A QoS-Aware Architecture for Multimedia Content Provisioning in a GRID Environment

Fabrizio Messina, Giovanni Novelli, Giuseppe Pappalardo, Corrado Santoro, Emiliano Tramontana
Dipartimento di Matematica e Informatica
Università di Catania, Viale A. Doria, 6 - 95125 Catania, Italy
{messina, novelli, pappalardo, tramontana}@dmi.unict.it, csanto@diit.unict.it

***Abstract*— In this paper, we propose a mobile agent infrastructure to support "Quality of Service" (QoS) parameters in multimedia content streaming. The infrastructure is able to face the issues concerning multimedia content transformation, in order to ensure that the client-side QoS is met. A Grid computing platform is exploited and content adaptation is performed through appropriate agents that are allocated as jobs and executed on Grid hosts. A location service is therefore devised and made available within the Grid system, which helps finding the hosts whose geographic position and network connections minimise the delays required to move the desired multimedia content from the storage to the processing site, and (after suitable transcoding) from this to the requesting client.**

Keywords: mobile agents, agent-based applications, GRID, QoS, Globus.

I. Introduction

Nowadays a large number of multimedia coding formats exists. Each format not only needs a proper decoder/player at the client side but also a different transport protocol featuring timing characteristics that require a network offering mechanisms to guarantee throughput, latency, packet loss ratio, jitter, etc. [2], [14], [17]. Such "Quality of Service" (QoS) requirements [10], if not fulfilled, can cause delays or interruptions in content playing and thus result in a service whose quality could be lower than expected.

As for coding scheme, even if some well-known general-purpose players, like Windows^{TM} Media Player or Xine, are able to support the majority of multimedia formats, there are still some schemes (e.g. QuickTime^{TM} MOV or Realplayer^{TM} RAM) that require their own players. Moreover, some players are not able to adapt the content to the quality of the connection used by the client, which could be too slow for the selected multimedia file, thus preventing proper reproduction. As a result, sometimes a multimedia content cannot be retrieved, played and used at client side, because the player is not available, the protocol is not supported, the client connection has not enough bandwidth or client performance is inadequate.

The reason behind this is due to the fact that current multimedia provisioning solutions give the client the responsibility of adapting the received content to both client and network capabilities. This may well be deemed unreasonable if we think that, in general, servers have more computational power and performances than traditional desktops or laptops, so having them adapt the provided content to each client seems more appropriate. Moreover, when multimedia provisioning is not performed by single servers but a *Grid computing environment* is employed [5], server-side on-line adaptation of streaming becomes not only feasible but also preferable: thanks to the ability of managing and offering a huge amount of storage space and CPU power, a Grid can be used not only to store and provide multimedia contents, but also to perform multimedia transcoding in order to satisfy at best the QoS parameters requested by the user, taking also into account network capabilities.

By considering the issues and goals outlined, this paper proposes a software architecture, based on mobile agents running in a computational Grid, for multimedia content provisioning and QoS satisfaction[1]. The main aspect of the proposed solution is that it is able to share the content adaptation cost between the client and the server, thus overcoming the problems described. A location service is also included, which aims at finding the host whose geographic position and network connections capabilities are able to minimise the time required to move the desired multimedia content *(i)* from the storage to the processing site, and *(ii)*—after suitable transcoding—from this to the requesting client.

The paper begins (Section II) with a description of a use-case that serves as a reference scenario to introduce the basic model of the solution. Then Section III illustrates the software architecture of the solution as integrated in a Grid environment, thus explaining how the main services of a Grid can be exploited for our purpose. Section IV describes the prototype implementation of the architecture, which has been developed, using GridSim. Section V discusses and compares other approaches. Section VI concludes the paper.

II. Use-Case Scenario and System Model

We consider a scenario in which a user asks for the retrieval and playing of a multimedia content by means of a client program or a web interface. We suppose that the client/player, when it connects to the server, specifies some *Quality of Service (QoS)* parameters regarding its capabilities, such as encoding scheme it is able to support, the speed of the connection used, etc. In particular, two sets of QoS parameters

[1]This work has been (partially) carried out within the Web-Minds, QUASAR and PI2S2 projects, funded by the Italian Ministry of Education, University and Research (MIUR), and the TriGrid VL project, funded by Regione Sicilia.

