Computer Standards & Interfaces xxx (2008) xxx-xxx

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Jxta-Overlay: An interface for efficient peer selection in P2P JXTA-based systems

Fatos Xhafa [a,*], Leonard Barolli [b], Thanasis Daradoumis [c], Raúl Fernández [c], Santi Caballé [c]

[a] Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord-Ed. Omega, C/Jordi Girona Salgado 1-3, 08034 Barcelona, Spain
[b] Departament of Information and Communication Engineering, Fukuoka Institute of Technology (FIT), 3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka 811-0295, Japan
[c] Department of Information Sciences, UOC, Av. Tibidabo, 39-43, 08035, Barcelona, Spain

ARTICLE INFO

Available online xxxx

Keywords:
P2P computing
Peer selection models
Economic models
JXTA technology
User interface

ABSTRACT

In this paper we address the problem of the efficient peer selection in P2P distributed platforms. To this end, we have developed a P2P distributed platform using Sun's JXTA technology, which is endowed with resource brokerage strategies to efficiently select peers using four selection models: (a) economic scheduling model; (b) priced-based model; (c) peer-priority selection model; and, (d) random selection model. Next, we have deployed the P2P platform in a real network using nodes of the PlanetLab and have experimentally evaluated the performance of the peer selection models. The P2P platform offers a user-friendly interface for efficient peer selection and configuration of the P2P platform.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction and motivation

P2P systems are evolving as new a distributed computing paradigm for the development of large-scale distributed applications by exploiting the large computing capacity offered by the nodes of the system altogether. The improvement of P2P protocols is enabling the development of P2P applications other than the well-known file-sharing applications. However, there is still few work to bring P2P system to real word P2P applications, mainly due to the lack of robust platforms that would allow the deployment of large P2P systems, in particular, for efficiently discovering and selecting peers. Some advances are being done in this direction; for instance, the JXTA platform [4,16,17] is making possible the development of P2P real-world applications. Moreover, projects such as seti@home [20] are showing the feasibility of using P2P platforms for real life applications.

This work is motivated by the need to design and implement several models for peer selection in P2P applications and offer them through a user-friendly interface. The aim is to implement and evaluate these models independently of P2P application domains in a way that they could serve to the development of high performance P2P application in general, that is, to facilitate the use of P2P infrastructures as distributed computing environments. The need for efficient peer selection arises in many P2P applications such as for job allocation, fast file transfer, etc. In general, no one model would be able to match the requirements of different scenarios/applications; rather, several models must be studied in order to identify which of them works best under which P2P infrastructure and/or application characteristics. The peer selection models considered in this work range from a simple random model to more advanced economic-based models. These models are as follows. (a) *Economic scheduling model* [7]: in this model the idea is to find/provision as many as possible available idle peers to which the new incoming jobs can be allocated. Crucial to this model is the ready time of peers in order to plan in advance the allocation of jobs to P2P nodes. (b) *Priced-based model* (e.g. [22]): in this model peers are associated a cost, which is computed using different criteria that range from peer's state to P2P infrastructure parameters. (c) *Peer-priority selection model*: in this model it is the user who selects the peer, among different candidate peers, based on previous traces/experiences of job allocations submitted by the user and, (d) *random selection model*: this is the simplest model in which a peer is selected uniformly at random among several peer candidates. It should be noted that the use of economic models in P2P systems is a hot research topic nowadays [5,9,13,18,21].

Our approach is exemplified using the Sun's JXTA open protocols and has been validated in practice through a simple distributed application scenario. We have joined the PlanetLab platform [14] –a planetary-scale distributed infrastructure– and used a slice of nodes to deploy a P2P network and have experimentally evaluated the performance of the proposed peer selection models. A distributed application for processing large size log files of a virtual campus, which requires both efficient file transmission and processing was chosen, for the experimental evaluation.

* Corresponding author. Tel.: +34 93 413 7880; fax: +34 93 413 7833.
E-mail addresses: fatos@lsi.upc.edu (F. Xhafa), barolli@fit.ac.jp (L. Barolli), adaradoumis@uoc.edu (T. Daradoumis), adaradoumis@uoc.edu (R. Fernández), adaradoumis@uoc.edu (S. Caballé).

0920-5489/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2008.03.015

ARTICLE IN PRESS

The rest of the paper is organized as follows. We briefly describe some related work in Section 2. The architecture of the P2P platform, called Jxta-Overlay, is presented in Section 3. In Section 4 we give the peer selection models considered in this work; the user interface is briefly presented in Section 5. The evaluation of the proposed models is given in Section 6. Finally, we conclude in Section 7 with some remarks and indicate directions for future work.

2. Related work

P2P systems are novel in technological, design and implementation issues. Recently, a considerable research effort is being done on several important issues related P2P systems. Much of this effort has been addressed on overlay networks [1–3] and quite a few address the design and implementation of libraries to support the development of P2P distributed applications. Also, the issue of discovery, resource location and allocation is addressed in several recent works [11,12]. Crowcroft et al. [6] addressed issues for P2P systems by putting special emphasis on: (a) deploying internet services by overlaying; (b) the need for scalability of P2P applications that would require keeping knowledge of a small fraction of global state in each peer; and, (c) the need for load balancing, which should be separated from the P2P applications. Regarding the efficient allocation of tasks to computational resources, most of the ideas from the Grid computing domain are also applicable to P2P domain, although some differences related to existing policies on resources should be taken into account. Given that P2P networks are usually large or very large as they are based on contributions of individuals, the peer selection model should be able to find/provision as many idle peers as possible while allocating tasks to P2P nodes. On the other hand, because P2P resources belong to different individuals and/or institutions around the world, the peer selection models based on economic-like models are quite desirable for P2P systems since they allow to easily incorporate incentive mechanisms, which are important for the deployment of P2P systems. One such interesting selection model is the one proposed by Ernemann et al. [7] for economic scheduling in Grid computing. Other related approaches are by Yu et al. [22] and Ping et al. [18]; in this later work JXTA technology is used.

