An Active Middleware to Control QoS Level of Multimedia Services

Paolo Bellavista, Antonio Corradi, Rebecca Montanari
DEIS - University of Bologna
{pbellavista, acorradi, rmontanari}@deis.unibo.it

Cesare Stefanelli
Dip. Ingegneria - University of Ferrara
cstefanelli@ing.unife.it

Abstract

The provision of novel Internet services has both to specify and to maintain differentiated Quality-of-Service (QoS) levels. Services should tailor to different user QoS preferences together with the differentiated quality properties deriving from servers and access points and devices, from workstations connected with high-capacity networks to wearable devices exploiting limited-capacity wireless links. The paper claims that service provision with negotiated and controlled QoS over best-effort networks calls for a support infrastructure that activates intermediate nodes along the path between clients and servers. In fact, the paper proposes MASQ, an active middleware solution for the QoS management of Video-on-Demand (VoD) streaming. At negotiation time, MASQ exploits code mobility to establish an active path between the requesting client and the VoD server chosen to tailor VoD flows based on user profiles and device properties. At provision time, MASQ dynamically controls the offered QoS level to adapt locally when and where network resource availability changes. MASQ significantly benefits from dynamic and flexible programmability stemming from the employment of high-level policies.

1. Introduction

The growing number of users who exploit the Internet to access multimedia services is a durable and lasting trend in current distributed computing. The same is for the number of multimedia service users with an ubiquitous access to the global network. These mounting trends impose to differentiate and tailor the Quality of Service (QoS) in multimedia streaming, on the basis of personal preferences and usage class, to differentiate billing depending on the actually received QoS levels.

The increasing diffusion of mobile access to the Web, such as the NTT DoCoMo i-Mode and the Wireless Application Protocol (WAP) [1], widens this perspective even further. Terminals interested in accessing multimedia services exhibit heterogeneous characteristics (computing power, storage, display size and resolution, ...) by ranging from traditional workstations and PCs, to laptops, personal digital assistants and smart phones, with wired/wireless continuous/intermittent connectivity.

The above scenario motivates service providers, network operators and final users to ask for technologies, mechanisms, and tools to support Video-on-Demand (VoD) services over the Internet with differentiated QoS levels. That means over a best-effort network infrastructure to control dynamically the QoS level provided at runtime. In addition, the competition of providers and the necessity of scalability force to propose scenarios where multimedia servers and resources can be replicated with the goal of a better service. This requires design and deployment methodologies that consider global resource availability, for instance, to decide suitable strategies for resource replication and allocation.

Such a service provision scenario introduces complex decisions: from choosing the best server that matches client QoS requirements to deciding new caching localities for successive requests, from identifying the current conditions of server load by monitoring network information to recognizing promptly congestion and abnormal traffic situations. The awareness of monitoring information together with the imposed overhead is crucial to estimate available resource state at service negotiation time and service provision time. Resource availability should be continuously controlled to achieve and maintain a specified QoS level, and to promptly adapt to modifications of QoS properties along the path between multimedia servers and clients [2]. For instance, one multimedia flow can be split into multiple paths if the bandwidth of active connections goes under a specified threshold. Alternatively, it is possible to react to network congestion by commanding application-specific transformations on VoD flows, for instance via format transcoding, media resynchronization, and by acting on packet buffering.

An emerging trend in multimedia QoS provisioning is to introduce distributed support infrastructures that are aware of all involved network resources and that can assign an active role to the nodes along the paths between clients and servers [3-10]. The paper claims that the best solution strategy for QoS support over a global best-effort network is an active middleware, where the duty of serv-

ice provision cannot be limited to server hosts and client nodes but should expand to the whole intermediate nodes. In fact, intermediate nodes have to participate not only as message routers but also as active service providers and perform management operations. They can monitor local resource state, can perform control actions and transformations on traversing VoD flows, can find new distribution paths more suited to current requirements, and can cache VoD contents. Intermediate nodes could offer their storage for distributed caching of popular VoD flows to decrease overall traffic and service response time. Another example is the case when statically negotiated QoS level cannot be maintained: the node that ascertains the problem is the most suitable for taking the needed correction locally and autonomously.