Application QoS	Network QoS
Encoding Scheme	End-to-end Throughput
Compression Scheme	End-to-end Throughput Jitter
Compression Ratio	Propagation Delay
Sampling/Frame Rate	Propagation Jitter
Picture Size	Packet-loss ratio
Colors	
Channels	

TABLE I

APPLICATION AND NETWORK QOS PARAMETERS

can be considered, which are summarised in Table I and explained in the following:

- **Application QoS parameters.** They are those parameters concerning application-level multimedia handling and include *encoding scheme*, *compression scheme* and *compression ratio*, for all types of contents, *sampling rate*, *sampling size* and *number of channels* for audio contents, and *frame rate*, *picture size* and *number of colors* for video data.
- **Network QoS parameters.** They refer to the capability of the network and subnetworks located on the path from the client to the server. Such parameters include the *end-to-end throughput*, the *propagation delay*, the *propagation jitter*, etc.

When the server receives a client's request, it has to try to fulfill it by matching the parameters specified with those of the multimedia content to be retrieved, also considering the capabilities of the network that has to carry the streamed data: if they differ or the network is not able to deliver the content according to the requirements, an adaptation would be needed, the complexity and feasibility of which depend on how much requested and offered contents differ from each other, and the type of conversions they imply.

Following such a use-case scenario, our solution aims at proving an efficient and reliable way to perform content transcoding and adaptation in order to fulfill at best the requirements of a client program. The basic model of the solution, which is based on *mobile agent technology*, exploits two agents, called `ClientProxy` and `ServerProxy`, running on the client and the server respectively, which act as "proxies" intercepting and adapting multimedia data. Basically (see Figure 1), the `ClientProxy` is designed as able to talk with the client application/player; it knows client's abilities in handle multimedia data and thus its application QoS parameters. On the other hand, the `ServerProxy` is able to handle both client's application QoS and the format of the multimedia content requested. Both agents start at the client side and thus both knows how to support the player characteristics, but the `ServerProxy` is a mobile agent; when the content streaming has to begin, the `ServerProxy` migrates to the server and both agents start to collaborate as follows:

1) The `ServerProxy` agent retrieves multimedia content and analyses it.
2) Both agents, according to the application QoS parameters of both the client and the content, and knowing the capabilities of the transfer link (network QoS parameters), establish an *inter-agent protocol* in order to transfer the multimedia content. Therefore, the `ServerProxy` performs the appropriate conversions and QoS adaptions to allow data transmission through the network with the negotiated inter-agent protocol.
3) The `ClientProxy` agent receives the data using the inter-agent protocol and performs the final conversions in order to supply data to the player with application QoS parameters requested.

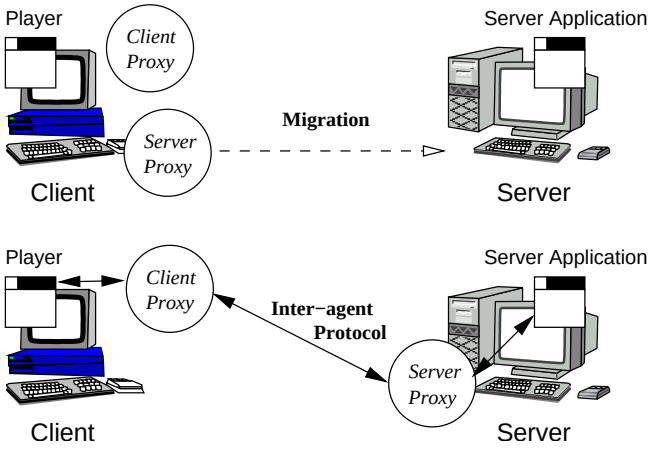

Fig. 1. Working Scheme of the Solution

Roughly speaking, this solution provides a significant advantage since it (basically) is able to avoid QoS loss. In fact, traditional solutions are based on some general purpose QoS translation algorithms running at server side, but such algorithms are not able to cover all the possible cases needed by a client. On the other hand, in the proposed solution, the transcoding part running in the server—the `ServerProxy` — originates from the client machine: it *exactly* knows what are the client's requirements and thus can encapsulate the right algorithms to perform a correct transcoding.

