Performance scalability of the JXTA P2P framework

Gabriel Antoniu, Loïc Cudennec, Mathieu Jan, Mike Duigou

Performance scalability of the JXTA P2P framework

Gabriel Antoniu[1], Loïc Cudennec[1], Mathieu Jan[1] and Mike Duigou[2]

[1] IRISA/INRIA
Campus de Beaulieu
35042 Rennes Cedex, France
Mathieu.Jan@irisa.fr

[2] Project JXTA
Sun Microsystems
Santa Clara, CA, U.S.A
Mike.Duigou@sun.com

Abstract

Features of the P2P model, such as scalability and volatility tolerance, have motivated its use in distributed systems. Several generic P2P libraries have been proposed for building distributed applications. However, very few experimental evaluations of these frameworks have been conducted, especially at large scales. Such experimental analyses are important, since they can help system designers to optimize P2P protocols and better understand the benefits of the P2P model. This is particularly important when the P2P model is applied to special use cases, such as grid computing. This paper focuses on the scalability of two main protocols proposed by the JXTA *P2P platform. First, we provide a detailed description of the underlying mechanisms used by* JXTA *to manage its overlay and propagate messages over it: the rendezvous protocol. Second, we describe the discovery protocol used to find resources inside a* JXTA *network. We then report a detailed, large-scale, multi-site experimental evaluation of these protocols, using the nine clusters of the French Grid'5000 testbed.*

1. Introduction

Programming distributed applications has always been a difficult task. This difficulty is further increased when systems target a large scale (i.e. millions of users). While trying to address this problem, the Peer-to-Peer (P2P) approach is receiving a growing interest thanks to its properties, primarily scalability and volatility tolerance. Consequently, by using a P2P model, applications can hope to offer higher scalability and availability despite dynamic changes in the underlying physical infrastructure. The P2P model was initially used for large-scale applications over Internet (such as file sharing, instant messaging, etc.), but has also been found attractive in the field of distributed scientific simulation on grid infrastructures. As an example, it can typically be used in connection with grid resource management middleware [9, 23].

Recently, a number of P2P libraries (e.g. Bamboo [28], FreePastry [29], JXTA [27], etc.) providing basic support for P2P interaction (for example discovery mechanisms) have been made available. Such libraries are intended to serve as generic building blocks for higher-level P2P services and applications. Among these P2P libraries, the JXTA framework is emerging as a de facto standard for building services or applications in the industrial world [30]. It is also used in various research projects, [1, 2, 21] to cite a few (see [31] for a more detailed list).

However, before using such generic layers, it is important to analyze their suitability with respect to the requirements of the target P2P service or application. Most published papers introducing these libraries give the cost of basic operations (e.g. routing and discovery) through complexity analyses and simulations. These theoretical evaluations are certainly necessary, but they clearly are only a preliminary step. To fully understand the behavior of the proposed P2P libraries, *experimental evaluations* on existing distributed testbeds are unavoidable. Such practical evaluations make it possible to better tune the proposed algorithms, depending for instance on the testbed scale or on the underlying network topology targeted by applications. For instance, the current convergence of grid computing and P2P computing [10] calls for precise requirements and guarantees to be defined for P2P algorithms. This makes it possible to better understand the benefits of the P2P model in such particular use cases.

In this work, we focus on the performance of the JXTA protocols. JXTA is an open-source initiative, sparked by Sun Microsystems. It was founded in order to develop a set of standard open protocols for P2P network applications. To the best of our knowledge, it is the most ad-

1-4244-0910-1/07/$20.00

vanced framework currently available for building services and applications based on the P2P model. In its 2.0 version, JXTA consists of a specification of six language- and platform-independent, XML-based protocols [32] that provide basic services common to most P2P applications, such as peer group organization, resource discovery, and inter-peer communication. A more detailed overview of JXTA can be found in [27]. In this paper, we focus on the performance analysis of the so-called *Loosely-Consistent Distributed Hash Table* (LC-DHT) [26] at a large scale. This mechanism has been introduced in JXTA 2.0 for resource discovery.

In order to evaluate the cost of this LC-DHT, we benchmark the *discovery protocol* from an application point of view. However, we also evaluate the underlying algorithms used by the LC-DHT, i.e. the *rendezvous protocol*, used to organize the JXTA overlay and propagate queries. We perform multi-site tests over the Grid'5000 testbed [6], an experimental grid platform consisting of 9 sites geographically distributed in France. Grid'5000 aims at gathering a total of 5,000 CPUs in the near future. We vary parameters of these algorithms: the overlay size and some parameters controlling the algorithm behavior. We also experiment two overlay topologies: chains and trees. These metrics are applied on the C reference implementation of the JXTA specifications, known as JXTA-C [25].

