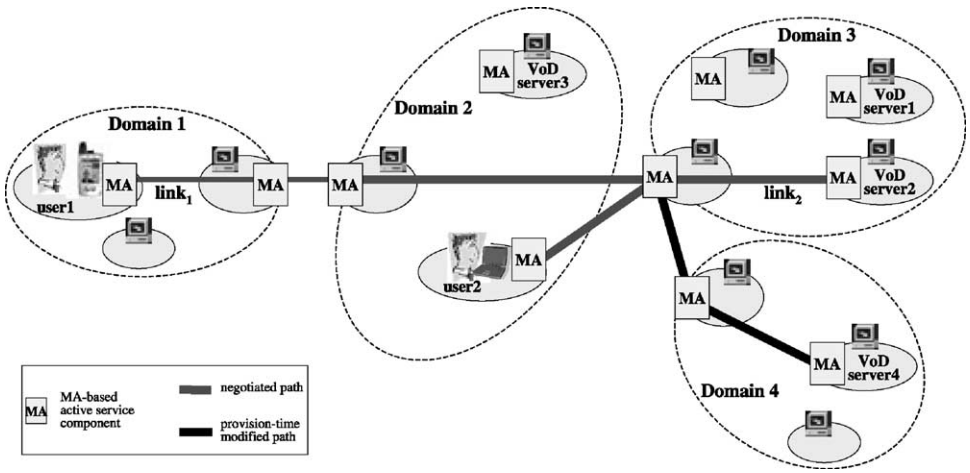

Fig. 1. An example of deployment scenario for an MA-based active service middleware.

rate) and ultimately also the modification of the established active path. In this case, it is necessary that a new negotiation phase takes place for a possible redistribution of MA-based active middleware components. MAs represent a suitable technology in this QoS management phase because they can operate locally to congested resources, by significantly reducing the needed RT. In addition, they can operate in a completely decentralized and autonomous way, thus improving the overall scalability of the QoS management solution.

To show more concretely how MAs can work on the tailoring, control and adaptation of the QoS of VoD flows, Fig. 1 presents a possible active service deployment scenario. Clients, servers and networked resources are organized in hierarchies of locality abstractions. Active service MAs (and their hosts) can be grouped into domains that usually correspond to (a set of) local area networks with common administration and management policies. At negotiation time, a suitable active path between the VoD clients and the servers is determined. Middleware MAs dynamically distribute on a subset of the hosts along this path. The different QoS requirements of user1 and user2 (and of their access devices) produce the distribution of a high-quality flow from the VoD server and the downscaling of the flow at the MA in domain2. At provision time, in case of degradation of $link_1$ bandwidth, the MA in domain2 can adapt the VoD transmission to user1 by reducing the frame resolution according to the receiver preference profile. If there are not enough resources to adapt QoS by respecting negotiated requirements, a new active path segment is established. The MA in domain3 tries to identify a suitable VoD server in its near domains; then, it negotiates with new MA-enabled hosts, and finally restarts the flow transmission from its interruption point (if server4 can support random-access to that VoD content). Apart from the time interval needed to establish the new path, the server swap is transparent to both receivers and intermediate nodes.

3. The ubiQoS active service middleware

The above solution guidelines have driven the design and implementation of an MA-based active service middleware, called *ubiQoS*, for the support of QoS tailoring, control and adaptation of VoD flows over best-effort networks. ubiQoS is built on top of an MA framework, called Secure and Open MAs (SOMA).[2] The choice of SOMA is motivated by its middleware facilities for the rapid development and deployment of MA-based Internet services. SOMA provides facilities for QoS awareness and QoS management (Monitoring and QoS facilities), for the definition of suitable trade-offs between security level and performance (Security facility), and for the interworking with other MA platforms, legacy systems, resources and services (Interoperability facility) [10,14]. In addition, ubiQoS exploits RTP for VoD flow transmission, due to the RTP diffusion in application-level approaches to QoS.

