Designing and Evaluating An Active Grid Architecture

F. Bouhafs[a], B. Gaidioz[a], J.P. Gelas[a], L. Lefèvre[a], M. Maimour[a], C. Pham[a b], P. Primet[a], B. Tourancheau[c]

[a]LIP Laboratory (UMR 5668 CNRS - ENS Lyon - INRIA - UCB Lyon), RESO Team, Lyon, France

[b]University Lyon 1, France

[c]Sun Labs Europe, Grenoble, France

Today's computational grids are using the standard IP routing functionality, that has basically remained unchanged for 2 decades, considering the network as a pure communication infrastructure. With the grid's distributed system point of view, one might consider to extend the *commodity Internet*'s basic functionalities. Higher value functionalities can thus be offered to computational grids. In this paper, we report on our early experiences in building application-aware components and in defining an active grid architecture that would bring the usage of computational grid to a higher level than it is now (mainly batch submission of jobs). To illustrate the potential of this approach, we first present how such application-aware components could be built and then some experiments on deploying enhanced communication services for the grid. We will show how reliable multicast and QoS mechanisms could deploy specific services based on the grid application needs.

1. Introduction

The simplest perception one has of a computational grid [14] is a pool of geographically distributed computers that can be accessed and used in a structured manner to solve a given problem. Such grid infrastructures are foreseen to be one of the most critical yet challenging technologies to meet the exponentially growing demands for high-performance computing in a large variety of scientific disciplines: high energy physics, weather prediction, mathematics and combinatorial problems, genome exploration, etc. In the past few years, many software environments for gaining access to very large distributed computing resources have been made available (e.g. Condor [21], Globus [15], Legion [18] to name a few). National and international projects have been launched all around the world to investigate the potential of grid computing: DataGrid (www.eu-datagrid.org), EuroGrid (www.eurogrid.org), GriPhyn (www.gryphyn.org), PPDG (www.ppdg.net), DAS (www.cs.vu.nl/das2) to name some of them.

Today's computational grids are using the standard IP routing functionality, that has basically remained unchanged for 2 decades, considering the network as a pure communication infrastructure. With the grid's distributed system point of view, one might consider to extend the *commodity Internet*'s basic functionalities. Higher value functionalities can thus be offered to computational grids. In that sense, these ideas are very similar to those of the peer-to-peer (P2P) community (with emerging popular P2P applications such as Napster or Gnutella) and web services. Going a step further, we want to embed more generally in the network infrastructure specific functionalities for computational grids and their applications. Therefore application-aware components (AAC) are introduced as opposed to traditional Internet routers. Such AACs will be distributed at the edge of the grid networking infrastructure and will be based on active networking technologies[1] that allow specific program codes

[1]Active networks/grid and P2P both allow the notion of having some type of application-specific processing "in-line" with communication, and that can take advantage of the actual communication topology. For this particular use ("in-line" processing), the primary difference between them is the implementation approach. P2P systems are typically user-level processes. The original active network concept was to put application-specific processing in the

to be uploaded into routers to perform specific functions related to the applications on the data flows.

The ideas of active elements and application-aware components is not new and some ideas have been taken from other areas. For example, using edge nodes to improve bandwidth, latency, resiliency, etc, has been previously proposed in [31,2,38,19,41,34]. However, the application in grids is both novel and practically useful. The resulting active grid architecture proposes solutions to implement the two main kinds of grid configurations : meta-cluster computing and global computing. In this architecture the network takes part in the grid computing session by providing efficient and intelligent services dedicated to grid data streams transport: caching, monitoring, filtering, interest management, reduction, election, etc.

In this paper, we report on our early experiences in building such application-aware components and in defining an active grid architecture that would bring the usage of computational grid to a higher level than it is now (mainly batch submission of jobs). We present how a high performance execution environment can be built and how dynamic services can be deployed. To illustrate the potential of this approach, we also present some experiments on deploying enhanced communication services for the grid. We will present how reliable multicast and QoS mechanisms could deploy specific services based on the grid application needs. The paper is organized as follows. First we present the AAC technologies and the active grid architecture. Then Section 3 presents the high-performance execution environment for building an active grid infrastructure that meet the current Gb/s requirements. QoS and reliable multicast on an active grid infrastructure are presented in Section 4. Section 5 concludes.

hardware router. It is much easier and practical, however, to experiment with active networks built as user-level processes. Hence, the difference between active networks and P2P becomes even less. Please refer to [12] for more details on grids and P2P systems.

2. Active Grid Architecture

There can be a large variety of grid infrastructures but in most cases they use a similar network architecture: local computing resources are connected together using any kind of local/system area networks (Fast/Giga-Ethernet, Fiber Channel, SCI, Myrinet, etc.) and gain access to the rest of the world through one or more routers. So we will mainly assume that the computing resources are distributed across an Internet-based network with a high-speed backbone network in the core, typically the one provided by the Telco companies, and several lower-speed[2] access networks at the edge.

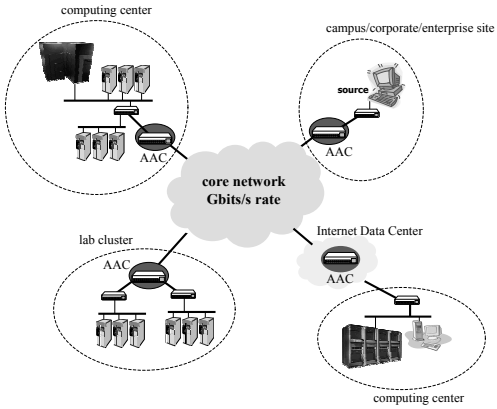

Figure 1. An application-aware grid infrastructure.

Figure 1 depicts such a typical grid infrastructure. For simplicity we represented an access network by a router but practically such networks would contain several routers. We propose an active network architecture dedicated to grid environments and grid applications requirements. Such an active grid architecture is based on a virtual topology of AACs deployed at the edge accesses. AACs nodes are connected between each

[2]With respect to the throughput ranges found in the backbone, say up to 1 Gbit/s.

other and each AAC manages communications for a small subset of grid nodes. Grid data streams can cross various AACs towards the passive backbone and then cross another set of AACs to the receiver node.

2.1. Application-aware component technology

The AACs concept calls for an infrastructure of network elements capable of executing specific processing functions (called services) on data flows, and this mostly on-the-fly. It is possible to build such an infrastructure by involving only end-hosts: it is the peer-to-peer paradigm. However, although this scheme may work, even on large scale systems such as demonstrated by the recently general public P2P applications, there are a number of drawbacks to this approach: redundancy of functionalities, high end-host overhead, limited range of applications, etc.