The paper describes MASQ (Multimedia Active Services for QoS), an active middleware for QoS management in VoD flow provisioning. At negotiation time, MASQ establishes an active path between the requesting client and the proper VoD server to tailor the quality of VoD flow depending on user profile, device characteristics and network resource availability during negotiation. At provision time, MASQ dynamically controls the offered QoS level to trigger local adaptation operations when and where resource availability changes.

MASQ is implemented in terms of coordinated Mobile Agents (MAs) that are capable and responsible of the activation of intermediate nodes along the VoD flow paths. MAs can reallocate dynamically, thus achieving the dynamic deployment required in open and global scenarios. The possibility to migrate both code and state during service provision permits session-dependent control and adaptation of network resources. In addition, QoS management operations can be performed locally to the critical points of the infrastructure, e.g., where there are discontinuities in bandwidth. Locality in resource monitoring, control and management is crucial to obtain effective response times in global provision scenarios [2].

The flexibility and applicability of the MASQ middleware significantly benefit from its dynamic programmability via the specification and enforcement of high-level policies. Event-triggered policies specify both how MASQ components allocate resources to served flows on the intermediate hosts along the active path and how MASQ should react to modifications in network resource availability to adapt currently served QoS levels. Policies permit to specify how to respond to changes in monitoring information and to separate cleanly QoS management strategies and actions from application-specific behavior.

2. QoS Management of VoD Flows

QoS awareness is a key requirement for VoD services over best-effort networks, and QoS visibility is the property that an active middleware should be built around, to tailor and to control dynamically provided QoS levels. We can distinguish two successive phases for QoS granting:

- ***QoS tailoring at negotiation time***, to customize service QoS levels to specific user requirements, access device properties, and resource availability. For instance, an active VoD service can reduce either frame rate or image resolution depending on current access device;
- ***QoS adaptation at provision time***, to dynamically change and maintain QoS level during provisioning, by taking advantage of the knowledge of run-time situations. Only a continuous monitoring of system state can trigger promptly management operations, such as renegotiation, new communication channels establishment, and multimedia flow transcoding.

The static phase that precedes the real data flow can be considered a static negotiation of the QoS level among client, server, and all intermediate network resources. The main goal of QoS negotiation is to choose the best possible resource set, on the basis of user profiles, access devices and the current system situation. The first step requires retrieving user preferences and accessing device characteristics; then, the active middleware should identify one server that best satisfies the specific QoS requirements. Once the server is known, the infrastructure should establish a server-to-client network path. The components of the active infrastructure should be available (or should install dynamically) on any needed intermediate host along the active path. Distributed middleware components are in charge of negotiating the QoS level maintained in any path segment and of operating possible multimedia downscaling operations. In addition, intermediate entities should perform application-level admission control and local resource reservation: one can admit new reservations for VoD flows (or for enhancing the QoS level of already established ones) only if enough resources are locally available. The VoD flow distribution is negotiated to match client QoS specifications, also depending on already admitted service flows and current resource availability.

The second adaptation phase takes place during service provision and this dynamicity imposes even stricter constraints than the ones on the static phase. Any deviation from conformity could make the service ineffective and could clash with the initially negotiated QoS level. In fact, the QoS levels of VoD flows should change depending on the state of system/network resources along distribution paths. Therefore, QoS should be controlled during the whole provision, and changes in resource availability should trigger adaptation to readjust QoS levels. Adaptation can affect the transmission of VoD data (from transcoding to frame resizing, from merging/splitting multi-layered tracks to reducing frame resolution and rate) and can ultimately modify established VoD paths. In this case, a new negotiation phase is preliminary before a redistribution of active middleware components. The cor-

rective operations at provision time should consider user/terminal profiles so to assign priorities to adaptation alternatives. For instance, a personal digital assistant with limited display capabilities can suggest lowering frame resolution instead of decreasing frame rate.