The ubiQoS ultimate goal is to allow ubiquitous accessibility of VoD services from any device and from any Internet access point, with the proper QoS level. Any client request is served after an initial negotiation phase that establishes an active path connecting the requesting client to a suitable VoD server, i.e. a server that could provide the requested VoD content with a QoS level greater or equal to the required one. In the current implementation, ubiQoS handles MJPEG flows and expresses the VoD QoS level as a tuple including frame rate, frame size, compression factor, and jitter. The requested QoS tuple is obtained by combining the requirements stored in the profiles of the user and of her current access device. If the QoS offered by the chosen VoD server is greater than needed, some ubiQoS MAs on the active path can downscale the flow. In this phase, MAs may migrate to intermediate nodes to install where needed operations are not yet available. For instance,

[2] The SOMA platform is available for download at http://lia.deis.unibo.it/Research/SOMA/.

any node in the active path requires the local presence of an admission control MA in charge of monitoring on-line local resource availability and of performing application-level reservation of local resources.

The provisioning of QoS-enabled VoD services over the Internet requires also a dynamic control of resource availability at provision time and the consequent handling of adaptation operations. These control phases should be enforced on any segment of the active path, and the MA technology can help in performing QoS monitoring in any locality traversed by VoD flows in order to decide locally any corrective intervention. Any local QoS degradation triggers adaptation operations on exchanged VoD flows at the ubiQoS MA adjacent to the congested segment. The middleware can locally decide how to work on the flow, e.g. via format transcoding, by maintaining the path, or to establish new path segments, either connecting to the same VoD server or to a less loaded one.

To reduce overall traffic and latency and to increase service scalability, ubiQoS organizes distributed caches of frequently accessed VoD flows. In particular, intermediate active nodes can maintain local caches depending on the access patterns of the clients in their locality. The amount of space that an active node should devote to its local cache, the refreshing time and the replacement policy are all choices that strongly depend on the characteristics of the locality, of its available resources and of the local usual clients. As a consequence, it is important that administrators can control and modify cache parameters during service provision by specifying a proper management policy. At the moment, ubiQoS offers domain administrators the possibility to choose which percentage of disk free space has to be exploited for cache storage and to adopt either a least-frequently used or least-recently used replacement policy.

3.1. The ubiQoS middleware components

Four types of ubiQoS MAs distribute along the active path between the VoD clients and servers for flow provisioning:

(1) *ubiQoS proxies* are in charge of admission control/reservation. They monitor system- and application-level state of their local resources and are able to trigger local adaptation operations. They coordinate with previous and next proxies in the active path both during the initial negotiation phase and at provision time when resource availability changes. In particular, as detailed in the following, they also collaborate to determine the active path in response to client requests.
(2) *ubiQoS processors* are in charge of tailoring and adaptation operations on VoD contents depending on the QoS levels required in the currently provided sessions. In addition, in response to new client requests, new processors retrieve profile information and migrate to the involved proxies in order to establish the needed active path depending on client/server location.
(3) *ubiQoS client stubs* forward VoD client requests to ubiQoS proxies and redirect RTP flows to their local visualization tools in a transparent way, to integrate ubiQoS with legacy VoD players. At the moment, we have implemented ubiQoS client stubs for JMF [21] and Mbone vic [22] players.
(4) *ubiQoS server stubs* answer to service requests from ubiQoS components by encapsulating VoD flows from legacy servers into RTP flows transparently. Up to now, we have implemented server stubs for JMF data sources.

All the above middleware components are implemented as MAs to permit dynamic installation and updating of existing functions even while ubiQoS is operating. Details about the design and implementation of the ubiQoS middleware components are presented in Ref. [23]. In the following, the paper focuses on how, in response to a specific service request, ubiQoS determines the proper QoS-dependent active path where processors and proxies dynamically and automatically install, if not already present.

3.2. Active path determination in ubiQoS

In ubiQoS, the determination of active paths is achieved dynamically by ubiQoS proxies, which organize a network of P2P mediators. ubiQoS proxies solve the QoS routing problem by collaborating in building suitable QoS-aware paths in response to client requests.

Any ubiQoS locality hosts at least one proxy. If multiple proxies are available in the same locality, only one of them has registered in the local discovery service (see Section 4 for additional details about the SOMA naming infrastructure) as the primary mediator. The other proxies register themselves as secondary mediators, capable of replacing the primary for fault-tolerance reasons, similarly to what occurs for DNS fault tolerance. Anytime either a ubiQoS client stub is likely to request a VoD flow or a ubiQoS server stub is likely to update the lists of offered VoD contents, they communicate their willingness to the primary proxy in their locality. At their first interaction, stubs interrogate the local discovery service to obtain the complete list of the proxies available in the locality. Successively, they re-interrogate discovery only in case of failure of both the primary proxy and all already known possible secondary ones.