Recently, a disruptive technology called "active networking" proposes a new vision for the Internet that involves routers (so-called active routers) in the processing of data packets. In active networking [36], routers can execute, through an execution environment similar to an operating system (such as the TAMANOIR system described in this paper later on), application-dependent functions on incoming packet. There are many difficulties, however, for deploying in a large scale an active networking infrastructure. Security and performance are two main difficulties that are usually raised. However, active networking has the ability to provide a very general and flexible framework for customizing network functionalities in order to gracefully handle heterogeneity and dynamic, key points in a computational grid.

In order to implement the AAC concepts, there is no need to go for a complete active networking scheme with an open architecture and the many security pitfalls. It is possible to adopt an operator's perspective which consists in deploying some well identified services (similar to any protocol supported by a router nowadays), chosen for their generic approach. However this approach needs a high amount of standardization thus only very generic and core services are likely to be deployed.

2.2. Deployment issues

AACs could be hosted in Internet Data Center (deployed by an operator) but this solution is very difficult at this time because grid computing is not yet the main concern of telecommunication operators or Internet Service Providers. The other solution is to deploy AACs in private domains. There is a great difference between grids and the Internet upon which our work is based: it is much easier to deploy customized solutions on a grid architecture than on the Internet because the number of cooperating sites is much lower. The idea of a global, unique grid on the scale to the planet is a nice idea but is usually impossible to deploy and manage: practically, several grids would coexist, especially if private companies are involved. Therefore, it is possible to deploy on a small scale, within a well-identified grid user community the application-aware idea. In this case, AACs could be deployed at the initiative of a computing center or an internal enterprise IT infrastructure or a campus for example. This solution is very possible now, with AACs based on dedicated hardware such as powerful PC-based active routers.

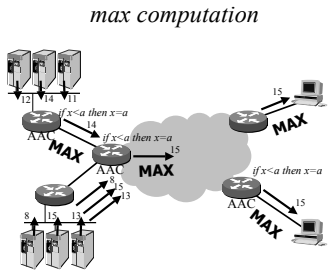

Figure 2. Example of a max operation with network support.

As shown in Figure 1, AACs are typically put at the edges of the core network. Since the core network is reliable and a very high-speed network (several Gb/s), adding complex process-

ing functions inside the core network will certainly slow down the packet forwarding functions. As described previously, AACs provide the ability for grids to dynamically adapt to new usages, and especially to use the network infrastructure as an efficient computing system. For instance, Figure 2 illustrates a network-supported max operation where AACs are used in the network infrastructure to assist for the computation of the max value by hierarchically computing/aggregating the values sent from end-hosts.

The range of utilization is much broader than the simple scenario illustrated in Figure 2. An application-aware grid could for example perform locality-dependent data encryption, heterogeneous video compression for remote displays, interest management and filtering, etc. Specific services can be deployed on demand upon the arrival of a data flow on an AACs.

2.3. Scenario of usage

To support most of grid applications, the A-Grid architecture must deal with the two main grid configurations :

- Meta cluster computing (Fig. 3) : a set of parallel machines or clusters are linked together with Internet to provide a very large parallel computing resource. Grid environments such as Globus, MOL[32], Polder[28] or Netsolve[4] are well designed to handle meta-cluster computing session to execute long-distance parallel applications.

  In this highly coupled configuration, an AAC is mapped on the network head of each cluster or parallel machine. This node manages all data streams coming or leaving a cluster. All AACs are linked with other AACs mapped at the backbone periphery. An AAC delivers data streams to each node of a cluster and can aggregate output streams to other clusters of the grid.

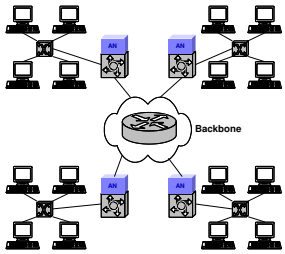

Figure 3. Meta-cluster computing.

- Global or Mega-computing (Fig. 4) : these environments usually rely on thousand of connected machines. Most of them are based on computer cycles stealing like Condor, Entropia[10], Nimrod-G[7] or XtremWeb[39].

  In this loosely coupled configuration, an AAC can be associated with each grid node or can manage a set of aggregated grid nodes. Hierarchies of AACs can be deployed at each network heterogeneity point. Each AAC manages all operations and data streams coming to/from grid nodes : filtering, interest management, results gathering, nodes synchronization and checkpointing, collective and gather operations, election, etc.

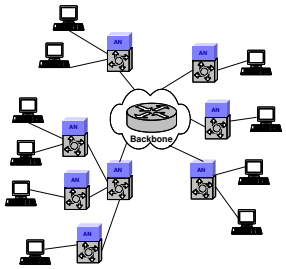

Figure 4. Global computing.

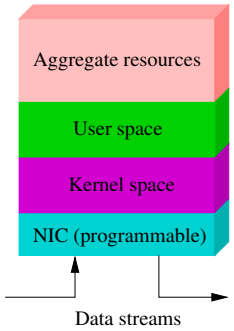

Figure 5. Execution Environment of an active node architecture.

For both configurations, AACs will manage the grid environment by deploying dedicated services adapted to the grid (as an infrastructure) requirements : management of nodes mobility, dynamic topology re-configuration, fault tolerance, etc.

3. Designing an high performance AAC to efficiently support an active grid infrastructure

When designing AACs capable of sustaining the high bit rate found in nowadays grid infrastructures, performance is the main concern. In this section we present a high performance active router architecture serving as an AAC to be deployed around high performance backbones.

We define an Execution Environment (EE) as an environment able to *load* and execute services, like network services. It must be also able to *direct* packets towards the required service thanks to appropriate headers filtering. By taking into account high performance challenges, we designed an active node where active services can be deployed at various levels depending on resources (processing capabilities, memory consumption, storage capacity) and intelligence (flexibility of the execution environment) they need. In order to provide an adapted EE for each type of services we designed the AAC architecture on 4 levels : Network Interface Card (NIC), kernel space, user space and distributed resources (Fig. 5).

This layered architecture proposes solutions to avoid the inefficient packet ascent and to allow the dynamic embedding of services optimally deployed on suitable levels :

- network programmable cards for ultra lightweight services (no-state, few cycles CPU requirements, scarcely updated and embedded) such as packets marking, dropping and filtering services;
- kernel space level for lightweight services (state with reduced memory usage, real-time) such as packets counting, QoS, management services, intelligent dropping and state-based services;
- user space level for middle-level services (complex with high level language, sequential, access to memory and disk) such as packets monitoring, reliable multicast, packets aggregating and data caching services;
- aggregate resources for high-level services (CPU and memory consuming, parallel and distributed) such as compression and multimedia transcoding on-the-fly services.