Figure 1 presents a possible deployment scenario where an active middleware tailors, controls and adapts the QoS of VoD flows. The infrastructure achieves scalability by organizing clients, servers and network resources in hierarchies of locality abstractions. Active hosts can be grouped into domains that usually correspond to (a set of) local area networks with common administration and management policies. Domains are the way to confine the visibility scope of middleware components thus limiting the management complexity.

At negotiation time, QoS requirements of user1 and user2 (and of their access devices) have suggested a high-quality flow from the VoD server and its scaling down at the active middleware component in domain2. In case of degradation of $link_1$ bandwidth during provisioning, middleware components in domain1 and domain2 coordinate with each other. It is the domain2 component to adapt the VoD transmission for user1 by reducing frame resolution according to user1 preferences. If there are no resources to adapt QoS by respecting negotiated requirements, e.g., in case of failure of $link_2$, a new VoD path segment is to be established. The domain3 infrastructure tries to identify a suitable VoD server in its neighbor domains. It then negotiates with new active hosts to finally restart flow transmission from its interruption point (if server4 supports random-access to the same VoD content). Apart from the time required to establish the new path, the server swap is transparent to the receiver and to all other intermediate nodes.

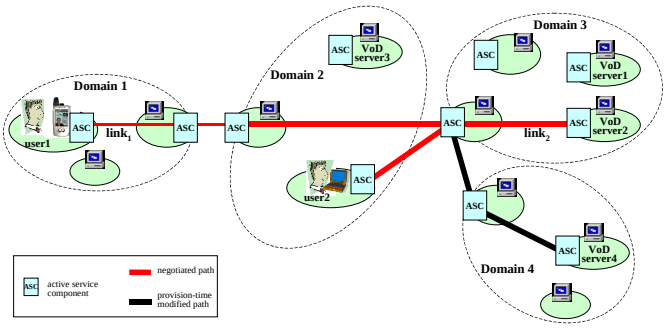

Figure 1. A possible deployment scenario of the active middleware

3. The MASQ Middleware

The previously presented solution guidelines have driven the design of the MASQ active middleware in supporting QoS tailoring, control and adaptation of VoD flows. MASQ is the result of the integration of policy-based specifications for QoS management with a previous prototype of active multimedia middleware called ubiQoS, described elsewhere [11]. After sketching some aspects of the MASQ middleware, the paper focuses specifically on how to exploit high-level policies to rule the management of multimedia QoS and on how the first MASQ prototype implements this approach.

An important preliminary to the presentation of the MASQ architecture is the choice of the adopted basic protocol for exchanging VoD flows. Our active middleware is based on the Real-time Transport Protocol (RTP) [12]. The motivation of RTP stems from its wide diffusion in application-level approaches to QoS and from its relevance in the areas of mobile communications and multimedia distribution [13, 14]. As any application-level QoS solution, RTP attempts to meet QoS requirements without modifying the underlying best-effort network level. RTP permits to obtain monitoring information about currently available QoS levels and to notify service components of any QoS modification. Most relevant information items are sender reports, generated by the sources of RTP-based multimedia flows, and receiver reports, filled by the target VoD clients. All reports include sender information (RTP timestamps and the number of packets and bytes already transmitted) and receiver statistics about the flow (highest sequence number received, inter-arrival jitter, and fraction of lost packets since last report).