On the one hand, a server with a newly offered VoD content sends an XML-based advertisement to its local ubiQoS proxy by specifying a title for the VoD flow, its textual description, a tuple indicating the maximum QoS level offered (in the current implementation, expressed only in terms of frame size and frame rate), and the associated cost as a polynomial function of the QoS parameters included in the tuple. ubiQoS proxies automatically invalidate VoD content entries from servers at regular

time intervals; server stubs are in charge of refreshing registrations by renewing the lease if they are interested to continue to offer the registered VoD flows.

On the other hand, a client requesting a VoD flow interrogates the proxy with an XML-based query that can include a search sub-string for the VoD title, a search sub-string for the VoD description, a tuple with the minimum acceptable values for QoS parameters, the maximum price the client is ready to pay for the service, and an initially void list of proxies already traversed by the query (the partially determined candidate for the active path).

When a ubiQoS proxy receives a query from one client stub, it first checks in its registration table to find a directly known server stub that provides a compatible VoD content. If several are found, the minimum cost server is chosen. Otherwise, the proxy acts as a client in the P2P mediator network and forwards the request to maximum N proxies (N is configurable and usually put to 5) in close ubiQoS localities. Close proxies are reached by exploiting the SOMA naming system and the SOMA locality organization [10,14]. Before any request forwarding, the proxy decrements the maximum cost specified by the client by the cost associated to the transmission of the VoD content with the minimum acceptable QoS level over the link client-to-proxy. In addition, it includes its GUID to the list of traversed proxies in the query. A request is forwarded if and only if:

- the client request can be still satisfied, signaled by a still positive cost value;
- the proxy examining the request is not already included in the list of traversed ones, to avoid loops;
- the number of traversed proxies is less than a configurable maximum number of hops.

Once reached a potentially suitable server, proxies start exchanging a new sequence of coordination messages on the way back of the determined path, this time going backwards from the server to the client. Any proxy performs the needed local resource reservation and possibly decides if a downscale tailoring operation should operate in its locality. The enforced policy is to downscale the flow at the first possible node in the server-to-client path, according to the QoS requirements of all served clients in the distribution sub-tree. In addition, any proxy determines the actual costs associated with its active path segment. These costs could be different from the ones determined in the client-to-server way, e.g. because of the tailoring decisions made on the way back.

At the end, the client stub receives all service bids together with their associated costs to let the user choose among them. The user choice triggers the VoD flow starting on the dynamically determined QoS-aware path. The client stub ignores any other service bid received after the choice. In addition, the intermediate nodes belonging to alternative active paths, not chosen for final service provisioning, invalidate their reservations after a configurable timeout and release the corresponding local resources.

Let us finally observe that proxies cannot only behave as clients when forwarding requests towards the VoD servers, but also act as servers when a received request can be satisfied by either a currently traversing VoD content or a locally cached one. This produces the dynamic determination of active multicast distribution trees. Details about ubiQoS multicast and its decentralized caching infrastructure are out of the scope of the paper and presented in Ref. [23].

4. Implementation insights

ubiQoS requires expressive naming solutions to retrieve dynamically all the information needed to the active service middleware, from the list of proxies and hubs in near network localities to the user/device profiles to drive QoS tailoring and adaptation. ubiQoS exploits the SOMA naming support that integrates discovery and directory servers. Ref. [10] reports a full description of the SOMA naming implementation, while this present paper only sketches some elements to permit to fully understand how the ubiQoS components interoperate.

SOMA discovery and directory servers provide different naming solutions suitable for different goals. They differ in visibility scope (local versus global), flexibility (rigidly predefined and simple structure vs. flexible content and organization), and performance (limited low-level efficient protocols vs. complete high-level searching/registering operations). For instance, in ubiQoS LDAP-compliant directory servers store the profiles of registered users and of recognized access devices. Profile information about new access devices may be described dynamically as an extension of already included profiles that exploit the CC/PP composition capabilities. In addition, Jini-based discovery servers permit to access the active middleware information visible in a single locality, i.e. the list of locally known proxies and hubs (usually, the ones within or close to that locality).