Depending on their software or hardware-assisted implementation, active services can be deployed on various combined levels.

3.1. The TAMANOIR framework

The aims of the TAMANOIR project is to design an high performance node validating the architecture described previously. The development of the high performance TAMANOIR EE has been done in several steps. First we implemented a EE running in user space, next we investigated the kernel space solution and finally the distributed computing approach.

3.1.1. High-level multi-threaded Execution Environment in user space

The TAMANOIR software suite [16] is a complete software environment dedicated to build AACs and to deploy services inside the grid infrastructure. TAMANOIR Active Nodes (TAN) are persistent AACs which are able to handle different applications and various data streams at the same time (multi-threaded approach). Both TCP and UDP are supported by TAN with the ANEP format [1] (Figure 6). New services are plugged dynamically in the TAN. The Active Node Manager (ANM) is dedicated to the deployment of active services and to update routing tables. The TAMANOIR EE running in user space is written in the JAVA^{TM} [3] language which provides a great flexibility and is shipped with the standard library. Services are also written in JAVA and deployed inside independent threads.

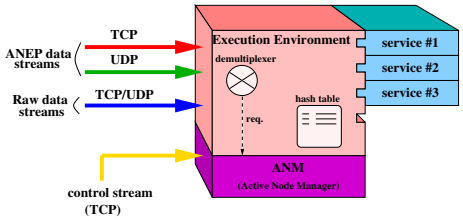

Figure 6. A TAMANOIR Active Node (TAN)

The dynamic injection of new services is performed out-of-band of data streams : services are deployed on demand when streams reach a TAN which does not hold the required service. As shown on Figure 7 two service deployment methods are available : (i) by using a *service repository* where TANs send all requests for downloading the required services and, (ii) by deploying the services from TAN to TAN. In order to avoid having a single point of failure, a service repository can be mirrored and replicated. Active data streams can of course go through legacy routers without any processing actions and overheads.

As mentioned in introduction, security in active networks can be a mandatory aspect for service deployment. In our Active Grid current deployment, we adopt an operator's perspective which consists in deploying some well identified services. This paper does not cover security and authentication aspects between users, service repositories and AACs.

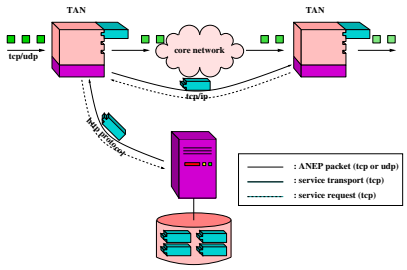

Figure 7. Dynamic service deployment.

3.1.2. Kernel space Execution Environment

Another part of our project is to investigate the idea of deploying lightweight services (packets counting, QoS, management services, state-based services) inside the kernel space of the operating system. Our main purpose here is to efficiently deport active functionalities from the high-level execution environment and virtual machine for the JAVA^{TM}platform (JAVA Virtual Machine or

[3] JAVA, JVM and Sun are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

JVM^{TM})[4] into the OS kernel.

Recent versions of the Linux kernel are well furnished with networking functionalities and protocols : QoS, firewall, routing and packet filtering. For packet filtering, kernels 2.4 propose IpTables with NetFilter [33]. NetFilter is a framework for packet modification, outside the normal Berkeley socket interface. With IPv4 communication protocol, NetFilter provides five *hooks* on the IP packet processing path. These hooks allow to develop and run modules written in C at the kernel-level. The function *nf_register_hook* is used to attach a personalized function to a specific hook. When a packet reaches the hook, it is automatically transmitted to this personalized function.

The various modules which are set up into the OS kernel can be modified dynamically by active services. A TAMANOIR active service, running inside the JVM^{TM}, configures the NetFilter module by sending control messages. These messages are captured by the NetFilter module and used to parameterize lightweight services (forward, packet marking, drop, etc.). This on-the-fly configuration allows to dynamically deport personalized functions inside the kernel.

3.1.3. Distributed service processing: cluster-based TAMANOIR

High-level and application-oriented active services require intensively computing resources. To support these services, a TAMANOIR Active Node embeds a dedicated cluster to efficiently deploy parallel services on streams.

The Linux Virtual Server (LVS)[22] software suite is dedicated to provide distributed servers (ftp, web, mail, etc.) offering best performances in terms of throughput and availability. We modified and adapted LVS functionalities for active networking and used them within the TAMANOIR EE. A TAMANOIR-LVS is a collection of TAN execution environments running on a cluster of machines and linked together with an high performance network (Myrinet or Giga-Ethernet). A dedicated machine is configured as a front-end and is used to route packets from the internet to back-ends machines. The TAMANOIR EE is replicated on each back-end machine.

[4]The terms "JAVA virtual machine" and "JVM" mean a virtual machine for the JAVA^{TM}plateform.

3.2. Experimental results

Our first experiments have been deployed on a Gigabit platform (Giga-Ethernet and Myrinet networks) with Dual-Pentium III, 1.4GHz. Each node runs GNU/Linux operating system (kernel 2.4.18).

3.2.1. Lightweight services in kernel space

We experiment the deport of active forwarding service inside the kernel for TCP active packets. We compare the latency of a forwarding functionality inside user space (Sun^{TM}JVM, IBM® [5] implementation of the JVM and GCJ [17] implementation) with the same basic functionality in kernel space (fig. 8).

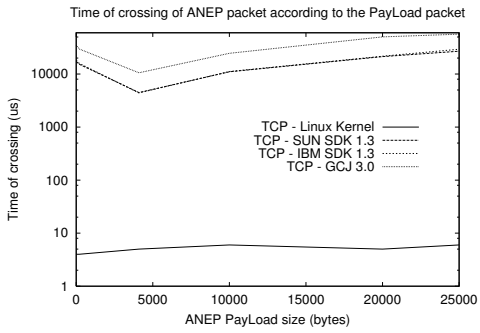

Figure 8. Latency time for ANEP packets using TCP protocol.

As we expected, when a packet is forwarded by the execution environment in the JVM level, raw performances are impacted. Packets crossing the TAN while passing by the Linux kernel spend only a few microseconds whereas they can spend several milliseconds (according to the payload size) while passing through the JAVA platform layer of the active node.

[5]IBM and the IBM Logo are registered trademarks of IBM Corp. in the United States and other countries, and are used under license by Sun Microsystems, Inc.

Results obtained in user space show large difference between JVM. The Sun JVM provides the best results on these tests. On the kernel level, the size of ANEP packet does not really affect performances as packets are simply redirected towards the exit hook by modifying the destination address with no packet copy.