The remainder of the paper is organized as follows. Section 2 introduces related work: we discuss some existing performance evaluations of DHTs as well as of JXTA. Section 3 provides an overview of JXTA's protocol stack, with a focus on protocols involved in the management of the LC-DHT evaluated in our experiments. Section 4 presents and discusses scalability experiments for the rendezvous protocol and for the discovery protocol. Finally, Section 5 concludes the paper and suggests directions for further research.

2. Related work

Papers introducing DHTs such as Pastry [19], have evaluated the benefits of this approach, but mainly through theoretical analyses and simulations. The most commonly used metric to evaluate DHTs is the number of hops required to look for a resource in a given overlay. Because of the use of simulations, past evaluations usually assume a static network. More recently, the impact of realistic conditions, such as for instance churns, on the performance of the lookup operation have been the subject of many papers [16, 18, 22]. Such work is sometimes based on theoretical analyses [17], but more usually on laws modeling the session length of a peer [16, 18] as well as on traces of existing deployed systems [22]. However, note that traces are subject to inaccuracies: only a subset of the network has been explored in a time-constrained period and lookup patterns might therefore not be representative. Large-scale experimental evaluations of DHTs in controlled environments are therefore required to compare any proposed improvements in a fair manner as well as to report performances of DHTs using *traditional* network metrics, such as latency, bandwidth, etc. To the best of our knowledge, only [8] provides such experimental evaluations, using 425 peers over 150 nodes of the PlanetLab [7] testbed and takes into account underlying network characteristics in the used metrics.

In the JXTA field, various implementations of the JXTA specification have been evaluated [11, 12, 20] and compared to other systems [5, 14] (especially as regards JXTA-J2SE). However most of these evaluations were performed at application-level, without any analysis of the internal behavior of JXTA's protocols. Such an analysis has recently been provided for the performance of JXTA's communication layers [3, 4]. The goal of this paper is to bring a similar contribution by analyzing and evaluating JXTA's LC-DHT mechanism, on which relies JXTA's protocols for resource publishing and discovery.

The LC-DHT has been introduced in JXTA-J2SE and JXTA-C starting from versions 2.0 and 2.2 respectively. The approach has been compared to a classical DHT-based approach in [24], by modifying JXTA's discovery protocol. However, no performance evaluation was reported. In [13] authors compare the LC-DHT approach to a centralized or flooding approach (which was the strategy used by JXTA 1.0), with respect to memory usage, reliability and query response time for different configurations of a JXTA virtual network. Benchmarks were performed against JXTA-J2SE. However, no benchmarks have been performed at a large-scale, as the experimental configurations used up to 32 peers only, based on a LC-DHT distributed on 4 peers. One of the main goals of this paper is precisely to test the scalability of the LC-DHT approach, using much larger configurations.

3. Description of JXTA

3.1. General overview

JXTA relies on a set of basic concepts: a "peer" is an entity able to communicate by exchanging messages; a "peer group" (also called an overlay in the remainder of this paper) is a set of peers with a common interest, and providing common services; an "advertisement" is an XML document describing a resource. JXTA specifies a set of language- and platform-independent XML-based protocols [32]. These protocols are used inside each peer group to provide a rich set of building blocks (called services) for the management of peer-to-peer systems: resource dis-

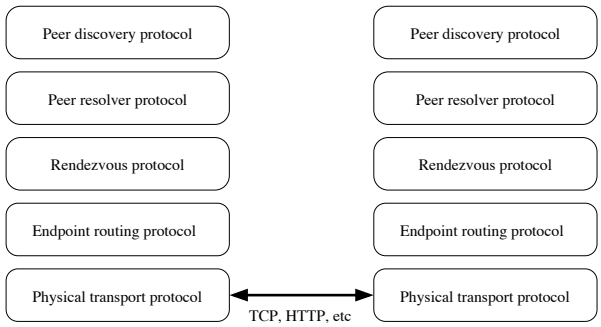

Figure 1. Partial view of JXTA's protocol stack as defined by the specification.

covery, overlay self-configuration, peer-to-peer communication, etc.