To determine the QoS of the VoD flow exchanged on any active path segment, any ubiQoS processor autonomously decides the QoS level to request to the following processor towards the server. A whole interval of QoS parameters is usually permitted; the processor chooses the QoS point to enforce in the permitted QoS space interval depending on the local resource consumption policy. In the current ubiQoS implementation, system administrators can choose two simple policies: Best QoS and Lower QoS. The Lower QoS policy aims at reducing the resource consumption by reserving only the set of resources that minimizes a specified local cost function. On the contrary, the Best QoS is a greedy policy that chooses the QoS point reserving the maximum local resource usage. While enforcing Best QoS, new VoD flow requests can also dynamically modify QoS points of accepted flows by pre-empting previously committed resources. In addition, ubiQoS provides administrators with GUIs to force the processor decision by

Fig. 2. Some visualization and control GUIs: RTP sender/receiver reports and QoS monitoring information, profile-dependent QoS parameters and adaptation weights, and a JMF-based player integrated with the ubiQoS client.

directly specifying low-level QoS parameters, e.g. by changing frame rate/size as shown in Fig. 2.

At provision time, the proxy can trigger the local processor to modify provided QoS by causing the solution to move in the QoS space according to the preferences expressed in the client profile. We use relevance weights associated with the different QoS parameters [23]. For instance, the profile of a device with limited display capabilities can specify a frame rate weight larger than frame resolution to indicate a preference in degradation of image quality instead of frequency decrease. In other words, the weights determine the preferred directions of correction actions in the QoS space when the proxy detects a modification in local resource availability. Only when the allowed correction region is null, the proxy triggers the P2P determination of a new QoS-aware active (sub-) path.

5. Experimental results

To evaluate the feasibility and effectiveness of our approach, we have deployed the ubiQoS infrastructure over a set of geographically distributed networks, with heterogeneous bandwidth (10/100 Mb/s) and interconnected via GARR, i.e. the Italian Academic and Research Network. Any local network is modeled by one ubiQoS locality and includes heterogeneous hosts (SUN Ultra5 400 MHz workstations with Solaris 7, 128 MB PentiumIII700 PCs with Microsoft WindowsNT and 128 MB PentiumIII700 PCs with SuSE Linux 7.1). In this deployment scenario, we have measured several performance figures, listed in the following, to estimate the overhead and the RT of the ubiQoS middleware.

As described in Section 3.2, the initial phase of client-to-server path determination in ubiQoS involves the client, the dynamically retrieved server, and some active intermediate nodes. The establishment of any path segment requires the creation of an RTP connection, the migration of one session-specific processor MA, the resource admission control/reservation on the destination node, and the negotiation of the suitable, possibly adapted, QoS level. In addition, at regular time intervals, configurable according to the degree of dynamicity of the deployment scenario, ubiQoS clients/proxies interrogate the local SOMA discovery to update the list of nearby ubiQoS proxy peers.

Fig. 3 reports the average value of the active path setup time measured over a large set of deployment scenarios, with 50 geographically and randomly distributed clients with heterogeneous QoS requirements and a set of four ubiQoS servers with partially replicated VoD contents. The experimental results show an almost linear dependence of the path setup time on the number of dynamically determined intermediate active nodes. The path setup time usually does not exceed 5 s, with an average number of active nodes less than 5 and an average number of required migrations less than 3. This interval is significantly larger than the one necessary to establish a single RTP connection between one client and one already known server, but is widely acceptable in most categories of multimedia services, i.e. non-interactive VoD ones, since it affects only the starting delay at the client side.

Let us observe that the migration of ubiQoS components is not required by any service request. In fact, ubiQoS proxies are infrastructure components that dynamically install where needed in response to a service request and that can persist there to serve future requests for VoD flows.

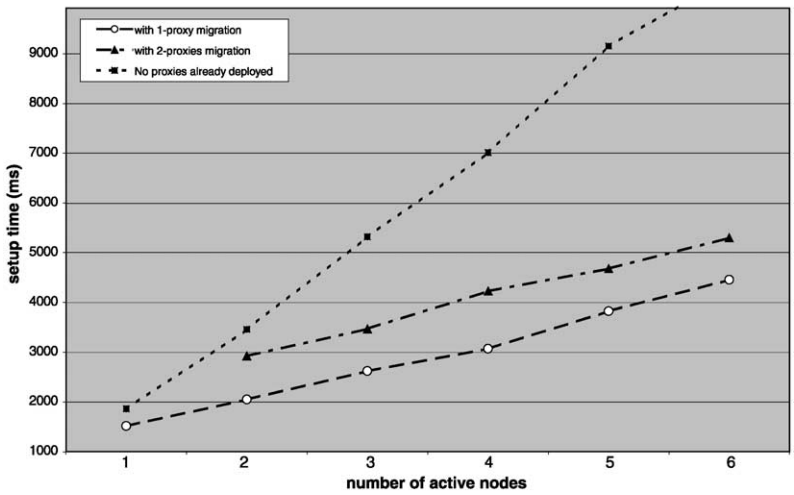

Fig. 3. ubiQoS path setup time.