3. The architecture of our P2P platform

In this section we present the architecture of the P2P distributed platform,[1] called Jxta-Overlay, we have developed using JXTA technology. The main building blocks of the platform are: (a) the *Broker* module; (b) the *Primitives* module; and, (c) the *Client* module. Altogether these three modules form a new overlay on top of JXTA (see Fig. 1).

Importantly, the new overlay is designed and implemented to be totally independent of any possible P2P applications, which will be built on top of the overlay. Clearly, one of the characteristics of the primitives module (see below) is their independence from the applications that will be using them. We give next a basic description of the three modules of the overlay.

Primitives: The objective of the overlay is to provide a set of basic functionalities, that we call primitives since they will be used by P2P application, as regards the discovery and allocations of resources. This set of primitives is intended to be as complete as possible as regards the functionalities for the discovery and allocations of resources. Roughly speaking, the set of primitives includes functionalities that allow: peer discovery, peer's resources discovery, peer selection, resource allocation, file/data sharing, discovery and transmission, instant communication, peer group functionalities. The primitives module is designed to be generic in a way that any application built on

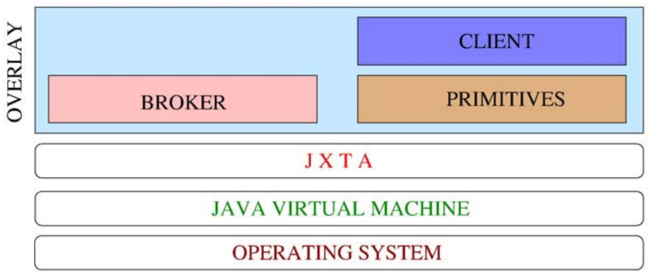


Fig. 1. The architecture of the P2P overlay.

top of the overlay can use it as a “black box”. To this end, we observed that the overlay should include, apart from primitive functionalities, two other modules: a *broker layer*[2] and a *client layer*.

Broker layer: This layer is in charge of achieving the resource allocation functionalities, resource monitoring, and management of executable tasks defined in the set of primitives. Note that broker peers do not interact with final user applications therefore they represent just one layer.

Client layer: This layer is in charge of receiving and managing all events produced in any application built on top of the overlay due to calls to the primitives.

By using the above architecture, we achieved the set of primitives to be completely independent of any application as the client layer will allow (final) user applications to communicate with the overlay. Moreover, the primitives allow to keep the intrinsic decentralized nature of Grid/P2P systems. The idea of the architecture using brokers has been initially explored in [19]. The set of primitives that allow to accomplish the aforementioned functionalities is organized in interfaces according to an affinity criterion. Thus, we have the *interfaces authentication*, *resource discovery and information*, *management of executable tasks*, *file sharing*, *discovery and transmission*, *resource statistics*, among others. An important place in the primitives is given to functionalities related to the management of executable tasks. These functionalities are intended to give service to users/applications on top of the overlay that submit executable tasks and receive results in turn. It should also be mentioned that file-sharing and transmission functionalities extend existing JXTA functionalities of sharing in P2P systems since an efficient file transmission is necessary for submitting tasks to resources. Resource statistics is another important interface in the overlay, and it is particularly useful for the selection of peers (statistics about peers, peer groups, brokers and clients.)

3.1. Peers, brokers and discovery in the Jxta-Overlay

Now we show how it implemented the set of the primitives. We take advantage that JXTA allows different types of peers and we classified them into two groups: *client peers* and *broker peers*. The former are *complete edge peers* while the latter act as *rendezvous* and *relays*.

3.2. Broker peers

Brokers are the *governors* of the network: they are connected to the to P2P platform and are in charge of receiving and allocating the requests sent by clients of the peer group. Whenever a broker receives a request, it selects, according to one or more peer selection models, the best peer candidate for processing that request and makes the allocation. It should be noted that the definition of the broker peers allows to keep the control on the resource allocation. Thus, any peer group has (at least) a broker to which client peers get connected and send their resource allocation petitions. This is done by redefining the

[1] For more details and updated information on this platform, please refer to http://jxta-overlay.dev.java.net.

[2] Throughout the paper we use indistinctly the terms *Broker module*, *Broker layer* or simply *Broker*. Similarly for the *Client*.

ARTICLE IN PRESS

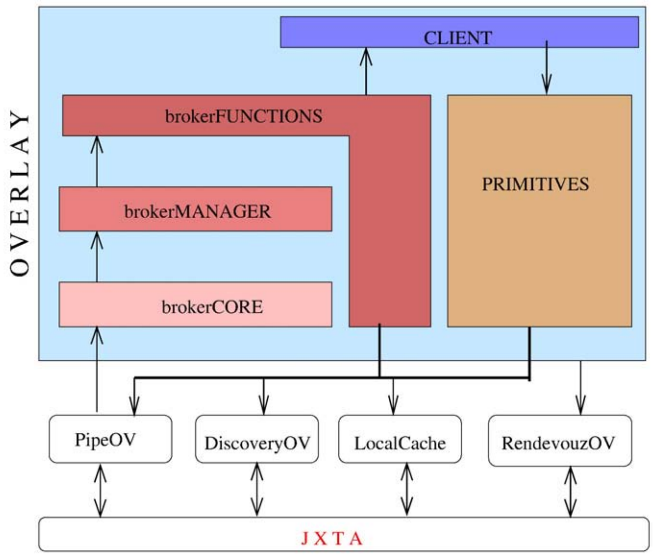


Fig. 2. Broker's design and the overlay architecture.