3.1. Architecture

The MASQ middleware is implemented in terms of two main types of components working along the active path between the (also multiple) VoD clients and servers for flow provisioning:

1) ***MASQ proxies*** are in charge of admission control/reservation for incoming/outgoing flows. They monitor system- and application-level state of local resources with the goal of triggering local QoS adaptation operations. Proxies are organized in chains to cover the whole path from clients to servers. They coordinate with their previous and next proxies in the active path both in the initial negotiation phase and at provision time for resource availability changes. Proxies exploit discovery and directory solutions to find other MASQ components in local and neighbor domains and to retrieve user/terminal profiles expressed according to the Composite Capability Preference Profile (CC/PP) specification;
2) ***MASQ processors*** are in charge of performing QoS tailoring and adaptation operations on VoD contents depending on the specific QoS requirements of established sessions. In response to a new client request, one of this processor is in charge of retrieving QoS re-

quirements of the user and of her current access device. The processor should carry this information by migrating to the nodes hosting MASQ proxies with the goal of establishing the active path.

In addition to proxies and processors, the MASQ middleware includes lightweight client/server stubs to permit the integration with legacy VoD players/servers.

Figure 2 depicts the modular architecture of MASQ components. They are implemented as MAs to permit dynamic installation and updating of existing functions even while the MASQ middleware is operating. Proxies are the principal middleware components that monitor and control the resources locally to the active nodes. They can install themselves permanently on new hosts taking part in active paths and their migration is typically single-hop. On the contrary, processors are transient and session/flow-dependent components that have to propagate from the client toward the server by carrying the QoS requirements of client user/device for that specific service flow. Let us note that resource reservation, adaptation operations and path decisions may depend on previously established path segments. This is the reason that makes important the multiple-hop potential typical of MAs.

Details about the implementation of MASQ components in terms of MAs are presented in [11], while the rest of the paper concentrates on the design and implementation of the policy-enabled MASQ proxy.

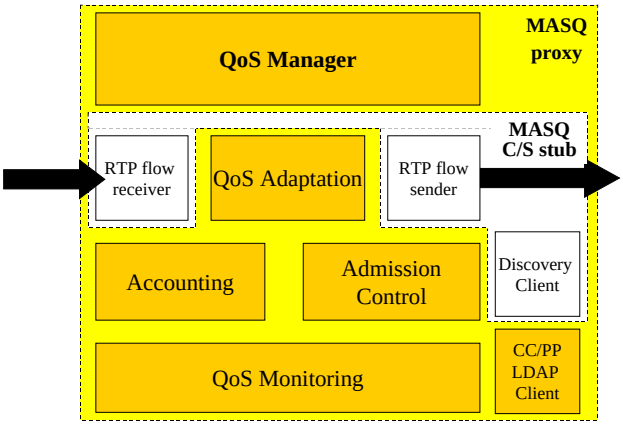

Figure 2. The MASQ modular architecture

3.2. The MASQ Proxy Architecture

MASQ proxies are the core components responsible for providing service tailoring and adaptation to user needs, to environment conditions and to communication link characteristics. As Figure 2 shows, the architecture of MASQ proxies is modularly designed and consists of different modules to provide different functions.

The *QoS Monitoring module* has the duty of observing the state of resources and services that are local to its hosting node. The module extracts and works on monitoring information about VoD flows from RTCP sender/receiver reports. It also achieves the visibility of Java Virtual Machine (JVM) indicators, e.g., method invocation and object/memory allocation for any Java thread, via the JVM Profiler Interface, an experimental API of the Java 2 platform for the monitoring of Java-based applications. In addition, this module monitors system indicators, e.g., CPU load, file system occupation, network packet collision rate, by exploiting the Java Native Interface to integrate with platform-dependent monitoring mechanisms [15]. Any variation in resource and system state produces the notification of events that can also be triggered by composing several low-level monitoring data. The module is in charge of event subscription and of notification even to mobile interested subscribers.

The *Admission Control module* maintains resource allocation information about all VoD flows currently served. Any entry include a unique identifier for the flow, a unique identifier for the VoD content, a tuple of QoS requirements associated with the flow, and the identifiers of previous and next MASQ components in the distribution path. Depending on information obtained from the QoS Monitoring module and on the current state of already accepted flows, the module has the responsibility of choosing the admission/denial of new service requests.