3.2.2. Middle-level active services in user space

We explored the limits of a TAN running on a single machine and deploying middle-level services in user space. Our experiments consisted in sending active packets from a set of sending hosts to another set of receiving hosts on Gb/s networks. These active packets are processed and routed by active middle-level services (packet monitoring) deployed on intermediate TANs. Figure 9 shows the performances of a TAN crossed by various TCP streams. The TAN scales compared to the number of streams and provides best performances with large active packets.

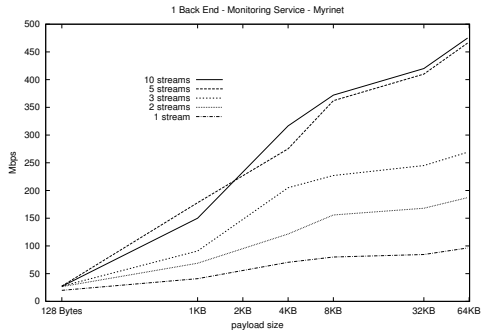

Figure 9. TAMANOIR Active Node performances on a middle-level monitoring service deployed in user space.

These first experiments show impressive results in terms of performance: a TAN running on a single machine and supporting middle-level service in user space can support up to 450 Mb/s of bandwidth depending on the packet size and the number of streams while lightweight services running in kernel space support up to 800 Mb/s.

3.2.3. Distributing services on a cluster-based TAMANOIR: fully supporting Gigabit networks

As a TAN running on a single machine is not enough to fully support a gigabit network, we explore the cluster-based active node solutions. We implemented a TAMANOIR node based on 3 back-end machines for the deployment of services and 1 front-end machine for streams distribution between back-end machines.

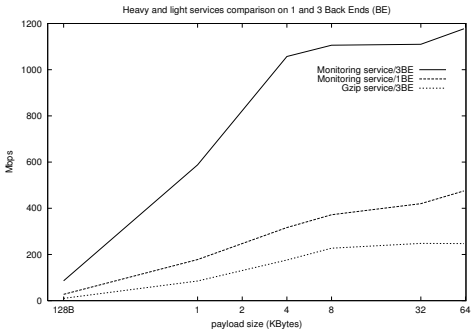

Figure 10. Cluster-based TAMANOIR experiments with middle-level and high-level active services.

We can see in Figure 10 that with this distributed architecture, TAMANOIR is able to support gigabit networks requirements for a middle-level active service (around 1.05 Gb/s for 4KB packets). In this experiment, active streams are equally processed between back-end machines. With larger packet size more throughput is achieved up to 1.4 Gb/s. Of course, obtaining these performances and fully supporting a gigabit network depends on the deployed service (high-level, middle-level or lightweight service) and its implementation quality.

In Figure 10, we also compare the impact of service level by deploying an high-level compression service (gzip on-the-fly). This service re-

quires high processing power and back-end nodes spend most of their time on performing gzip compression. A Gzip service on a 3 back-end TAN, supports around 200 Mb/s of bandwidth. With this kind of service, we should add a few more back-end machines to fully support a Gb/s bandwidth.

We have deployed TAMANOIR active nodes around VTHD[6] high performance backbone. First experimental results are presented in Figure 11. We compare our long distance results over VTHD with benchmarks made on a Myrinet gigabit, low latency, local experimental platform. We show that with 8KB packets and 25 streams we sustain a throughput of 919 Mb/s. It's a good result because network technology employed over VTHD is Gigabit Ethernet which is different and less performant than Myricom technology. Moreover, because it's a long distance experiment, lot of network devices have to be crossed and this increase latency.

By providing high performance, Tamanoir active nodes are able to support active services deployed by Grid applications on high performance backbones.

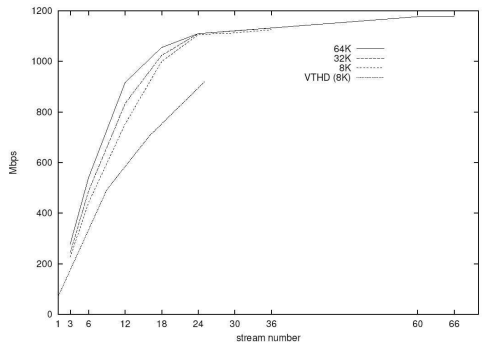

Figure 11. Cluster-based TAMANOIR experiments on a local network and around VTHD

These first experiments show that the TAMANOIR framework can support a grid environment without adding to much latency to all data streams. Next sections will focus on two services for enhanced communications on grids: QoS and multicast communications.

[6]See http://www.vthd.org

4. Enhanced communications on an active grid architecture

This part presents the opportunities the application-aware components and the active network technology to provide enhanced communication facilities on an grid architecture for the next generation of applications. We will present (i) dynamic network QoS management and control and (ii) reliable multicast to illustrate how an active grid architecture can contribute to provide a higher level of performances.

4.1. Grid QoS

The network Quality of Service is traditionally defined as a set of tools and standards that give network managers the ability to control the mix of network characteristics like bandwidth, delay, variance in delay (jitter) and packet loss. Controlling this mix allows to provide better and more predictable network services. The purpose of Active Grid QoS is to offer a set of facilities that gives the grid applications the capacity of *specifying and dynamically control the end to end network characteristics* like end to end transfer delay or end to end throughput. The QoS requirements in a grid environment is very large. Several analysis (e.g. [29]) have demonstrated that the QoS performances requirements spectrum of grid streams is broad comprising needs like the guaranteed delivery of a complete data file, the predictability of the TCP throughput, the bounded delivery time of a file transfer and a stability-level for data-delivery. As detailed in section 3.1, the network has to efficiently support both grid control flows and grid application flows. The quality of service the grid control flows will receive may affect the overall grid infrastructure performances and utilization level while the quality of service the application flows will experiment will impact each individual application performance. In both cases, each type of traffic has to be treated differently in terms