JXTA rendezvous, the Pipe, Discovery and Rendezvous of JXTA (denoted RendezvousOV –rendezvous overlay–, PipeOV, etc.). Among broker's functionalities we distinguish: (a) event management; (b) controlling the resources connected to the broker; (c) maintaining the organization of resources in groups; (d) finding the best resource for file sharing; (e) finding the best resource (according to scheduling policy/economic models) for task execution; and, (f) maintaining updated statistic information (as regards task executions, file transfers, etc.). Further, we also note that the design of the broker is organized in several layers/modules: *brokerCore*, *brokerManager* and *brokerFunctions*. We give in Fig. 2 the design of the broker and its relation with the overlay.

Observe that we have redefined the Pipe, Discovery and Rendezvous defined in JXTA (here abbreviated PipeOV –Pipe Overlay–, DiscoveryOV and RendezvousOV, respectively). This is done mainly to ensure reliability of the overlay. Indeed, JXTA Pipe Service doesn't check whether a message has been successfully delivered to its destination peer and in case of failure, JXTA doesn't attempt a new delivery of the message. Also, JXTA maintains the peer group information by the notification of presence that each node publishes in the cache of other peer nodes. But this procedure is not automatic in the JXTA library.

Further, note the design of the broker in several layers/modules: *brokerCore*, *brokerManager* and *brokerFunctions*. We briefly explain them next.

3.2.1. Broker

The mission of a broker node is to manage and control all requests for executions of tasks to resources. Also, broker peers manage the events produced by such requests and propagate them to the superior level (see Fig. 2). Among broker's functionalities we distinguish:

- Event management (according to their relevance).
- Controlling the resources connected to the broker resource.
- Maintaining the organization of resources in groups.
- Find the best resource for the file sharing.
- Find the best resource (according to scheduling policy and economic models) for executing submitted task.
- Maintaining updated statistic information (as regards task executions, file transfers, etc.).

BrokerCore module. The brokerCore is in charge of *listening* the events produced by the PipeOV, such as received and sent messages, message failure, update statistic information etc., and propagating it to its upper layer (brokerManager). Thus, an instance of brokerCore is permanently waiting for new events from PipeOV.

BrokerManager module. This module is in charge of managing the resource allocation. An instance of the module is permanently running in the broker resource. It keeps queues of pending tasks to be submitted to resources. In order to achieve its responsibilities, the broker calls functionalities of its superior layer, the *brokerFunctions*.

BrokerFunctions module. This module contains all the functionalities needed to achieve the broker responsibilities regarding the allocation of resources. To this end, it can use different scheduling policies and economic models to choose the best peer candidate for task allocation.

3.3. Client peers

Client peers instantiate the Client module, which serves as a communication layer between the primitives and the final user application. A client peer is in charge of receiving and managing all events produced in any application (built on top of the overlay) due to calls to the primitives. It is organized in a similar way as the broker: *clientCore*, *clientManager* and *clientFunctions* (see Fig. 3).

4. Peer selection models

As part of the set of primitives we have implemented four models for peer selection. These primitives are then used as resource brokerage strategies by the broker peers. The peer selection models considered in this work range from a simple random model to more advanced economic-based models. These models are: *Economic scheduling model*; (b) *Priced-based model*; (c) *Peer-priority selection model*; and, (d) *Random selection model*.

4.1. Economic scheduling model

In this model [7] the idea is to find/provision as many as possible available idle peers to which the new incoming jobs can be allocated. Crucial to this model is the ready time of peers in order to plan in advance the allocation of jobs to P2P nodes. Thus, many parts of the application are processed in parallel in different peers and moreover peers can communicate among them during task realization. In this model is crucial the ready time (expected starting time to compute) of a given peer for a given task. In the case of task execution this

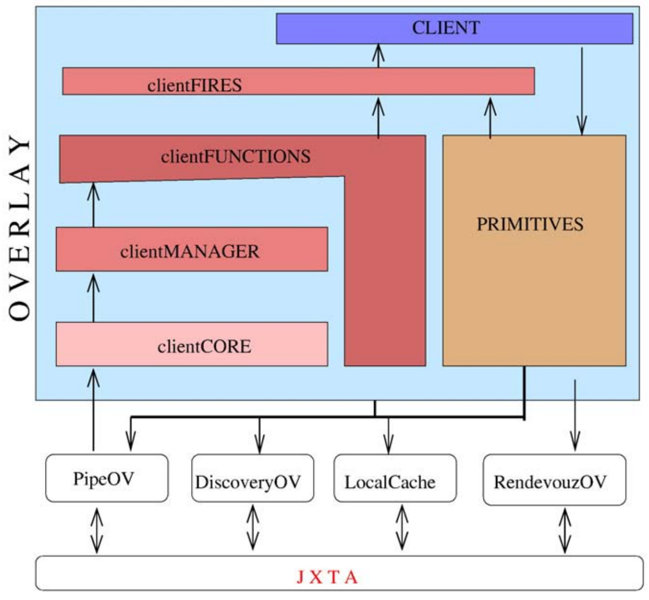


Fig. 3. Client module design and the overlay architecture.