The *Accounting module* exploits the monitoring functions to keep a local log of the QoS level actually provided to the different receivers. It is in charge of authenticating users and associating them with the requested VoD flows and the corresponding accounting information. For any couple <VoD flow identifier, user identifier>, accounting data record possible modifications of QoS levels during provision. These data are stored in log files local to the MASQ proxies and can be processed off-line, for instance when billing users.

The *QoS Adaptation module* is responsible for any transformation of data depending on the negotiated QoS level. The module exploits JMF de/multiplexers, codecs and renderers, together with a set of ad hoc Java-based transcoding components. In addition, the module maintains buffers for current VoD flows to respond timely to incoming client requests in its locality and to pre-fetch the transcoding of served flows when needed. For any flow, it locally stores the version received with the maximum QoS level [11].

The *QoS Manager module* coordinates the other modules and decides the QoS levels for the MASQ components in the VoD path. During the negotiation phase, it combines QoS requirements from user/terminal profiles and reservation data from admission control. On this basis, it decides to enforce a specific point in the space of possible values for QoS parameters, e.g., frame rate, size and resolution. Usually, an interval for QoS parameters is permitted; the Manager module chooses at default the QoS point to maintain by minimizing local resource con-

sumption. However, service providers can also specify other different allocation policies for their administered resources, as shown in the following section. At provision time, the QoS Manager module can modify the provided QoS level by moving in the QoS space to maximize a cost function with weighted QoS parameters specified in user/terminal profiles. For instance, a device with limited display capabilities can specify a cost function with a frame rate weight larger than the frame resolution one, to indicate a preference in degradation of image quality instead of frequency decrease.

3.3. MASQ Proxies and Processors at Work

Let us describe how MASQ proxies and processors coordinate in a possible case of service provision scenario, as in Figure 3. Any client request for an active VoD service is served by one MASQ processor. The processor has the duty of finding and carrying the full information about QoS requested parameters and related profiles [11]. Once the whole data has been collected, the processor helps in establishing the active path, by involving all necessary proxies. MASQ processors migrate toward the available proxies in the locality and in close/connected domains to present there their carried request. It is up to proxies the decision phase by comparing requested QoS level and local resource availability.

If the MASQ proxy has direct local access to the VoD content with the proper QoS level, it behaves as the final VoD server in simple client/server architectures. After the proxies command the negotiated tailoring operations to the processors on the path, the VoD active service starts to flow, without any further proxy intervention.

If the VoD content is not directly available to the local proxy, then the establishment of the proper active path is passed to a neighbor MASQ proxy. The processor responsible for establishing the active path is cloned and forwarded to the next proxy. The forwarded processor can have knowledge of the previous path segments and can bring the history of previous choices. This propagation goes on until a successful match occurs between requested QoS levels and locally offered VoD contents. At this point, the whole active path has been successfully established, and all intermediate nodes host the needed MASQ components. Similarly to the chain of processors traversed by the flow, there is an analogous path of proxies that play only a control role.

Location awareness and knowledge of local monitoring information at provision time drive the adaptation when the agreed QoS level cannot be maintained. The processor-based distribution of QoS requirements throughout the whole active path permits optimal decisions avoiding further negotiations. Proxies have the duty of continuously monitoring currently offered QoS levels and of identifying possible deviations. We claim that locality is the key for prompt identification: as soon as a proxy ascertains a problem, i.e., any QoS parameter can no longer be granted, it commands a corrective action to the processor. The most common situation is a congestion point in the local path segment, with the corrective action of downscaling the VoD flow to a reduced quality provisioning.

At negotiation time, MASQ proxies have to distribute on all nodes of the active path. Two different processors (for user1 and user2) establish two partially overlapping active paths by migrating and cloning on any involved active node. It is the user1 processor that performs the VoD flow downscaling as specified in user1 terminal profile. At provision time, in case of degradation of $link_1$ bandwidth, proxies in domain1 and domain2 coordinate and command the user1 processor in domain2 to further reduce frame resolution of user1 VoD flow according to the receiver profile. In case of failure of $link_2$, a new path segment is established. The proxy in domain3 tries to identify a suitable server stub in close domains. Then, it starts a negotiation phase with the proxy of domain4 by cloning and migrating two new processors to this new domain.