A significant drawback of the solution is however the computational power needed, at server side, to perform transcoding in the case of many concurrent requests. To face such an issue, we consider our solution as running on a Grid infrastructure since it can provide the power needed; to this aim a suitable architecture has been designed and detailed in the following Section.

III. RUNNING TRANSCODING AGENTS IN A GRID ENVIRONMENT

For obtaining the transcoding of a multimedia content, agents are allocated and executed in capable hosts of a Grid system. The most widespread software system used for Grid environment is the Globus Toolkit (GT) [13]. This includes services to access and monitor computing and data resources, enforce security, allow users to send and execute their applications. Computing resources have a specialised role according to their characteristics. Hence, a host providing a large amount of storage space is a *Storage Element (SE)*; a host holding a

certain number of CPUs that can execute jobs is a *Computing Element (CE)*; whereas users are able to log into GT, send their commands and receive results, by means of a host called *User Interface (UI)* [6], [4].

In order to determine the host where the transcoding agent should run it is appropriate to take into account: (i) the location of the agent, (ii) the location of the user requesting the multimedia content, and (iii) the location of available hosts.

The services necessary for performing this selection are provided by the architecture components described in the following (see Figure 2).

In order to make the multimedia contents available to users, these are initially transferred into available SEs. Each content is characterised by its name, encoding, length, description, size, etc.

A user requests a multimedia content, by running a client software on his/her UI. The request is characterised by: (i) the name of the multimedia content or its description, and (ii) the needed encoding format.

The request is processed by service *MuM* (for *Mu*ltimedia *M*ining) that finds available versions of the content. When transcoding from an available format to the requested version is required, other services are involved in order to: find an appropriate CE that will host one or more capable agents, find and transfer both the source version of the content and the transcoding agent into the selected CE. The output of the transcoding agent, which is a new format for the initial multimedia file, is then provided to the user on his/her UI and also stored to some SE so as to better serve further requests for the same file and for the newly available format.

A. Services for Selecting Hosts

The service *MuM* is able to find the locations of a user requested multimedia content (see in Fig. 2 interactions (1, 2, 3)). *MuM* returns the location, in terms of SE address, where each format of the content can be found. *MuM* is organised as a repository based on the existing Grid Resource Information Service (GRIS) [3][2]. *MuM*'s data are updated when new contents are inserted, new formats are available, and existing ones are re-organised (e.g. moved to better support accessing them).

Service *Agent Locator (AL)* is responsible to store the location of transcoding agents. While agents are initially stored only on the users' hosts, they are re-located to several CEs according to the requests for transcoding the multimedia contents. To minimise the re-location overhead, agents are kept on the CEs even when idle, i.e. after contents processing has finished. Service *AL* traces where each agent is, thus when *AL* is queried for a given transcoding agent (see (4) in Fig. 2), it returns the list of CEs addresses where that agent is located.

Service *CE Locator (CEL)* traces the availability and CPU capacity of CEs. This service initially asks known GRIS about data concerning the number of hosts in a CE, and the characteristics of the hosts, in terms of CPU type and amount of RAM. Additionally, *CEL* periodically asks the length of the job queue on a CE. All these data are asked asynchronously from a user query and stored locally by *CEL*. This helps reducing the performance penalties given by the interactions with GRIS. When an agent has to be allocated, *CEL* is asked for available hosts (see (5) in Fig. 2) and returns a list of CE addresses, whose hosts are idle or lightly loaded.

Finally, service *Resource Finder (ReF)* selects a CE for hosting transcoding agents. In selecting a CE, *ReF* tries to minimise latencies due to the network and achieve the shortest response time for having the adapted content. Latencies are calculated according to the following three "distances": (i) the "distance" $d(SE_i, CE_j)$ between the SEs where the needed content is stored (provided by MuM) and the available CEs (provided by CEL); (ii) the "distance" $d(CE_j, UI)$ between available CEs and UI; and (iii) the "distance" $d(CE_j, A(H_k))$ between the available CEs and agent A that can be found in one of several hosts H_k (the locations where agent A is found is provided by AL).