ARTICLE IN PRESS

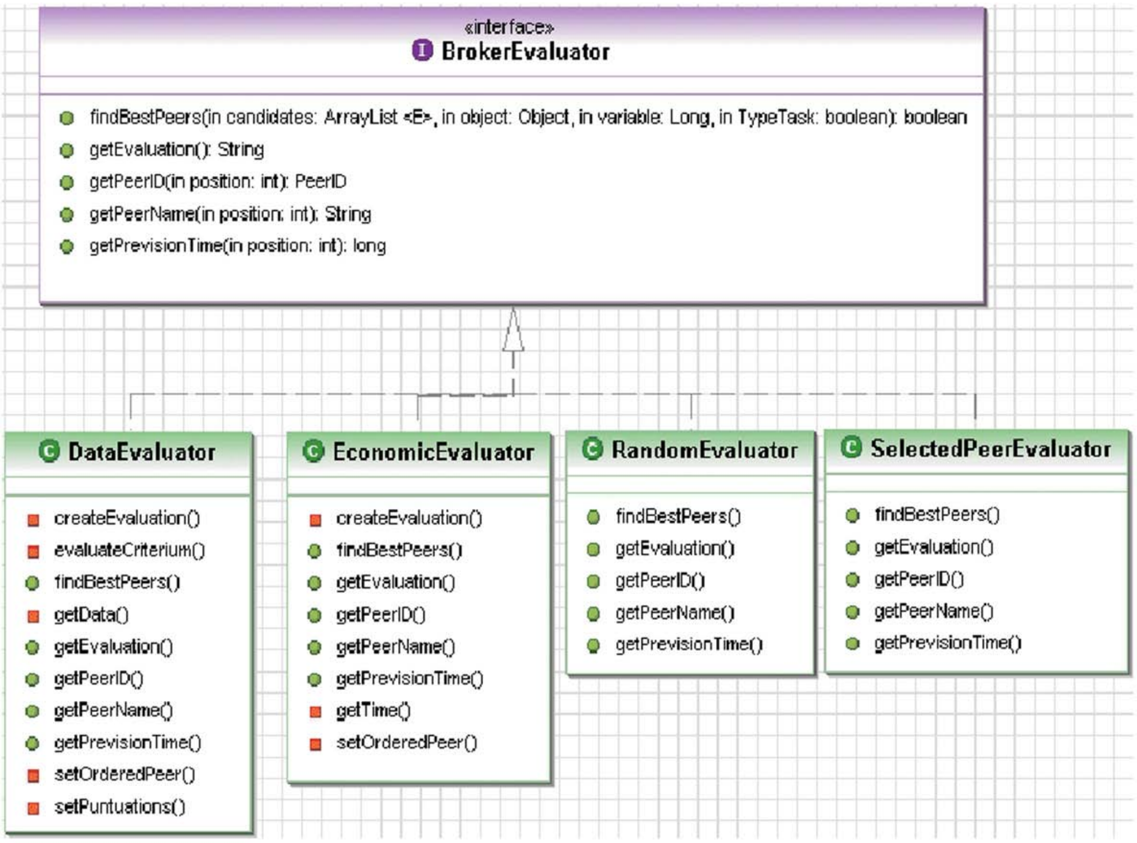


Fig. 4. Diagram of peer selection models.

information is either extracted from historical data or is specified by the user.[3] On the other hand, the peer advertisements are very important to know the state information of peers. However, this could be problematic for tasks needing a short or very short execution time since advertisement are periodically updated. In this case, and estimated time is computed by the broker based on historical data kept for the peers. In case several peers are available candidates for executing the task, some additional criteria such as CPU speed are used.

4.2. Priced-based model

In this model[4] peers are associated a cost, which is computed using different criteria that range from peer's state to P2P infrastructure parameters. The set of criteria used to identify the best peer(s) are classified into: (a) *global criteria* (percentage of successfully sent messages in the current session, percentage of successfully sent messages in all sessions (total), percentage of successfully sent messages during the last *k*-hours; number of messages in the outbox queue now, average number of messages in the outbox queue, number of messages in the inbox queue now, average number of messages in the inbox queue, average number of attempts in outbox in the current session, average number of attempts in outbox in all sessions (total), etc.; Amount of sent bytes in the current session, similarly for all sessions (total); Peer's bandwidth IN, Peer's bandwidth OUT, etc. (b) *specific task execution criteria*[5] (percentage of successfully executed tasks in the current session, percentage of successfully executed tasks in all sessions (total), percentage of tasks accepted by the peer for execution in the current session, percentage of tasks accepted by the peer for execution in all sessions (total), percentage of cancelled tasks in the current session, percentage of cancelled tasks in all sessions (total), etc. (c) *Specific file request criteria*[6] (percentage of sent files in this session, percentage of sent files in all sessions (total), percentage of cancelled file transfers in the current session; Average file transfer ratio, etc.

Each of the above criteria is given a certain weight (either user defined or pre-specified) meaning that some criteria are more important than others. A broker peer, upon receiving a request (task execution or file transfer) from a peer, evaluates the above criteria, applies the weights and thus assigns a *price* (a score) to each candidate peer. The best score peer is then chosen for executing the task. The user can specify two ways of computing the peer's score, namely *fixed point* (w.r.t. absolute position in the peer list) and *variable point* (w.r.t relative position in the peer list). Moreover, regarding the criteria's weights, the following specific ways have also been implemented: *all disabled* (no weights are considered, the peer is randomly chosen); *same priority* (the weights are the same, i.e., all criteria are equally important); *quickest peer* (only the criteria related to task execution performance and file transmission are considered independently of the peer reliability); *reliable peer* (only criteria related to peer reliability w.r.t. task execution/file transmission are considered, independently of peer's performance); *balancing peer* (only the criteria related to load balancing are considered). It should be noted that this model has a high computational cost.

[3] The reader is also referred to [10] (The Cornell Theory Center) and the Parallel Workload Archive [15].

[4] Also referred to as Data model in this work.

[5] A total of 47 criteria have been implemented in this model for peer selection for task execution purposes.

[6] A total of 39 criteria have been implemented in this model for peer selection for file transmission purposes.

ARTICLE IN PRESS

4.3. Peer-priority selection model

In this model it is the user who selects the peer, among different candidate peers based on previous traces/experiences of request (task execution or file transmission) submitted by the user. This model is useful when the user knows the performance of some peers in advance, for instance, from previous submissions of the tasks. In this case, the broker has to just assure that the selected peer is available for executing the task and therefore this model has a very low computational cost as opposed to the computational cost of the previous models.

4.4. Random selection model

This is the simplest model in which a peer is selected uniformly at random among several peer candidates. Although simple, this model could be useful when peer candidates are almost homogeneous.

We show in Fig. 4 the UML diagram of the considered models, which are instantiated by the broker module.