ubiQoS processors are session-specific, migrate at the setup of any new active path, but they are significantly lighter than proxies and can exploit the local availability of suitable Java classes on the hosting nodes. In fact, ubiQoS maintains processor classes deriving from previous VoD sessions in a local code repository, managed with a least-recently used replacement strategy. For the above reasons, the figure also reports how the number of needed ubiQoS proxy migrations affects the path setup time: the delay significantly reduces in the usual case of a service request that triggers the installation of proxies only over a small set of new active nodes.

In addition, even in the case of large-scale deployment scenarios, such as the typical Internet-wide service distribution, the number of active nodes along the server-to-clients path tends to be very small. In fact, it is usually much lower than the number of traversed routers/gateways. ubiQoS proxies and processors need to operate only where there are either strong bandwidth discontinuities or forking of the VoD multicast tree. In any experimented scenario, the dynamically determined ubiQoS active paths never included more than four intermediate active nodes. We needed to force the middleware decisions by adding ad hoc bottlenecks to have longer active paths in order to measure the setup times, shown in Fig. 3, for the cases with 5 and 6 active nodes.

Anyway, the path setup overhead is largely counterbalanced by the possibility of performing prompt reactions at provision time in response to dynamic changes in network resources. ubiQoS proxies integrate standard RTP report transmissions with event-triggered exchanges of QoS-related information provided by SOMA monitoring [20]. This permits to overcome the RTP limit related to the frequency of reports, which is statically determined in RTP as a fixed percentage of the maximum network bandwidth available at the time of connection establishment [24]. Fig. 4 shows the ubiQoS RT when the bandwidth reduction over an active path segment triggers the downscaling of the transmitted VoD flow. RT is measured as the time interval between the congestion occurrence and the starting of the adapted flow

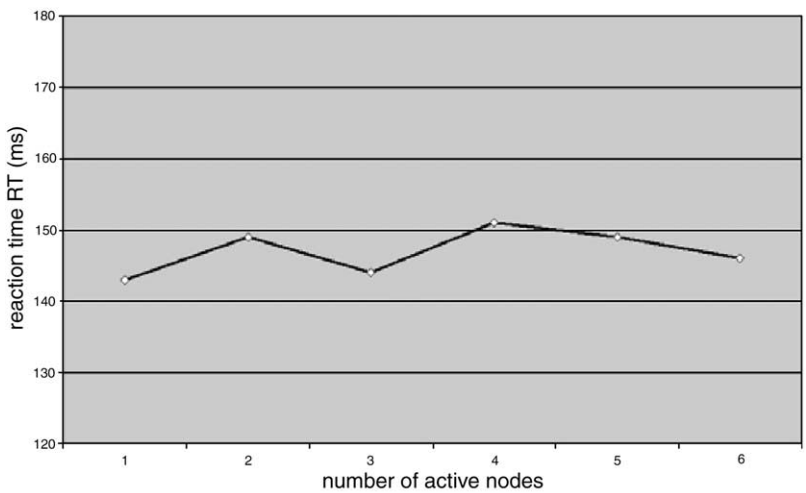

Fig. 4. ubiQoS reaction time.

transmission from the downscaling processor. The figure reports the average results obtained by measuring RT for all the different possible positions of the congested segment in the path (from the server to the first active node, from the first active node to the second, etc. and from the Nth node to the client). The RT average is around 148 ms and, most important, almost independently of the distance between the client/server and the congested segment. This is made possible by the fact that any proxy autonomously monitors, controls and manages its local path segment. RT values exhibit a very small variance and very limited random fluctuations around the average value, exclusively due to runtime variations in the non-ubiQoS network traffic over GARR.