of end-to-end delay, end-to-end loss rate or end-to-end throughput. In general, applications deployment and large parameters transfers between nodes are more concerned about grid QoS services based on bandwidth requirements while grid control streams and data streams of pipelined coupled applications are more concerned about delay based QoS services. Grids generally rely on IP networks for wide area communications. But, the IP protocol, being a connectionless protocol, has no inherent traffic contract concepts, no admission control and provides a simple packet forwarding service with no delay and loss guarantees. Since more than ten years, the Internet community tries to find a way to introduce the capabilities required to support QoS in the Internet infrastructure and to develop mechanisms and algorithms that scale, while enabling a wide range of QoS guarantees. The IP network QoS proposed solutions are based on resource reservation (IntServ [6]) or on class-based services (DiffServ [5]). IntServ requires per flow signaling and state to be maintained at all routers on a flow's path. This is not feasible for deployment in wide-area backbone routers since there could be an unbounded number of flows. DiffServ has only a fixed number of service classes, regardless of the number of flows. Hence, while this only provides a statistical guarantee, it is much more practical for real-world deployment. Today, DiffServ solutions have been deployed in some Internet clouds. DiffServ services like Premium Service, Assured Service and Less than best effort service are the most deployed ones. By construction, DiffServ service properties are statistically guaranteed only within a single DS domain. As grids rely on a complex interconnection of DS domains and flat IP domains, end-to-end flow performances cannot be guaranteed or predicted. Then, for achieving end-to-end QoS objectives the remaining deficiencies of the network performances have to be masked by adaptation performed at host level. The grid flows have actually to integrate adaptive algorithms or delay-transparent algorithms to compensate the low quality of service or the lack of transfer delay bounds provided by the networks interconnections. The drawback of such an approach is that it imposes the application to be network-aware and complexifies the programming and the execution of its communication parts. The next section will examine how the integration of AACs and active network technology may enable grid users and grid applications to simply access and fully benefit from the QoS capabilities, like DiffServ, partially offered by the network.

4.1.1. Dynamic QoS control and Management with active services

The aim of the active approach, instanciated in the QoSinus (QoS invocation and utilization service) service is to introduce flexibility and dynamicity in the management, the control and the realization of end-to-end QoS in a grid context. The proposed model aims to cumulate the advantages of the QoS solutions offered at the network level and the end-to-end adaptation approach. The goal is to avoid the respective drawback of both approaches and to propose an interface to make both approaches to efficiently interoperate. The aims are to:

1. provide an extensible end-to-end differentiated services set to meet heterogeneous grid flows QoS objectives,
2. refine the network QoS granularity by processing grid flows aggregates rather than coarse DiffServ classes,
3. rely on the existing IP QoS services provided by the core inter-network for improving the individual packet performances,
4. realize a dynamic and appropriate adaptation along the path according to the real state of the link, the QoS mechanisms configured along the path and the experienced performances of the flows.

The main issues to be solved are firstly to allow grid components or grid applications to program and control their QoS and, secondly, to define generic QoS services that can be on demand deployed to realize the required QoS flexibility in a specific networking context. The first issue is addressed by an API that provides the user the ability to characterize the flow's needs in terms of qualitative or quantitative end-to-end delay,

end to end throughput, end-to-end loss rate or in terms of relative weight of these three main metrics. This first stage dissociates the end to end QoS objectives specification from its effective realization. The second issue is to enable the flows to activate dedicated processing components in the network in order to meet their QoS objectives when crossing heterogeneous DiffServ domains. We have identified four types of QoS active functions for flexible QoS programming and control: QoS programming service, QoS measurement service, QoS adaptive control service and QoS enforcement service. The QoS programming services are invoked for propagating and storing the flow QoS goals in the active nodes belonging to the path. The QoS monitoring entities are responsible of the QoS characterization of a class of a specific link. The QoS adaptive control components are responsible for controlling the correctness and for adapting the packet forwarding performances regarding the QoS goals of the flows. The QoS enforcement service intercepts and processes the flows when necessary. Other agents may be added for status report exchange. This extensible and open approach has been implemented in the QoSinus active software we have designed and developed within the context of the e-Toile project [30] (an experimental wide area grid testbed using the TAMANOIR environment). Qosinus will be extended and evaluated in a multi-domain grid testbed deployed across Europe. QoSinus acts an active QoS service that connects the Grid application QoS specification with an adaptive packet marking at DiffServ domains frontiers. The QoSinus model is composed of several main entities: a monitoring component that measures the performances of each class on each path to egress active nodes (as presented in table 1), a local controller that monitors the allocated bandwidth of each class and realizes the flow conditioning and a decision component that decides which class has to be attributed to each packet to meet the QoS flow requirements when crossing the next DiffServ domain. From the user point of view the QoSinus service is invoked in two phases. The first one is called the programming phase: the Open Service Level Specification is transmitted to all QoSinus modules on the path from sender to receiver. A QoS session code is returned back to the sender. Then the sender transmits data packets, identified by the attributed QoS session code. The QoSinus decision component dynamically chooses the appropriated DiffServ class to cross the DS domain, trying to optimize the class utilization. The e-Toile grid is based on the french Gigabit VTHD backbone which provides four DiffServ classes (EF, TCP AF, UDP AF and BE). The principle of the DiffServ configuration in VTHD is that classes are statically allocated and provisioned in the edge routers. The consequence is that each access point has to control and shape the traffic injected in each class. In the e-Toile context, where only one DiffServ domain is available, the decision component of the ingress TAN is responsible for the flexible mapping of end-to-end high-level QoS needs of grid flows to the actual QoS provided by the network cloud. When a DiffServ class is allocated by the decision component, the corresponding available bandwidth parameter has to be updated. In this case, the flow QoS enforcement component of QoSinus implements a token bucket for admission control. The advantage of the active approach here, is that it permits to easily deploy dedicated and heterogeneous enforcement components for each DiffServ domain. Table 1 shows the performances obtained from one end (ENS Lyon) to an other end (CEA in Paris) when using the EF and BE classes in two extreme experimental conditions: empty network and congested network. The TCP stack used is the standard one (TCP NewRENO) with no Iperf tuning. For UDP, Iperf gives a throughput of 100 Mb/s. Links are denoted with EC (ENS to CEA) and CE (CEA to ENS). The e-Toile experiment shows that the flexible and modular approach proposed by QoSinus is functional. To verify that the API is well adapt to Grid context, we conducted experiments with real grid applications that have QoS and flow isolation requirements. Next step is to conduct the same experiments with the same Grid applications across multiple DiffServ domains to examine how QoSinus helps to solve the inter-domain and partial deployment problems of DiffServ in the Internet. We expect to validate the per-domain adaptive

Metric	Link	DS	no load	congested
RTT	EC	BE	5.8 ms	32ms
RTT	EC	EF	5.8 ms	6.8ms
RTT	CE	BE	5.8 ms	72ms
RTT	CE	EF	5.8 ms	6.8ms
TCP	EC	BE	86 Mb/s	100 - 200 Kb/s
TCP	EC	EF	86 Mb/s	44 Mbs/s
TCP	CE	BE	49.6 Mb/s	100 - 200 Kb/s
TCP	CE	EF	49.6 Mb/s	49.6 Mb/s
UDP	EC	BE	100 Mb/s	73Mb/s+ 30% loss
UDP	EC	EF	100 Mb/s	100 Mb/s + no loss
UDP	CE	BE	105 Mb/s	64 Mb/s + 38% loss
UDP	CE	EF	105 Mb/s	105 Mb/s no loss