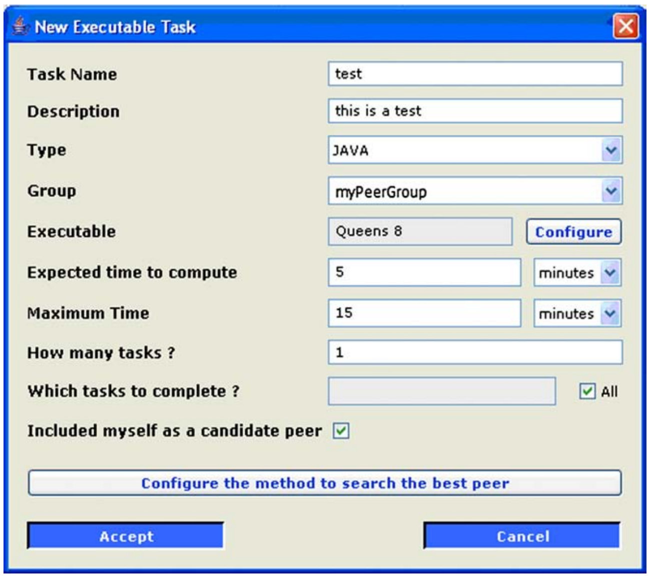
5. The interface of the Jxta-overlay

An important place in the development of the Jxta-Overlay is given to the user interface aiming the ease of use and configuration of the platform for distributed applications. We present next a sequence of snapshots of the overlay related to the execution of tasks of a distributed application in the peer nodes of the platform.

Through the interface the user can submit the execution of a distributed application, which could be executed as a whole in a peer node or could be made up of many parts to be submitted to different peer nodes in the platform. We show in Fig. 5, the snapshot of task submission configuration.

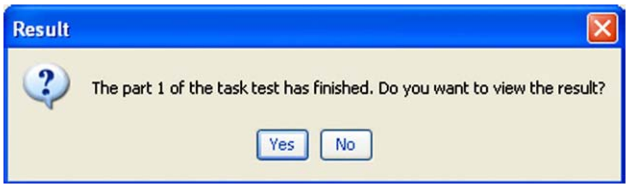
As can be seen from Fig. 5, the parameters to provide are:

- Task name.
- Description.
- Type: Indicates whether incompatibilities between tasks and peer characteristics (such as operating system) are to be checked.
- Group: The peer group to start searching the peers.
- Executable: The binary file (selected through Examine).
- Expected Time to Compute: Indicates the expected time to compute for the task.
- Maximum time: Maximum time the user is willing to wait for the completion of his task.



Task Name	test
Description	this is a test
Type	JAVA
Group	myPeerGroup
Executable	Queens 8
Expected time to compute	5 minutes
Maximum Time	15 minutes
How many tasks ?	1
Which tasks to complete ?	☑ All
Included myself as a candidate peer	☑

Fig. 5. Snapshot of task configuration to be submitted to the P2P platform.



The part 1 of the task test has finished. Do you want to view the result?

Fig. 6. Keeping track on executable tasks submitted to the P2P platform.

- How many tasks?: Indicates into how many parts is split the application (it is equal to 1 for applications running as a whole in a peer node).
- Which tasks to complete?: Allows to select the parts of the application (and their order) to complete or indicate to complete them all.
- Include myself as a candidate peer: Includes the peer that does the submission as a possible candidate to complete the submitted tasks.
- Configure the method to search the best peer: Allows to choose the peer selection method. Once a method is selected, the user is prompted to configure its parameters. For instance, in case of Data model, the user can indicate the weights of the criteria through a user-friendly interface.

When the user has completed the configuration steps, the submission is sent to a broker of the P2P platform. Then, the broker chooses the best peer(s) according to the indicated peer selection model and informs the user about the result.

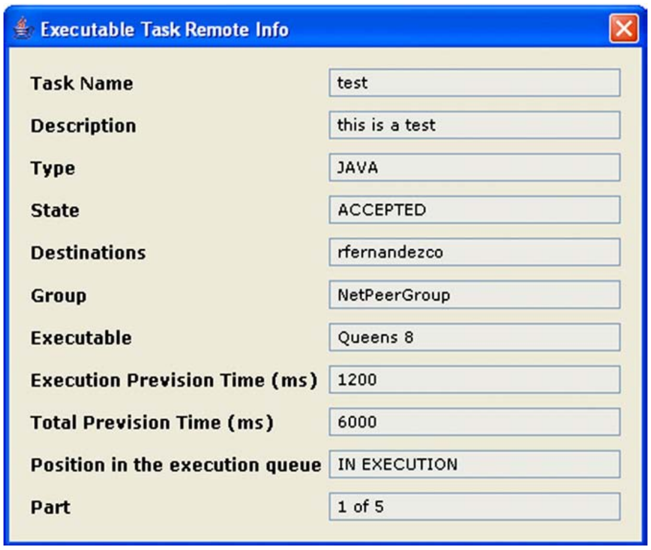
The user can make several submissions to the P2P platform. The interface allows the user to keep track of the state of its submissions, either to see the result of a completed task (see Fig. 6) or the state of the task on the remote peer (see Fig. 7).

6. Experimental evaluation

In this section we present the experimental study, starting with the deployment of the Jxta-Overlay P2P platform in a real network and the experimental setup for measuring the performance of the peer selection models.

6.1. Deployment of the P2P network

In order to evaluate the performance of the presented peer selection models, first we deployed the P2P network using nodes of the PlanetLab platform. PlanetLab [14] is an open platform for developing, deploying



Task Name	test
Description	this is a test
Type	JAVA
State	ACCEPTED
Destinations	rfernandezco
Group	NetPeerGroup
Executable	Queens 8
Execution Prevision Time (ms)	1200
Total Prevision Time (ms)	6000
Position in the execution queue	IN EXECUTION
Part	1 of 5

Fig. 7. State information on executable tasks submitted to the P2P platform.