6. Related work

Many research groups have recently claimed the suitability of programmable networks for a wide spectrum of Internet services. Programmable networks can help in fast prototyping and deploying new network-layer protocols, e.g. for congestion control, topology-aware reliable multicast and virtual private networks [5,6]. Among the active service approaches, network programmability is exploited to deal with application-specific requirements, as in distributed information filtering and Web caching [7,25]. Most active service projects implement prototypes on top of the Java programming environment to facilitate code portability and mobility; some of them explicitly adopts the MA technology [8,9].

In the specific application domain of QoS and multimedia services, there are a few notable approaches that exploit intermediate active nodes. Baldi et al. designed a videoconference service by uploading Java mobile code in active routers, thus adopting a network-layer approach [26]. Amir et al. implemented a Media Gateway (MeGa) for the adaptive transcoding of multimedia flows [27]. Their work, however, is more focused on algorithms for efficient downscaling and adaptive bandwidth allocation than on dynamic reconfiguration and code distribution.

The Reflector project has proposed the active infrastructure most similar to ubiQoS [28]. Reflector is an application-level multimedia distribution system implemented in C++. It has been designed and deployed mainly to test and verify the feasibility of distributing low and medium bandwidth VoD flows to thousands of simultaneous users over the Internet. The Reflector technology had a significant success in the live broadcast of NASAs Pathfinder mission. However, Reflector designers learned from wide-scale deployment experiences that it is crucial to adopt technologies to facilitate the support to dynamic reconfiguration, code distribution and adaptation to changes in network resource availability. It is interesting that they are addressing the observed limitations of Reflector by adopting the MA technology to enhance the extensibility of their system [29].

Several research efforts in the last years addressed the issue of determining the QoS-optimal client/server path. The problem is usually identified in the literature as *QoS routing*. The achievement of the optimal decision imposes the collection of the updated and complete state of the QoS levels available on any involved network link/host. On this basis, several algorithms have been proposed to rapidly determine the path that optimizes either a QoS parameter (queuing delay, propagation delay, tailoring delay, etc.) or the resource utilization (number of hops, generated network traffic, load balancing, etc.) [30,31]. In general, this represents a NP-complete optimization problem and is usually faced by imposing simplification constraints, e.g. by reducing it to a Delay Constrained Least Cost path problem or a Bandwidth Constrained Least Cost path problem [31]. However, also with these simplifications, the execution time has been proved to be exponential in the worst case.

Therefore, several researches are investigating the determination of non-optimal solutions via different kinds of QoS-specific heuristics. A significant recent result is described in Ref. [32] where the QoS routing optimization is simplified and transformed into a Steiner Minimal Tree problem. It is significant that, apart from the adopted algorithm, the approach uses MAs to locally monitor the state of the distributed resources and to consequently determine the QoS-aware path by exploiting MA mobility. We have experimented the approach and developed a prototype version of it on top of SOMA. Even if effective in the case of small-scale networks with quasi-static topologies and rare congestion/hot spots, where the QoS path determination can be performed also off-line with a reasonable estimate, the solution does not scale well in large-scale open environments where QoS-aware routing must be solved dynamically in response to any service request, as in the service provisioning scenario addressed by ubiQoS.

Some interesting hints for QoS routing solutions can be borrowed by the recent P2P research, involved in investigating the best way to organize content distribution networks in a completely decentralized way [33,34]. In fact, P2P proposes solutions for discovering information resources at provision time via dynamically organizing advertisement/query systems with no points of centralization. P2P discovery include content-based and content-agnostic solutions where, respectively, the organization of the peer interaction depend or does not depend on the indexed resources. However, P2P solutions currently focus on the simple retrieval of resources and do not provide the needed visibility of routing/system information, which is crucial in multimedia QoS management. For instance, JXTA search completely hides the network topology and does not support the association of any routing hop with a dynamically evaluated QoS-dependent cost [35]. To the best of our knowledge, ubiQoS is the first MA-based middleware adopting a P2P scheme of solution for the determination of

QoS-aware active paths in the multimedia distribution scenario.

7. Conclusions and on-going work

The work accomplished within the ubiQoS project has shown the feasibility and the effectiveness of addressing the QoS issues of VoD services via a middleware infrastructure of active nodes. This choice is suitable to enable QoS differentiation according to user/terminal profiles and to perform domain-specific flow tailoring, control and adaptation over a best-effort network. The effectiveness of the ubiQoS implementation in terms of MAs depends on operating close to controlled resources to take locality-dependent management decisions and on dynamically distributing middleware components at provision time.