Each "distance" is measured as the number of seconds necessary to deliver 100Mb across two end points, thus it takes into account both the available bandwidth and the latency, i.e. the physical space that has to be crossed. Measures between each two pairs of known sites are performed off-line and stored.

The selected sites $SE_i^{'}$, $CE_j^{'}$, representing, respectively, the host from which the source content will be loaded and the host allocating the transcoding agent are those that minimise

$$d(SE_i, CE_j) + d(CE_j, UI)$$

for i, j ranging over the available sets of SE_i, CE_j.

Once $CE_j^{'}$ has been determined, the value $d(CE_j^{'}, A(H_k))$ is minimised, i.e. the host $H_k^{'}$ is chosen from which the agent will be transferred.

Note that *ReF* collects the necessary data and determines the CE that should host the transcoding agents, the CE providing such an agent, and the SE where the source multimedia content is stored. In Fig. 2, they are CE_1, CE_2 and SE_2, respectively.

In order to activate multimedia content transformation and delivery, *ReF* submits a job to the selected CE, e.g. CE_1, (see (6) in Fig. 2). This job executes the following set of steps. Firstly, it collects the necessary parts, i.e. the transcoding agent from CE_2 (6.1) and the multimedia content from SE_2 (6.2). Secondly, it executes the agent that will process the content. Finally, the result of content transformation is directly passed on to the user front-end.

B. Handling QoS for Multimedia Contents

In order to have a short response time between a user request for a multimedia content and the transcoded version, the capabilities of the supporting infrastructure, i.e. available bandwidth and computing "power" have to be determined beforehand.

The preferences set by a user in terms of frame size and coding format for a multimedia content affect the activities that are performed by the infrastructure to serve the user request.

[2]GRIS is part of the Monitoring and Discovery Service (MDS) provided by the Globus Toolkit.

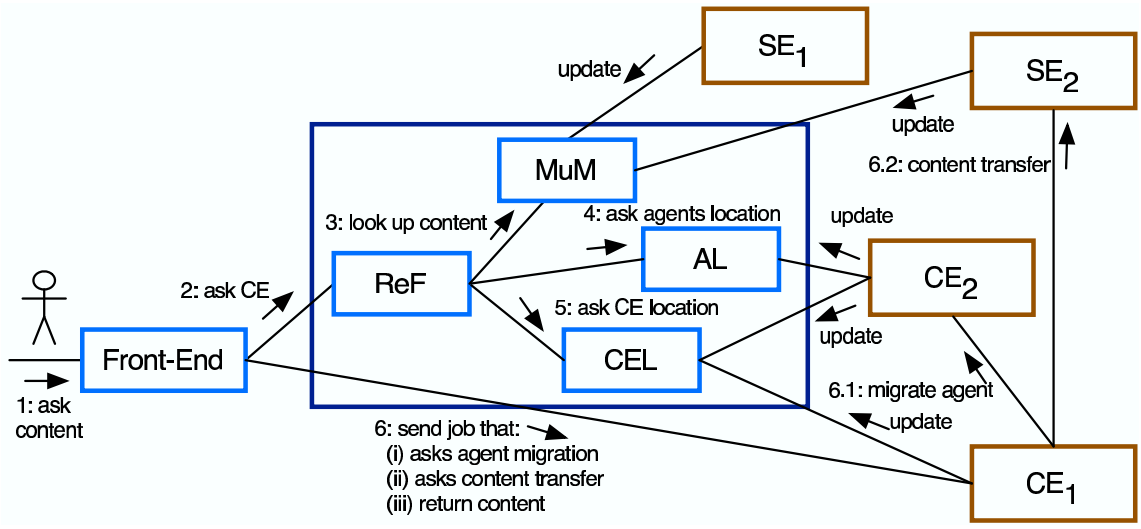

Fig. 2. Overview of interactions between front-end and services.

One of these activities consists in the multimedia processing, another one is the a priori reservation of bandwidth and hosts.