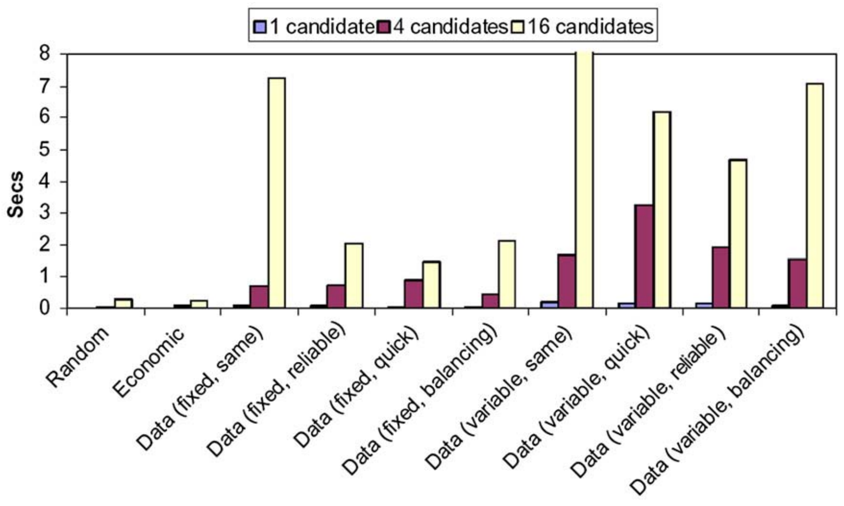


Fig. 8. Broker's processing time for discovering the best peer.

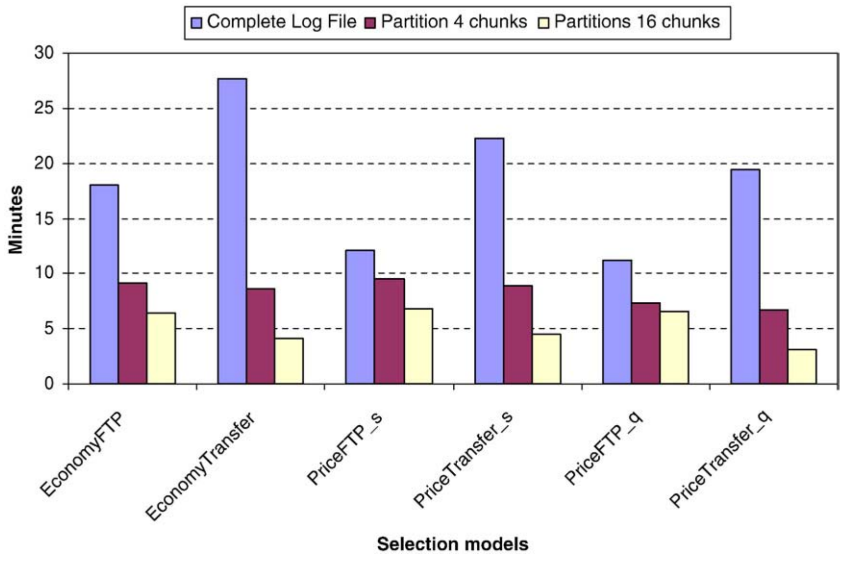


Fig. 9. Total processing time of log files in the P2P platform.

and accessing planetary-scale services. It is, at the time of this writing, composed of 838 nodes at 412 sites. Each PlanetLab node is an IA32 machine that must comply with minimum hardware requirements (i.e. 1 GHz PIII+1 GB RAM) running the same base software, basically a modified Linux operating system offering services to create virtual isolated partitions in the node, called slivers, which look to users as the real machine. PlanetLab allows every user to dynamically create up to one sliver in every node, the set of slivers assigned to a user form what is called a slice. It is said that a PlanetLab node can run up to 100 concurrent slivers. The sample of PlanetLab's machines forming our slice is about 25 nodes. Moreover we used the cluster nozomi.lsi.upc.edu (a main control node+five computing nodes). The main node was used as one the brokers of the P2P network.

6.2. The distributed application scenario

Next, we have chosen a simple but representative application to run on the resulting P2P platform. This application consists in processing large log files kept by the Virtual Campus at the Open University of Catalonia,[7] which offers distance education through the Internet in different languages. As of this writing, about 40,000 students, lectures and tutors from everywhere participate in some of the 23 official degrees and other PhD and post-graduate programs resulting in more than 600 official courses.

All users' requests are chiefly processed by a collection of Apache web servers. Each web server stores in a log file all users' requests received in this specific server and the information generated from processing the requests. Once a day, all web servers in a daily rotation merge their logs producing a single very large log file containing the whole user interaction with the campus performed in the last 24 h. A typical daily log file size may be up to 10 GB. Log file entries are structured following a type of format known as Common Log Format [8]. Unfortunately the log file is not human readable making thus indispensable its processing to extract relevant information that would serve as basis for later statistical processing. The problem of processing log files of the virtual campus represents several interesting characteristics. Log files are of large size making thus relevant a parallel processing using the P2P network. Further, due to their structure (Common Log Format) the log file can be very easily parallelized using the Master-Worker paradigm since the file can be split by a master node into many independent parts and processed in parallel by other peer nodes (slaves). Finally, the processing requires efficient file transmission.

[7] http://www.uoc.edu.

ARTICLE IN PRESS

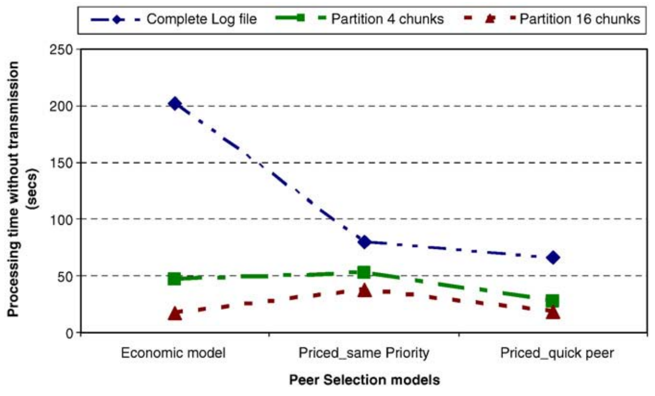


Fig. 10. Processing time of log files in the P2P platform (without transmission time).