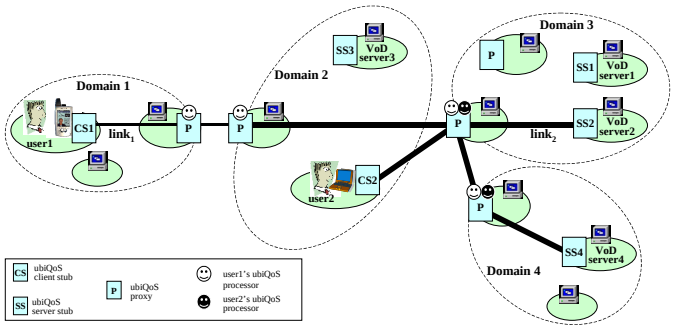

Figure 3. MASQ proxies and processors in a possible service provision scenario

4. Policy-based QoS Management in MASQ

MASQ currently provides a set of predefined QoS management strategies that address most common environment situations and user requirements. However, static QoS decisions built a-priori into the QoS Manager lacks flexibility and involves re-engineering efforts wherever there is an unexpected change. To overcome static strategies for QoS tailoring and control, MASQ administrators can represent and manage QoS adaptation requirements without hard-wiring them into QoS Managers. The ultimate goal is to reuse MASQ proxies in different environments and application scenarios without requiring modifications to the core architecture. This calls for solutions to dynamically define, install and possibly modify QoS management strategies tailored to the specific usage con-

text. In this direction, a key issue is the separation between QoS management rules and the code of MASQ components. This promotes dynamic QoS tailoring and adaptation with no impact on the QoS Manager implementation. We claim that policy-based approaches provide promising solution guidelines in these contexts of application [16, 17].

Policies can be defined as explicit rules governing choices in the behavior of a system, cleanly separated from the components in charge of their interpretation. Policies are typically expressed in a declarative way at a high abstraction level. They specify only the management tasks to perform and not how to achieve them, being details left to the run-time policy support [16]. MASQ policies can specify QoS management decisions that QoS Managers have to perform to tailor and adapt multimedia flows to client requirements and current system state. Both negotiation and adaptation can benefit from policy-based approaches: policies can specify how to allocate resources to address user requirements and can drive management operations for dynamic QoS adaptation.

Policy adoption for QoS management makes necessary the integration of MASQ with a policy-based framework. Our current effort is to develop a policy-enabled MASQ prototype that integrates a policy framework called Ponder and to make available a set of services for the support of automatic and transparent policy lifecycle, from policy initial distribution to policy activation and control. Ponder is a declarative object-oriented language for the specification of several types of distributed management policies. The Ponder framework allows administrators to represent QoS management decisions at a high level of abstraction [18]. The MASQ middleware only uses a subset of Ponder policies, i.e., obligation ones, typically event-triggered. Obligation policies are essentially declarative event-action-condition rules, defining the actions that policy subjects must perform on target objects when specific events occur and when specific conditions hold at event occurrence.

Let us introduce a simple example to illustrate Ponder obligation policies. Figure 4 reports two different policies used in the MASQ middleware with different goals. The P1 policy is used at negotiation time to support admission control management. In particular, P1 states that, when a service request is accepted (the event following the `on` field), the QoS manager (the `subject` field) has to choose in the permitted interval the QoS point that maximizes the amount of allocated resources (the `action` field), to express the management requirement of maximizing local resource usage. This is possible only if the number of currently served VoD flows is under a specified threshold (the `when` field). The P2 policy, on the contrary, is exploited at service provision time to command the QoS manager to request the QoS Adaptation module (the `target` field) to transcode flows from MotionJPEG to MPEG3 when the available bandwidth falls under a specified threshold.

```
inst oblig P1
 on ServiceRequest (VoD flow identifier, user identifier) ;
subject s = QoS Manager;
do s.allocateMaximumResources(VoD flow identifier)
    when #acceptedVoDflows()≤threshold
```

```
inst oblig P2
 on Decrease (bandwidth, threshold) ;
subject s = QoS Manager;
 target t = QoS Adaptation
do t.transcode(MPEG3)
```