At least one capable host will be reserved to run the transcoding agent. However, just one host could be not sufficient for the necessary processing. When the estimation of the time needed to sequentially transcode the content on a single host would result in an output rate that is less than the rate necessary to ensure that the content can be seen/listened smoothly, then a parallel transcoding activity is organised. For this parallel processing, the initial multimedia content is fragmented and each chunk processed by an agent on a dedicated host.

Let us first express the way we estimate the time needed to transcode a content from a format to another.

We have processed several videos having different resolution (see Table II and III) and observed that processing time is mainly related with the frame resolution and size in bytes of the frames to be processed. I.e. for the same video, having e.g. a resolution of 320x240 pixels, the processing time is almost proportional to the size (in bytes) of the JPEG frame that has to be transcoded.

TABLE II

CHARACTERISTICS OF SAME SAMPLE VIDEOS.

content	source	resolution	length (s)	size (MB)
chinese	AVI	320x191	152	20.3
wr	AVI	320x240	14	1.0
cartoon	AVI	480x272	115	7.6
dhl	AVI	640x480	43	17.7

In Table III, we report for several videos (whose characteristics are found in Table II) the time in milliseconds necessary to convert from AVI to MPEG, MJPEG, h263p and rv10 (see the 4 right-most columns, respectively) several chunks of the content lasting 2 seconds each and having 50 or 60 frames, as indicated in the column frames. Transcoding is performed from the set of frames (each stored as a JPEG file) and whose overall size in megabytes is indicated in column size.

TABLE III

TRANSCODING TIME FOR SOME SAMPLE VIDEOS.

content	frames	size	MPEG4	MJPEG	h263p	rv10
chinese	1– 50	0.27	87	79	107	110
	51–100	1.51	181	156	179	181
	101–150	1.25	182	139	180	183
	151–200	1.35	191	148	189	188
wr	1– 50	2.28	247	212	244	238
	51–100	1.69	204	180	200	206
	101–150	1.76	204	186	205	202
	151–200	1.55	193	170	193	194
cartoon	1– 60	1.76	240	240	230	250
	61–120	1.64	280	260	300	290
	121–180	3.32	380	340	370	380
	181–240	4.10	470	400	450	450
dhl	1– 60	8.00	935	894	901	891
	61– 120	6.77	853	738	835	842
	121– 180	6.64	808	733	815	806
	181– 240	8.95	956	905	953	923

The reported values are just an excerpt of the experiments we have performed. In other experiments we have converted the whole reported video, moreover other videos have been processed in a similar way.

The estimation function for the time needed to transcode into MPEG4, but the same applies to the other output formats, is the trend function calculated according to the values stored in a database of observed times. The resulting trend function takes as input the resolution and frame size as parameters and returns the estimated time.

In order to calculate the number of hosts needed for the transcoding, we compare the estimated transformation time and the length in seconds of the processed video. When transformation time is greater than processed video then we must use more than one host for the transformation and the

number of needed hosts is the minimum natural number bigger than the ratio between transformation time and length.

Bandwidth reservation aims at readily transfer data from a disk to the selected CE that will perform transcoding and from the CE to the user device.

The amount of bandwidth necessary is calculated as the ratio between size of the content and length (this is just an approximation since the amount of bytes per frame are not constant for the whole video).

IV. Simulating Environment

In order to assess the timing characteristics of the proposed software infrastructure and how it reacts when a large number of users send requests for accessing multimedia contents, we are in the process of developing a model of the proposed infrastructure using the simulation environment GridSim [8]. GridSim is a Java toolkit that can be programmed to simulate a Grid system, thus it offers support, in terms of classes, for simulating CEs, SEs, Virtual Organisations, etc.

In our proposed infrastructure, transcoding agents can move and communicate within a Grid environment. The underlying support for allowing such agents to move, communicate, etc. is an agent-platform, such as Jade [1]. For simulation purposes, on top of GridSim we have developed a class library that models the activities of an agent platform (similar to Jade). Such a library, called *GAP (Grid Agent Platform)*, includes the classes: AgentPlatform, AgentSite, and Agent.