6.3. Computational results and evaluation

For the experimental study we used daily log files and well-stratified short samples of about 100 Mb consisting of representative daily periods with different activity degrees (e.g. from 7 p.m. to 1 a.m. as the most active lecturing period). The computational results presented here are obtained by running the same experiment five times and the results are averaged.

We show in Fig. 8 the time needed by the broker to find the best candidate peer for each model when the log file was split into 1, 4 and 16 parts. As can be seen from this figure, the price-based model[8] is the most computationally expensive among the proposed models.

Two different modes for sending files to peers for processing were used: the FTP transfer (that is, peers download the file chunks from an FTP site) and JXTA file transfer. We show in Fig. 10 the resulting processing time of log files of 100 Mb when using the best peer found according to economic model and price-based model[9] (the most relevant for this experimental study.) Further, in Fig. 9 the processing time without taking into account the file transmission time (from the master node to peers and vice-versa) is shown.

As can be seen from the above results, as expected, it's worth using the P2P platform to process the log files. In particular sending just one file via FTP takes most of the overall processing time while it is much more efficient to split the file into chunks and send them at the same time to different peers, achieving thus different degrees of granularity. Then, when partitioning the file into chunks, the direct JXTA transfer shows to perform better than the FTP transfer. We noticed however that the file transmission was the most time consuming overall. Regarding the different selection models, they showed different performance. The price-based model with quick peer, which computes the best peer w.r.t. the peer's communication and peer's historical performance showed to perform better. On the other hand, the price-based model with the same priority performed not as good and showed a higher computational cost.

It should be noted however that the performance of different models depends on the state of the network; in particular the economic model could perform better if the provision of task allocation is relevant. To see this effect, we considered the following simple scenario: the log file was split into four chunks and 8 peers were candidates for processing them. The 4 chunks were submitted for processing twice. The results showed now to be different: the price-based model used the four best (fastest) peers for processing the 4 chunks and then used exactly the same peers for processing the second battery of 4 chunks while the economic scheduling model sent the 4 chunks to the four best peers and next sent the second battery of four chunks to the four idle peers.

7. Conclusions and future work

In this work we have presented the Jxta-Overlay, a P2P distributed platform using JXTA technology in which four peer selection models are implemented and experimentally evaluated. These models are the economic scheduling model, priced-based model, peer-priority selection model and random selection model. Their evaluation is done in a real P2P network that uses, among others, nodes of the PlanetLab platform. The performance of the proposed models is studied by using a distributed application scenario for processing large size log files of a virtual campus. The Jxta-Overlay is endowed with a graphical user interface aiming to facilitate the use and configuration of the platform for distributed applications.

In our future work we would like to measure the performance of the proposed peer selection models in large-scale distributed application involving a large number of peers as well as a large number of tasks with interdependencies to be allocated to the peer nodes. Also, we plan to investigate other peer selection models and extend the experimental results of this study.

Acknowledgement

This work has been partially supported by the Spanish MCYT project TSI2005-08225-C07-05.

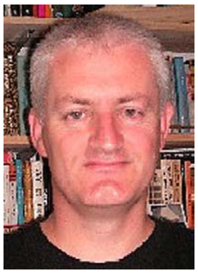
References

[1] L. Alima, A. Ghodsi, S. Haridi, A framework for structured peer-to-peer overlay networks, Global Computing, 2004, pp. 223–249.
[2] D. Andersen, H. Balakrishnan, M. Kaashoek, R. Morris, Resilient overlay networks, Proc. of the 18th ACM Symposium on Operating Systems Principles, 2001, Canada.
[3] D. Andersen, H. Balakrishnan, M. Kaashoek, R. Morris, Experience with an evolving overlay network testbed, ACM SIGCOMM Computer Communication Review 33 (3) (2003) 13–19.
[4] D. Brookshier, D. Govoni, N. Krishnan, J. Soto, JXTA: Java P2P Programming, Sams Pub., 2002
[5] R. Buyya, D. Abramson, J. Giddy, H. Stockinger, Economic models for resource management and scheduling in grid computing, Concurrency and Computation: Practice and Experience 14 (13–15) (2002) 1507–1542.
[6] J. Crowcroft, T. Moreton, I. Pratt, A. Twigg, Peer-to-peer technologies, in: Kesselman, Foster (Eds.), The Grid: Blueprint for a New Computing Infrastructure, chapter 29, Morgan Kaufmann, 2003, pp. 593–622.
[7] C. Ernemann, V. Hamscher, R. Yahyapour, Economic scheduling in grid computing, The 8th Int. Workshop on Job Scheduling Strategies for Parallel Processing, 2002, pp. 128–152.
[8] Common Log Format. April 2008. http://httpd.apache.org/docs/1.3/logs.html#common.
[9] C. Grothoff, An excess-based economic model for resource allocation in peer-to-peer networks, Wirtschaftsinformatik 3 (2003) 285–292.
[10] S. Hotovy, Workload evolution on the Cornell theory center IBM SP2, Proc. of Job Scheduling Strategies for Parallel Proc. Workshop, 1996, pp. 27–40.
[11] H. Hsiao, M. Baker, Ch. King, A peer-to-peer mechanism for resource location and allocation over the grid, ISPA04, 2004, pp. 604–614.
[12] H. Hsiao, Ch. King, Similarity discovery in structured P2P overlays, ICPP03, 2003, p. 636.
[13] J. Hwang, Ch. Lee, J. Song, K. Pyo, Grid and P2P economics and market models, Proc. of the 1st Int. Workshop on Grid Economics and Business Models, IEEE Computer, 2004, pp. 3–18.
[14] Planet Lab. April 2008. http://planet-lab.org/.
[15] Parallel workload archive. The Hebrew University Parallel Systems Lab. April 2008. http://www.cs.huji.ac.il/labs/parallel/workload/.
[16] S. Li, Early Adopter JXTA, Wrox Press, 2003.
[17] S. Oaks, B. Traversat, L. Gong, JXTA in a Nutshell, O'Reilly, 2003.
[18] T. Ping, G. Sodhy, Ch. Yong, F. Haron, R. Buyya, A market-based scheduler for Jxta-based P2P computing system, Int. Conf. Computational Science and Its Applications, Proc., Part IV, vol. 3046 of LNCS, Springer, 2004.
[19] J.E. Riasol, F. Xhafa, Juxta-Cat: a Jxta-based platform for distributed computing, Proc. of the 4th Int. Symp. on Principles and practice of programming in Java, ACM Press, 2006, pp. 72–81.
[20] Seti@Home. April 2008. http://setiathome.berkeley.edu/.
[21] O.4 Wolfson, B. Xu, A. Sistla, An economic model for resource exchange in mobile peer to peer networks, Proc.of the 16th Int. Conference on Scientific and Statistical Database Management, IEEE Computer Soc., 2004, pp. 235–244.
[22] J. Yu, M. Li, Y. Li, F. Hong, M. Gao, A framework for price-based resource allocation on the grid, Proc. of Parallel and Distributed Computing: Applications and Technologies, vol. 3320 of LNCS, Springer, 2004, pp. 341–344.