First ubiQoS performance results are encouraging and stimulate refinements and extensions of the implemented middleware. In particular, we are extending the middleware to include VoD client stubs based on the Java 2 Micro Edition and VoD multimedia players such as the TealMovie one [36], targeted to Palm portable devices with very limited visualization capabilities.

With regard to the adopted solution for active path determination, we are extending it to support the autonomous and dynamic organization of discovery servers in hierarchies depending on the dynamically determined network topology, on the history of previously established active paths, and on the location/frequency patterns of client requests. Even if not in the specific area of QoS-aware VoD distribution, recent activities in the P2P research are demonstrating that significant performance improvements can be achieved in large-scale distribution networks by organizing nodes in a very small number of hierarchical layers (3/4), especially when combined with distributed caching such as in ubiQoS [37]. Finally, we are starting to investigate the possibility to drive the active path determination by exploiting hashing functions that depend on the requested VoD content and by consequently pruning routing sub-trees as in the FreeNet content-based search.

Acknowledgements

Work supported by the Italian Ministero dell'Istruzione, dell'Università e della Ricerca in the Project 'MUSIQUE: Infrastructure for QoS in Web Multimedia Services with Heterogeneous Access'.

References

[1] J. Krikke, Graphics applications over the wireless web: Japan sets the pace, IEEE Computer Graphics and Applications 21 (3) (2001).

[2] X. Xipeng, L.M. Ni, Internet QoS: a big picture, IEEE Network 13 (2) (1999).

[3] N.E. Andersen, et al., Applying QoS control through integration of IP and ATM, IEEE Communications 38 (7) (2000).

[4] P. Bellavista, A. Corradi, C. Stefanelli, An integrated management environment for network resources and services, IEEE Journal on Selected Areas in Communication 18 (5) (2000).

[5] K. Psounis, Active networks: applications, security, safety, and architectures, IEEE Communications Surveys 2 (1) (1999).

[6] H. Yasuda (Eds.), Second International Working Conference on Active Networks (IWAN'00) Proceedings, Japan, Springer-Verlag Lecture Note on Computer Science, 2000.

[7] A. Ghosh, M. Fry, G. MacLarty, An infrastructure for application level active networking, Computer Networks 36 (1) (2001).

[8] D. Putzolu, S. Bakshi, S. Yadav, R. Yavatkar, The phoenix framework: a practical architecture for programmable networks, IEEE Communications 38 (3) (2000).

[9] S. Karnouskos, I. Busse, S. Covaci, Agent based security for the active network infrastructure, First International Working Conference on Active Networks (IWAN'99) Proceedings, Germany, Springer-Verlag Lecture Notes on Computer Science, 1999.

[10] P. Bellavista, A. Corradi, C. Stefanelli, Mobile agent middleware for mobile computing, IEEE Computer 34 (3) (2001).

[11] J. Bolliger, T. Gross, A framework-based approach to the development of network-aware applications, IEEE Transactions on Software Engineering 24 (5) (1998).

[12] P. Bellavista, A. Corradi, R. Montanari, C. Stefanelli, Security in programmable network infrastructures: the integration of network and application solutions, Second International Working Conference on Active Networks (IWAN'00) Proceedings, Japan, Springer-Verlag Lecture Notes on Computer Science, 2000.

[13] IKV++, Grasshopper 2: the Agent Platform, http://www.grasshopper.de.

[14] P. Bellavista, A. Corradi, C. Stefanelli, Protection and interoperability for mobile agents: a secure and open programming environment, IEICE Transactions on Communications E83-B (5) (2000).

[15] R.H. Glitho, Emerging alternatives to today's advanced service architectures for Internet telephony: IN and beyond, Computer Networks 35 (5) (2001).

[16] M. Baldi, G.P. Picco, Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications, International Conference on Software Engineering (ICSE'98) proceedings, IEEE Computer Society Press, Silver Spring, MD, 1998.

[17] B. Pagurek, Y. Wang, T. White, Integration of Mobile Agents with SNMP: Why and How, IEEE/IFIP Network Operations and Management Symposium (NOMS 2000) Proceedings, USA, IEEE Press, New York, 2000.