Table 1
The measured end-to-end performances on VTHD DiffServ classes from/to ENS (Lyon) to/from CEA (Paris)

packet marking features and the end to end QoS monitoring facilities. Scalability and performance aspects will also be studied in detail. The QoSinus architecture can be functionally compared with the GARA (Globus Architecture for Reservation and Allocation) architecture [13] that provides advance reservations and end-to-end management for quality of service on different types of resources (network, storage and computing). While in Gara, DiffServ class reservation is made, QoSINUS requires only a Service Level Specification. No resource reservation is made, only QoS objective are stored. After this phase, the data packets are sent transparently and the sender has never to care about QoS management or adaptation, except if it wants to modify the QoS objectives. The active network technology permits to easily replace and upgrade the modular QoS adaptation engine. As active components, the different functions can be easily modified to meet heterogeneous requirements of grid environment or upgraded to adapt the evolution of the network QoS architectures and mechanisms. For example, when on demand optical links will be made available in some networks, only the QoSinus mapping entity of the TAN located near the access equipment would be dynamically replaced and configured.

4.2. Reliable multicast communications on grids

Multicast [9] is the process of sending every single packet from the source to multiple destinations in the same logical multicast group. Since most of communications occurring on a grid imply many participants that can be geographically spread over the entire planet, these data transfers could be gracefully and efficiently handled by multicast protocols provided that these protocols are well-designed to suit the grid requirements. Motivations behind multicast are to handle one-to-many communications in a wide-area network with the lowest network and end-system overheads while achieving scalability (end-system or application-multicast solutions can succeed in providing low complexity but lack scalability). In contrast to best-effort multicast, that typically tolerates some data losses and is more suited for real-time audio or video for instance, reliable multicast requires that all packets are safely delivered to the destinations. Desirable features of reliable multicast include, in addition to reliability, low end-to-end delays, high throughput and scalability. These characteristics fit perfectly the need of grid computing and distributed computing communities. Embedding a multicast support in a grid infrastructure would not only optimize the network resources in term of bandwidth saving, but mainly increase both performances for applications, and interactivity for end-users, thus bringing the usage of grids to a higher level than it is at the moment.

Meeting the objectives of reliable multicast is not an easy task. In the past, there have been a number of propositions for reliable multicast protocols that rely on complex exchanges of feedback messages (ACK or NACK) [35,11,27,40]. All of the above schemes do not provide exact solutions to all the loss recovery problems and therefore are hardly suitable for distributed or interactive applications running on a grid. This is mainly due to the lack of topology information at the end hosts. In this section, we illustrate how an active grid infrastructure can help to provide low latency and high-bandwidth multicast communication for grid applications. In particular, we show how specific services can be composed and selected by the grid application to meet its networking requirements.

4.2.1. Specific services for reliable multicast

DyRAM [23] is a reliable multicast protocol framework that uses active network technology to allow an application to off-load specific functionalities traditionally performed by the end hosts in the commodity Internet. Several high-level mechanisms were thus identified to improve the performances (latency, scalability) of a multicast communication and were separated as application-aware services (AAS) to be off-loaded in the active grid infrastructure (in the AACs). Therefore the end host part of the protocol is kept very simple and the main AAS consist in:

1. the early detection of packet losses and the emission of the corresponding NACKs.

2. the suppression of duplicated NACKs (from end-hosts) in order to limit the NACK implosion problem.

3. the sub-cast of the repair packets only to the relevant set of receivers that have experienced a loss. This service limits the scope of the repairs to the affected subtree.

4. the dynamic replier election which consists in choosing a link/host as a replier one to perform local recoveries. Dynamic election provides robustness to host and link failures.

These services have been designed with the following motivations in mind: (*i*) to minimize AAC's load to make them supporting more sessions (mainly in unloading them from the cache of data) and (*ii*) to reduce the recovery latency for distributed and interactive grid applications. Avoiding cache in AACs for instance is performed by a dynamic replier election service that will be briefly described later on.

4.2.2. Simulation and experimental results on e-Toile

The performances of the previously described services have been investigated by simulation and validated by real experimentations on a grid testbed as part of the RNTL e-Toile french project. The simulation model has been implemented in the PARSEC language developed at UCLA [3]. The grid model considers an application multicasting data files to R receivers through a packet network composed of a fast core network and several slower edge access networks. AACs are located at the edge of the core network as described previously. The study shows that, depending on the need of the grid application, the deployment of specific services (amongst those implemented) at some strategic points can provide enhanced performances.

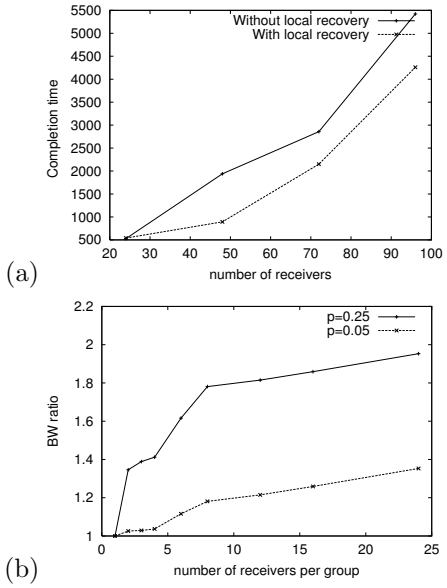

Figure 12. (a) Completion time, $p = 0.25$. (b) Consumed bandwidth ratio.

Figure 12a shows the completion time as a function of the number of receivers. p is the end-to-end probability of a packet loss perceived by a receiver. Local recovery has been enabled with edge AACs performing the replier election function. These results have been obtained from simulations of a multicast session with 48 receivers distributed among 12 local groups. The curves

show that local recoveries decreases the latency as the number of receivers increases. Although not shown in this paper, the load at the source is also decreasing when local recovery is enabled. This is especially true for high loss rates.