Class AgentPlatform represents a federation of sites offering facilities that allow agents to move, find each other, communicate, etc. This class allows the access to services that are expected from an agent platform, such as: (i) a registry informing about available agents on any site, simulated as class DirectoryFacilitator; and (ii) an observer for all the movements of agents, simulated as class NetworkMonitor.

Class AgentSite is responsible to hold data about the location and the state (e.g. running, stopped, idle) of existing agents in a given CE within the Grid system. AgentSite extends GridSim class GridSim and holds an instances of GridSim class GridResource, which represents the CE.

Class Agent represents an agent and holds data about the identifier, state, type, and size of the agent. Each Agent instance simulates the computation performed by an agent, which can be affected by events that are notified to it through messages. The events that can affect an agents are: stop, resume, terminate, and migrate. Messages are exchanged between agents while performing their computation. Each time an agent migrates or change its state, both classes DirectoryFacilitator and AgentSite are informed. An Agent instance simulates some processing by generating jobs and submitting them into the CE where the agent is located. A generated job is an instance of GridSim class Gridlet.

The network connections between CEs and SEs are modelled according to GridSim class Link. Instances of such a class are set with values representing bandwidth and delay of the network link, as well as the instances of GridResource it connects.

An additional User class models the user requests, thus the ability to ask for a content and receive a result.

Along the above classes, all the services of the proposed infrastructure (i.e. ReF, MuM, AL and CEL) are modelled as classes that execute on the Grid system, process requests and provide corresponding results.

For running a simulation, the number of CEs, transcoding agents types and users are set. The number of instances of AgentSite created is according to the the given number of CEs chosen. Each AgentSite instance is set with a CE identifier. Each type of transcoding agent is modelled as an instance of Agent. Several instances of Agent are created and each is set with: a value for the size of the code of the agent (in bytes); its initial location, using an instance of AgentSite; and the state of the agent, as idle. Finally, instances of User are created.

Once a simulation is executed, GridSim output is a report that for each modelled class describes the number of received requests, the amount of time necessary to process the requests, the time spent waiting for replies, etc. This report will provide valuable feedback on all the assumptions concerning the expected time of network connections to transfer multimedia contents, hosts to process fragments of contents, infrastructure to select hosts for processing, etc.

V. Related work

Several existing approaches, including [7], [15], generate off-line different transcoded versions of a multimedia content, i.e. before users request a format this is prepared and stored into the disk. Thus serving a request is just a matter of reading the content from the disk. Of course, generating different formats for a given content requires a large amount of computing resources, thus these approaches recur to parallel processing, using clusters of hosts, for the initial content.

An approach for the on-line transcoding of multimedia contents by means of software system for a Grid environment is proposed in [16]. In such an approach, a broker component finds and dynamically allocates transcoding jobs on available Grid resources, according to the incoming requests. Similarly to our approach, the broker component selects capable and unloaded host for ensuring a short response time. However, we additionally take into account both static and dynamic network conditions. Moreover, in this approach the server hosts have to be equipped with all the necessary transcoding libraries beforehand. In contrast, our approach employs agents that move to available hosts, thus we do not need to install transcoding support into all the Grid hosts.

On-line transcoding is also faced in [11], where the authors propose a support for fitting multimedia content into mobile devices connected to a wireless network. Transcoding, which is based on changing frame size, color depth and Q-scale, is performed inside a HTTP proxy, therefore the approach is suitable to adaptation of video content that could be found on the web. In contrast to our proposal, this approach considers only some parameters that are fixed at server-side and does not perform automatic parameter adaptation on the basis of network connection monitoring. Moreover, in our approach

agents are dispatched from the client, they are aware of the target player's capabilities and are thus able to adapt the content at best, taking also into account the varying communication conditions.

When having to serve requests for contents, *Content Delivery Network* (CDN) [9] can bring benefits in terms of efficient delivering, since contents are replicated and relocated by the CDN according to request types and origins. Our approach is aimed not only at finding the nearest location for the requested multimedia content, but especially for on-the-fly generation of new a content format through transcoding according to client requirements.