[8] In the figure abbreviated as "Data".

[9] The notation in the figure reads as follows: EconomyFTP: economic scheduling model using ftp; EconomyTransfer: economic scheduling model using JXTA transfer; PriceFTP s: Price-based model with same priority using ftp; PriceTransfer s: Price-based model with same priority using JXTA transfer; PriceFTP q: Price-based model with quick peer using ftp; PriceTransfer q: Price-based model with quick peer using JXTA transfer.

ARTICLE IN PRESS



Fatos Xhafa received his PhD in Computer Science from the Polytechnic University of Catalonia (Barcelona, Spain) in 1998. He joined the Department of Languages and Informatics Systems of the Polytechnic University of Catalonia as an Assistant Professor in 1996 and is currently Associate Professor and member of the ALBCOM Research Group of this department. His current research interests include parallel algorithms, combinatorial optimization, approximation and meta-heuristics, distributed programming, Grid and P2P computing. His research is supported by several research projects from Spain, European Union and NSF/USA. He has published in leading international journals and conferences and has served in the Organizing Committees of many conferences and workshops. He is currently the Organizing Chair of ARES 2008, PC chair of CISIS 2008, Workshops co-chair of CISIS 2008 and General co-chair of HIS 2008 conferences. He is also a member of the editorial board of several international journals.

Leonard Barolli received his B.E. and PhD degrees from Tirana University and Yamagata University in 1989 and 1997, respectively. From April 1997 to March 1999, he was a JSPS Post Doctor Fellow Researcher at the Department of Electrical and Information Engineering, Yamagata University. From April 1999 to March 2002, he worked as a Research Associate at Department of Public Policy and Social Studies, Yamagata University. From April 2002 to March 2003, he was an Assistant Professor at the Department of Computer Science, Saitama Institute of Technology. From April 2003 to March 2005, he was an Associate Professor, at the Department of Information and Communication Engineering, Fukuoka Institute of Technology (FIT). From April 2005, he is a Professor, at the Department of Information and Communication Engineering, FIT. Dr. Barolli has published more than 200 papers in referred journals and international Conference proceedings. He was an Editor of the IPSJ Journal and has served as a Guest Editor for many international journals. Dr. Barolli has served as a PC member, PC chair, General co-chair and Workshops co-chair of several international conferences. His research interests include network traffic control, fuzzy control, genetic algorithms, agent-based systems, ad-hoc networks, sensor networks, and distance learning systems. He is a member of SOFT, IPSJ, and IEEE.

Thanasis Daradoumis has a PhD in Computer Science from the Polytechnic University of Catalonia, Spain, a Masters in Computer Science from the University of Illinois, and a Bachelors in Mathematics from the University of Thessaloniki, Greece. Since 1984, he has been an Assistant Professor at several universities in the USA, Greece and Spain, teaching a variety of courses in Mathematics and Computer Sciences. Since 1998, he has been working as a Professor in the Department of Computer Science, Multimedia and Telecommunication at the Open University of Catalonia where he coordinates several online courses as well as the development of teaching materials appropriate for virtual learning. He is co-leader of the Distributed, Parallel and Collaborative Systems (DPCS) Research Group at the Open University of Catalonia. His research focuses on e-learning and network technologies, Web-based instruction and evaluation, distributed and adaptive learning, CSCL, CSCW, interaction analysis, and grid technologies.

Raúl Fernández is an Engineer in Informatics from the Faculty of Informatics of Barcelona, Technical University of Catalonia (Barcelona, Spain). He is interested on networked-based applications. He has served as a research fellow at the Open University of Catalonia and has participated in Spanish research projects.

Santi Caballé has a Masters and Bachelors in Computer Science from the Open University of Catalonia. Since 2003, he has been an Assistant Professor at the Open University of Catalonia teaching a variety of courses in Computer Science in the areas of Information Systems, Software Engineering and Collaborative Learning. Since early 2006 he has been working as an Associate Professor of the Department of Computer Science, Multimedia and Telecommunication at the Open University of Catalonia where he coordinates several online courses in the area of Software Engineering. He is a member of the Distributed, Parallel and Collaborative Systems Research Group at the Open University of Catalonia, where he is currently carrying out his PhD. His research focuses on e-learning, software engineering, network technologies, distributed learning, computer-supported collaborative learning, interaction analysis, and grid technologies.