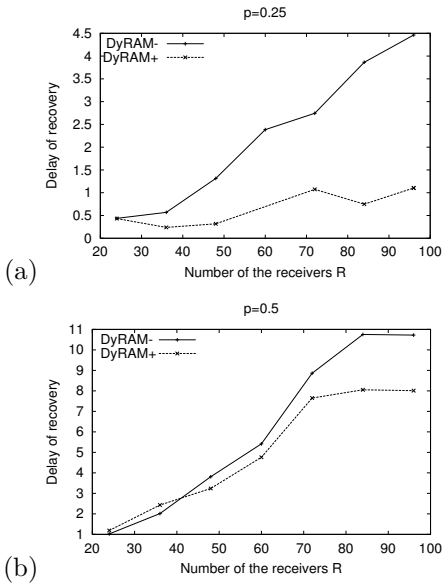

Figure 13. The normalized recovery delay with (a) $p = 0.25$ and (b) $p = 0.5$

Putting the recovery process in the receivers requires at least 2 receivers under a given AAC otherwise local recoveries can not be realized. Therefore the local group size (B) is an important parameter. In a real grid environment, B would represent the number of end hosts participating in a computation. In order to study the impact of B, simulations are performed with the 48 receivers distributed among groups of different sizes and Figure 12b shows how much bandwidth (in ratio) can be saved with the local recovery mechanism. As the number of receivers per group increases, the consumed bandwidth is reduced by larger ratios for large loss rates. In fact, when the group size is larger it is more likely that the members of the group can recover from each other. For instance, Figure 12b shows that with only 6 receivers per group and local recovery we can achieve a gain of 1.8. This result is particularly interesting for distributed applications with many collective and reduction operations.

The next simulations set shows how the early loss detection service could further decrease the delay of recovery. To do so, we simulate an application with low latency requirements that select two configuration noted DyRAM- and DyRAM+. DyRAM- has no loss detection services in AACs whereas DyRAM+ benefits from the loss detection service in the source AAC. Figure 13 plots the recovery delay (normalized to the RTT) for the 2 cases as a function of the number of receivers. In general, the loss detection service allows DyRAM+ to complete the transfer faster than DyRAM-. For $p = 0.25$ and 96 receivers for instance, DyRAM+ can be 4 times faster!

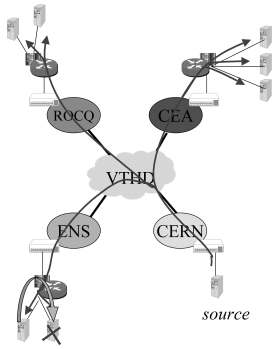

Figure 14. Testbed topology.

DyRAM has been tested and validated on a real grid infrastructure that connects 4 computing sites by a high-speed optical network (VTHD, 2.5 Gb/s). Figure 14 illustrates the test topology where a source located at CERN in Switzerland multicast a data file needed by a biology application to 7 receivers in 3 different sites. AACs located at the edge of the optical backbone provides the local recovery mechanism. Figure 15

is a snapshot of the test realized where the various steps of the local recovery are indicated. In this scenario, one computer in ENS lost packet 28 which is repaired by the other computer elected by the AAC located at the ENS campus.

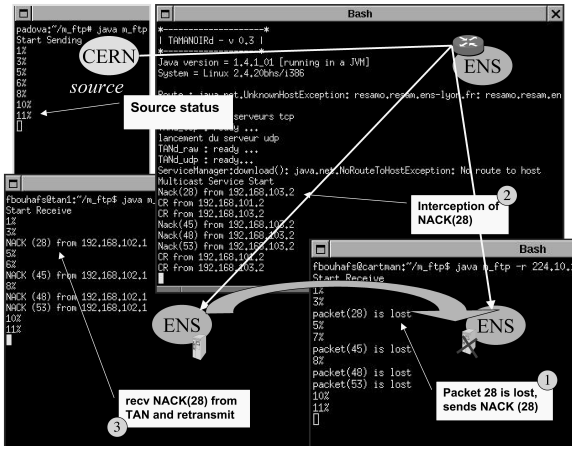

Figure 15. Augmented snapshot.

5. Conclusion and future works

This paper reported our experiments in proposing a new grid architecture that can take into account a large variety of existing grid computation models. The grid architecture we proposed relies on the definition of application-aware components distributed at the edge of the grid networking infrastructure where application-specific functionalities can be deported in the network in a transparent way for the grid designer and the grid user. These AACs are practically implemented using the TAMANOIR high-performance execution environment. The main motivations for this work is to show that application-aware components and services are viable concepts and bring more performance for the end-users. Enabling and generalizing the use of application-aware services on a grid infrastructure will provide a level of flexibility that have never been achieved in the Internet before and will certainly help for a massive usage of distributed computing.

To illustrate the usage of such a grid architecture we specifically explored how communications on a grid could be enhanced, first with QoS support and, second with a multicast support. The results show that the concept of AACs distributed at the edge of the grid networking infrastructure can provide a higher level of performances both in terms of latency and bandwidth. In the future, we would like to merge the TAMANOIR framework with a grid middleware environment (such as Globus) in order to provide more transparency for grid applications.

REFERENCES

1. S. Alexander et al. Active Network Encapsulation Protocol (ANEP). RFC Draft, Category: Experimental, `http://www.cis.upenn.edu/switchware/ANEP/`, July 1997.
2. D. Andersen, H. Balakrishnan, M. Kaashoek, R. Morris. Resilient Overlay Networks Proc. 18th ACM SOSP, Banff, Canada, October 2001.
3. R. Bagrodia. et al. Parsec: A parallel simulation environment for complex systems. *Computer Magazine, 31(10), October 1998, pp77-85.*
4. M. Beck et al. Logistical quality of service in netsolve. *Computer Communication*, 22(11):1034–1044, july 1999.
5. S. Blake et al. An architecture for differentiated services. RFC 2475, December 1998.
6. R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. RFC 1633, June 1994
7. R. Buyya, J. Giddy, and D. Abramson. An evaluation of economy-based reseource trading and scheduling on computational power grids for parameter sweep applications. *Active Middleware Services Workshop, 9th IEEE International Symposium on High Performance Distributed Computing, 2000.*
8. M. Calderón, M. Sedano, A. Azcorra, C. Alonso. Active Networks Support for Multicast Applications. *IEEE Networks, May/June 1998.*
9. S. E. Deering and D. R. Cheriton. Multicast