Adaptive Web Systems (AWS) [12] provide unaware users with customised versions of contents, selected at server side on the basis of user profile, location, access mode, terminal, etc. In contrast, our approach suggest that clients have an active role in the content adaptation process, i.e. they are represented by their agents for this purpose. In principle, this permits more sophisticated and dynamic forms of adaptation, but of course requires a supporting infrastructure and a more capable and aware client.

VI. Conclusions

This paper has presented a mobile agent architecture for multimedia content provisioning and QoS adaptation which exploits a computational Grid, for its CPU power and storage space availability. The basic working scheme of our proposal entails two mobile agents, one at client side and the other at server side, what cooperate with each other to perform multimedia adaptation and transcoding, thus aiming at adapting the QoS of the content to the QoS requested by multimedia player.

The architecture consists of a set of components entailed with the task of finding *(i)* the storage element that possesses the multimedia content to be played and *(ii)* the computing element where to host the mobile agent in charge of server-side transcoding. To this aim, some metrics are introduced, intended to minimize the distances between user and the computing element, and between the latter and the storage element where the multimedia content is stored.

In order to evaluate the feasibility of our solution, numerous tests have been performed to determine the time taken by various types of transcoding. A prototype implementation, running with a simulation tools, has also been developed, in order to evaluate not only the feasibility of the solution but also its performances.

As a future work, our goal is to package the system as a Globus Toolkit extension, so as to obtain a complete and standard solution.

References

[1] F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Software - Practice and Experience*, 31(2):103–128, 2001.

[2] Anjali Bhargava and Bharat Bhargava. Measurements and Quality of Service Issues in Electronic Commerce Software. In *Proceedings of IEEE International Conference on application-specific Software Engineering and Technology (ASSET-99)*, Texas, March 24-27 1999.

[3] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proceedings of the 10th IEEE Symposium On High Performance Distributed Computing*, 2001.

[4] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, June 22 2002.

[5] Ian Foster and Carl Kesselman, editors. *The Grid (2nd Edition): Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[6] Fabrizio Gagliardi, Bob Jones, Mario Reale, and Stephen Burke. European DataGrid Project: Experiences of Deploying a Large Scale Testbed for E-science Applications. *Lecture Notes in Computer Science*, 2459, 2002.

[7] F. Gonzalez-Castano, R. Asorey-Cacheda, R. Martinez-Alvarez, E. Comesana-Seijo, and J. Vales-Alonso. Dvd transcoding with linux metacomputing, 2003.

[8] Grid Computing and Distributed Systems (GRIDS) Laboratory. *GridSim: A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing*, 2005. http://www.gridbus.org/gridsim/ .

[9] Markus Hofmann and Leland R. Beaumont. *Content Networking*. Morgan Kaufmann, 2005.

[10] ITU-T (CCITT). Recommendation. Terms and Definitions Related to The Quality of Service and Network Performance including Dependability, August 1994.

[11] Ricardo Oliveira Parixit Aghera, Advait Dixit and Vidyut Samanta. Wireless middleware: Dynamic video transcoding. In *Communication Systems Software and Middleware*, New Delhi, India, jan 2006.

[12] Mike Perkowitz and Oren Etzioni. Adaptive web sites. *Communication ACM*, 43(8):152–158, 2000.

[13] The Globus Alliance. Home page and publications. http://www.globus.org .

[14] Andreas Vogel, Brigitte Kerhervè, Gregor von Bochmann, and Jan Gecsei. Distributed Multimedia and QoS: A Survey. *IEEE Multimedia*, 2(1):10–19, 1995.

[15] Gord Watts and Steve Forde. *Grid Computing: A Practical Guide to Technology and Applications*, chapter Grid Enabling Software Applications, pages 252–263. Charles River Media, Hingham, MA, 2003.

[16] Angelo Zaia, Dario Bruneo, and Antonio Puliafito. A scalable grid-based multimedia server. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE)*, pages 337–342, 2004.

[17] John A. Zinky, David E. Bakken, and Richard D. Schantz. Architectural Support for Quality of Service for CORBA Objects. *Theory and Practice of Object Systems*, January 1997.