Figure 4. Ponder obligation policies

We are currently testing the first prototype that enriches the MASQ proxy architecture with the following support services (see Figure 5):

- the *Policy Specification Service* (PSS), designed to enable the specification and compilation of Ponder obligation policies. PSS currently consists of a Policy Editor and a Policy Compiler. The Policy Editor handles the editing/deleting and browsing of policies. The Policy Compiler provides tools for parsing policy specifications and for the automatic translation of Ponder specifications into low-level policies to be interpreted in the Java programming environment. PSS is in charge of compiling, distributing and installing policies into the Enforcement Service. Any successive variation in the policies forces PSS to coordinates with the Enforcement Service to ensure a consistent and update vision. Because prevention of potential policy conflicts is essential for the success of policy-based management, we plan to extend the PSS with tools capable of supporting static policy analysis;
- the *Enforcement Service* (ES), targeted at supporting the parsing and activation of Ponder obligation policies when needed. For this purpose, we have implemented ES as the composition of two main logical modules, Coordinator and Executor. The Coordinator plays a key role at both initial policy specification and execution time. When policies are first defined, the Coordinator parses them and retrieves relevant information, such as their triggering events, actions and conditions. Relevant events are registered to the MASQ monitoring; then, policies are distributed and installed into Executors in charge of their interpretation. At service provision time, events can trigger the Coordinator to command corrective management to the Executor.

Let us note that all the QoS management operations performed by MASQ proxies can be either specified directly within the QoS Manager module code or expressed separately from the module code as Ponder obligation policies. In the former case, any change of QoS management requirements implies to suspend service provision and to update and re-compile the proxy code to reflect variations. On the contrary, in the latter approach, the evolution of

QoS management goals can be addressed by simply changing the set of high-level policy specifications without stopping the proxy execution. It is the underlying run-time policy support that transparently disables/enforces old/new policies when required and propagates changes to QoS Manager modules without impacting on their code implementation.

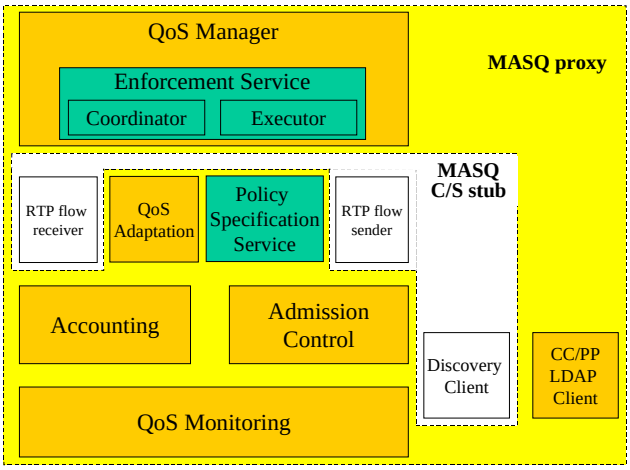

Figure 5. The policy-enabled MASQ proxy

5. Current Work

We are currently investigating some directions of work:

- how to avoid/resolve possible conflicts among different policies specified in different localities by different users, administrators and service providers;
- how to bind client requests with most proper service offerings by establishing most effective active paths in a completely decentralized and dynamic way, without assuming global knowledge of all resources available;
- how to guarantee a worst-case reaction time between modifications in distributed state and corresponding corrective actions triggered by the enforced policy.