Routing in Datagram Internetworks and Extended LANs. In *ACM SIGCOMM'88* and *ACM Trans. on Comp. Sys., Vol. 8, No. 2.*
10. Entropia : high performance internet computing. `http://www.entropia.com`.
11. S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM ToN*, 5(6):784–803, 1997.
12. Foster and Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. IPTPS'03.
13. I. Foster, A. Roy, V. Sander. Int'l Workshop on Quality of Service, 2000.
14. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.*
15. I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl J. Supercomputing Applications, 11(2):115-128, 1997*
16. Jean-Patrick Gelas and Laurent Lefèvre. TAMANOIR: A high performance active network framework. *Active Middleware Services Workshop, 9th IEEE International Symposium on High Performance Distributed Computing, 2000.*
17. GCJ. The gnu compiler for the java programming language. `http://sourceware.cygnus.com/java/`.
18. A. Grimshaw, A. Ferrari, F. Knabe and M. Humphrey. Legion: An Operating System for Wide-area computing. *IEEE Computer, 32(5):29-37, May 1999*
19. S. K. Kasera et al. Scalable fair reliable multicast using active services. *IEEE Networks, Special Issue on Multicast*, 2000.
20. L. Lefèvre, C. Pham, P. Primet, B. Tourancheau, B. Gaidioz, J. P. Gelas, M. Maimour. Active Networking Support for The Grid. *Proceedings of the third International Working Conference on Active Networks (IWAN'01)*, 2001.
21. M. Litzkow and M. Livny. Experience With The Condor Distributed Batch System. In *IEEE Workshop on Experimental Distributed Systems, October 1990.*
22. J. Mack LVS HOWTO. `http://www.linuxvirtualserver.org/`.
23. M. Maimour and C. Pham. Dynamic Replier Active Reliable Multicast (DyRAM) *Proceedings of the 7th IEEE Symposium on Computers and Communications (ISCC 2002).*
24. M. Maimour and C. Pham. An analysis of a router-based loss detection service for active reliable multicast protocols. *Proceedings of the International Conference On Networks (ICON 2002), Singapore.*
25. K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. Internet Request For Comments RFC 2474, Internet Engineering Task Force, December 1998.
26. C. Papadopoulos, G. M. Parulkar, and G. Varghese. An error control scheme for large-scale multicast applications. In *IEEE INFOCOM'98*, pp1188–1996.
27. S. Paul and K. K. Sabnani. Reliable multicast transport protocol (rmtp). *IEEE Journal of Selected Areas in Communications, Special Issue on Network Support for Multipoint Communication*, 15(3):407–421, April 1997.
28. The polder metacomputing initiative. `http://www.science.uva.nl/projects/polder`.
29. P. Vicat-Blanc/Primet. High Performance Grid Networking in the DataGrid Project. Special Issue Future Generation Computer Systems, Elsevier, January 2003.
30. P. Vicat-Blanc/Primet. e-Toile project: development and implementation of a high-performance grid. Proceedings of the french National RNTL workshop, Toulouse, November 2002.
31. N. S. V. Rao NetLets: End-To-End QoS Mechanisms for Distributed Computing in Wide-Area Networks Using Two-Paths. Journal of High-Performance Computing Applications, Vol. 16(3), August 2002, pp285-292.
32. A. Reinefeld, R. Baraglia, T. Decker, J. Gehring, D. Laforenza, J. Simon, T. Romke, and F. Ramme. The mol project: An open extensible metacom-

puter. In *Heterogenous computing workshop HCW'97,IPPS'97*, Geneva, April 1997.
33. R. Russell. Linux Filter Hacking. HOWTO, July 2000.
34. R. State, O. Festor, E. Nataf. A Programmable Network Based Approach for Managing Dynamic Virtual Private Networks In *Proceedings of PDPTA 2000*, Las Vegas, June 26-29.
35. T. Strayer, B. Dempsey, and A. Weaver. XTP – THE XPRESS TRANSFER PROTOCOL. *Addison-Wesley Publishing Company*, 1992.
36. D. L. Tennehouse et al. A survey of active network research. *IEEE Comm. Mag.*, pp80–86, January 1997.
37. D.J. Wetherall, J.V. Guttag and D.L. Tennehouse. ANTS: a Toolkit for Building and Dynamically Deploying Network Protocols. In *IEEE OPENARCH'98, San Francisco, April 1998.*
38. L. Wei, H. Lehman, S. J. Garland, and D. L. Tennenhouse. Active reliable multicast. In *IEEE INFOCOM'98.*
39. Xtremweb : a global computing experimental platform. `http://www.xtermweb.net`.
40. R. Yavatkar, J. Griffoen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, pages 333–344, 1995.
41. S. Zabele et al. Improving Distributed Simulation Performance Using Active Networks. In *World Multi Conference, 2000.*

Laurent Lefèvre obtained his Ph.D in Computer Science in January 1997 at LIP Laboratory (Laboratoire Informatique du Parallelisme) in ENS-Lyon (Ecole Normale Superieure), France. From 1997 to 2001, he was assistant professor in computer science in Lyon 1 University and in the RESAM Laboratory (High Performance Networks and Multimedia Application Support Lab.). Since 2001, he is permanent researcher in computer science at INRIA (The French Institute for Research in Computer Science and Control). He is member of the RESO team (High performance networks, protocols and services) from LIP laboratory in Lyon, France. He has organized several conferences in high performance networking and computing and he is member of several programm committees. He has co-authored more than 30 papers published in refereed journals and conference proceedings. He is member of IEEE and takes part in several research projects. His research interests include : high performance active networks, active services, Grid and Cluster computing network support, Active Grid, Distributed Shared Memory systems and Data consistency. His web site is : http://www.ens-lyon.fr/ llefevre

Congduc Pham received his Ph.D. degree in computer science from the University of Paris 6, France in 1997. He is currently an Associate Professor at the University of Lyon 1, France and member of the INRIA RESO project. His research interests include parallel simulation of computer networks, cluster computing, active networking and multicast protocols. He is a member of the IEEE Society.

Pascale Primet graduated in Computer Science in 1984, obtained a Ph.D in Computer Science in 1988 with a thesis on Real-Time systems and an HDR (Habilitation à Diriger les Recherches) in 2002. Since 1989 she has been working at the Ecole Centrale de Lyon, teaching and doing research in Computer Science, Operating Systems, Internet protocols and Computer Networks. Member of the LIP laboratory and leader of the INRIA team project RESO since 2002, her research interests include High Performance Grid and Cluster Networking, Internet (TCP/IP), End to end Quality of Service, Transport protocols, Active Networks, Large Scale Distributed Systems, Collaborative Work. She has been the manager of the Network Workpackage of the European DataGRID project. She is also the scientifical coordinator of the French RNTL E-Toile grid Platform project funded by the French Research Ministry.

Bernard Tourancheau is currently principal investigator at Sun Labs Europe. His research interests include networking, communication models and protocols, clusters and parallel computing, parallel and distributed systems. From 1996 to 2000, he was a Professor at the University of

Lyon after two years as a CRPC research associate at the University of Tennessee, Knoxville and a decade as a research associate at CNRS in ENS-Lyon. He received his Ph.D. from INP-Grenoble in 1989 and his HDR from University of Lyon in 1994. He is a member of the IEEE Society.