[18] D. Gavalas, M. Ghanbari, M. O'Mahony, D. Greenwood, Enabling Mobile Agent Technology for Intelligent Bulk Management Data Filtering, IEEE/IFIP Network Operations and Management Symposium (NOMS 2000) Proceedings, USA, IEEE Press, New York, 2000.

[19] G. Czajkowski, T. von Eicken, JRes: a Resource Accounting Interface for Java, ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'98) Proceedings, USA, ACM Press, New York, 1998.

[20] P. Bellavista, A. Corradi, C. Stefanelli, Java for on-line distributed monitoring of heterogeneous systems and services, The Computer Journal 45 (6) (2002).

[21] Sun Microsystems, Inc.—Java Media Framework (JMF) API, http://www.java.sun.com/products/java-media/jmf.

[22] UCL Networked Multimedia—MBone Conferencing Applications, http://www-mice.cs.ucl.ac.uk/multimedia.

[23] F. Baschieri, P. Bellavista, A. Corradi, Mobile Agents for QoS Control, Tailoring and Adaptation over the Internet: the ubiQoS Video on Demand Service, Second International Symposium on Appli-

cations and the Internet (SAINT'02) Proceedings, Japan, IEEE Computer Society Press, Silver Spring, MD, 2002.
[24] S. Wenger, RTCP-based Feedback: Concepts and Message Timing Rules, IETF Internet Draft, http://search.ietf.org/internet-drafts/draft-wenger-avt-rtcp-feedback-01.txt, 2000.
[25] W. Marshall, C. Roadknight, Provision of quality of service for active services, Computer Networks 36 (1) (2001).
[26] M. Baldi, G.P. Picco, F. Risso, Designing a Videoconference System for Active Networks, Second International Workshop on Mobile Agents (MA'98) Proceedings, 1998.
[27] E. Amir, S. McCanne, R. Katz, An Active Service Framework and its Application to Real-time Multimedia Transcoding, ACM SIGCOMM Conference Proceedings, ACM Press, New York, 1998.
[28] F. Kon, R. Campbell, S.M. Tang, M. Valdez, Z. Chen, J. Wong, A component-based architecture for scalable distributed multimedia, 14th International Conference on Advanced Science and Technology (ICAST'98) Proceedings (1998).
[29] F. Kon, R.H. Campbell, K. Nahrstedt, Using dynamic configuration to manage a scalable multimedia distribution system, Computer Communications 24 (1) (2001).
[30] A. Jüttner, B. Szviatovszki, I. Mécs, Z. Rajkó, Lagrange Relaxation Based Method for the QoS Routing Problem, IEEE INFOCOM Conference Proceedings, IEEE Press, New York, 2001.
[31] S. Chen, K. Nahrstedt, Distributed QoS Routing with Imprecise State Information, Seventh International Conference on Computer Communications and Networks Proceedings, IEEE Press, New York, 1998.
[32] S. González-Valenzuela, V.C.M. Leung, QoS routing for MPLS networks employing mobile agents, IEEE Network 6 (3) (2002).
[33] S. Botros, S. Waterhouse, Search in JXTA and Other Distributed Networks, First International Conference on Peer-to-Peer Computing Proceedings, IEEE Press, New York, 2002.
[34] D. Stolarz, Peer-to-Peer Streaming Media Delivery, First International Conference on Peer-to-Peer Computing Proceedings, IEEE Press, New York, 2002.
[35] Project JXTA—http://www.jxta.org.
[36] TealPoint Software—TealMovie, http://www.tealpoint.com.
[37] F.S. Annexstein, K.A. Berman, M.A. Jovanovic, K. Ponnavaikko, Indexing Techniques for File Sharing in Scalable Peer-to-Peer Networks, 11th International Conference on Computer Communications and Networks Proceedings, IEEE Press, New York, 2002.

Paolo Bellavista is a research associate of computer engineering at the University of Bologna. His research interests include MAs, mobile computing, network and systems management, location/context-aware services, and adaptive multimedia systems. He received a PhD in computer science engineering from the University of Bologna. He is a member of the IEEE, the ACM, and the Italian Association for Computing. Contact him at pbellavista@-deis.unibo.it.

Antonio Corradi is a full professor of computer engineering at the University of Bologna. His research interests include distributed systems, object and agent systems, network management, and distributed and parallel architectures. He received an MS in electrical engineering from Cornell University. He is a member of the IEEE, the ACM, and the Italian Association for Computing. Contact him at acorradi@deis.unibo.it.