Acknowledgements

Work supported by the Italian Ministero della Ricerca Scientifica e Tecnologica ("MUSIQUE: Infrastructure for QoS in Web Multimedia Services with Heterogeneous Access") and by the University of Bologna (Funds for Selected Research Topics: "An integrated Infrastructure to support Secure Services").

References

[1] J. Krikke, "Graphics Applications over the Wireless Web: Japan Sets the Pace", *IEEE Computer Graphics and Applications*, Vol. 21, No. 3, May/June 2001.

[2] P. Bellavista, A. Corradi, C. Stefanelli, "An Integrated Management Environment for Network Resources and Services", *IEEE Journal on Selected Areas in Communication*, Vol. 18, No. 5, May 2000.

[3] K. Psounis, "Active Networks: Applications, Security, Safety, and Architectures", *IEEE Communications Surveys*, Vol. 2, No. 1, 1999.

[4] H. Yasuda (ed.), *2nd Int. Working Conf. Active Networks (IWAN'00)*, Springer-Verlag LNCS, Japan, Oct. 2000.

[5] R. Koster and T. Kramp, "Structuring QoS-Supporting Services with Smart Proxies", *IFIP/ACM Int. Conf. Distributed Systems Platforms (Middleware 2000)*, Springer-Verlag LNCS, Apr. 2000.

[6] W. Marshall, C. Roadknight, "Provision of Quality of Service for Active Services", *Computer Networks*, Vol. 36, No. 1, June 2001.

[7] M. Baldi, G. P. Picco, F. Risso, "Designing a Videoconference System for Active Networks", *2nd Int. Workshop on Mobile Agents (MA'98)*, 1998.

[8] E. Amir, S. McCanne, R. Katz, "An Active Service Framework and its Application to Real-time Multimedia Transcoding", *ACM SIGCOMM Conf.*, 1998.

[9] F. Kon, R. H. Campbell, S. Tan, M. Valdez, Z. Chen, J. Wong, "A Component-based Architecture for Scalable Distributed Multimedia", *14th Int. Conf. Advanced Science and Technology (ICAST'98)*, 1998.

[10] F. Kon, R. H. Campbell, K. Nahrstedt, "Using Dynamic Configuration to Manage a Scalable Multimedia Distribution System", *Computer Communications*, Vol. 24, No. 1, Jan. 2001.

[11] F. Baschieri, P. Bellavista, A. Corradi, "Mobile Agents for QoS Tailoring, Control and Adaptation over the Internet: the ubiQoS Video-on-Demand Service", to be published in *2nd IEEE Int. Symposium on Applications and the Internet (SAINT'02)*, Japan, Jan. 2002.

[12] T. Braun, "Internet Protocols for Multimedia Communications - Part II: Resource Reservation, Transport, and Application Protocols", *IEEE Multimedia*, Vol. 4, No. 4, Oct. 1997.

[13] D. Chalmers, M. Sloman, "A Survey of Quality of Service in Mobile Computing Environments", *IEEE Communications Surveys & Tutorials*, Vol. 2, No. 2, 1999.

[14] A. T. Campbell, "QoS-aware Middleware for Mobile Multimedia Communications", *Multimedia Tools and Applications*, Vol. 7, No. 1-2, 1998.

[15] P. Bellavista, A. Corradi, C. Stefanelli, "How to Monitor and Control Resource Usage in Mobile Agent Systems", to be published in *3rd IEEE Int. Symp. on Distributed Objects and Applications (DOA'01)*, Italy, Sep. 2001.

[16] R. Wies, "Policies in Network and System Management – Formal Definition and Architecture", *Journal of Network and Systems Management*, Vol. 2, No. 1, 1994.

[17] M. Sloman, "Policy Driven Management For Distributed Systems", *Journal of Network and Systems Management*, Vol. 2, No. 4, 1994.

[18] Imperial College - Ponder, http://www-dse.doc.ic.ac.uk/research/policies/software/