Figure 1 shows the stack of protocols defined by JXTA. Above the physical transport protocols, the *endpoint routing protocol* (ERP) is used to find available routes from a source peer to a destination peer. The *rendezvous protocol* is responsible for message distribution and topology management. On top of the rendezvous protocol, JXTA uses a standardized query/response protocol: the *resolver protocol*. It provides a generic, topology-independent query/response interface which other higher-level services may use to invoke various P2P operations. For instance, the *discovery protocol* defines one such service, which provides a specific API for publishing and discovery, which is implemented using the resolver protocol. The discovery protocol is used to publish and to find available resources within a JXTA overlay. It relies on the use of resource advertisements, whose life duration can be controlled via the discovery API. Discovery resource announcements and queries are forwarded by the underlying resolver protocol to the appropriate rendezvous peers.

In this paper, we focus on the *rendezvous protocol* and on the *discovery protocol*. Before introducing in more details these two protocols, let us stress that, for each protocol, JXTA only defines *the syntax* for queries and responses; it does not specify the behavior of algorithms used by implementations willing to be compliant with the protocol specification. For instance, various message routing approaches can be used by JXTA implementations, making JXTA the perfect framework for testing topology-based routing algorithms. Moreover, the specification does not define the structure of a JXTA-based overlay. In current implementations (JXTA-C or JXTA-J2SE), a JXTA overlay is a structured network based on the use of mainly two peer types: super-peers, commonly called *rendezvous* peers, and regular peers, called *edge* peers. Each edge peer is attached to a rendezvous peer. Queries and responses are forwarded among the super-peers using a *Loosely-Consistent Distributed Hash Table* (LC-DHT), whose behavior is detailed in Section 3.3. However, note that alternate implementations of JXTA may provide different types of overlays.

Let $\mathbf{S}$ be a JXTA overlay, composed of r rendezvous peers and e edge peers. If rendezvous peers are noted R_i, and edge peers are noted E_j, then $\mathbf{S}$ is defined as follows:

$$\mathbf{S} = \{R_i, i = 1..r\} \cup \{E_j, j = 1..e\} \quad (1)$$

3.2. The peerview protocol: managing the overlay

As stated by the JXTA specifications, the rendezvous protocol is divided into three sub-protocols:

1. the *peerview protocol*, used by rendezvous peers to organize themselves by synchronizing their views of each other;
2. the *rendezvous lease protocol*, used by edge peers to subscribe to the reception of messages propagated by the rendezvous peers;
3. the *rendezvous propagation protocol*, which enables peers to manage the propagation of individual messages within a group.

In this paper, we focus on the first protocol: the peerview protocol. This protocol allows rendezvous peers to work together to form a so-called *global peerview*: an ordered list (by peer ID) of peers currently acting as rendezvous peers within a given group. This list is used to route messages within that group. Each rendezvous peer maintains a local version of the list, which represents its view of the global peerview. Let g be the size of the global peerview for a given peer group $\mathbf{S}$, and l_i the size of local peerview of rendezvous peer R_i. Let $\mathbf{T}$ be the set of values that can take the time during a given execution of a JXTA application. The goal of the peerview protocol is to form and keep the individual local peerviews consistent across all rendezvous peers participating in a given group. This can be translated in the following property:

$$\exists\, t_1 \in \mathbf{T}, \forall\, t_2 \in \mathbf{T}, t_2 > t_1, \forall\, R_i \in \mathbf{S} : l_i = g \quad (2)$$

We shall see in Section 3.3 that this consistency is required for an optimal query propagation over a JXTA overlay.

To achieve this goal, rendezvous peers periodically probe other members of the peerview. Algorithm 1 shows the main steps of this process used by JXTA-C for this purpose. The algorithm is run by every rendezvous peer of $\mathbf{S}$. The elapsed time between two iterations of the algorithm is controlled by the `PEERVIEW_INTERVAL` constant (set by default to 30 seconds). Note that the algorithm

Algorithm 1: Pseudo-code of the main steps of the periodic algorithm used for the convergence of local peerviews of R_i.

```
1  repeat
2      wait for PEERVIEW_INTERVAL;
3      remove entries from the local peerview for which
       time > PVE_EXPIRATION;
4      l_i = size of the local peerview of R_i;
5      for rdv ∈{upper_rdv, lower_rdv} do
6          if l_i < HAPPY_SIZE then
7              probe rdv;
8          else
9              if rand() % 3 == 0 then
10                 update our entry in the peerview of
                   rdv;
11             else
12                 probe rdv;
13     if l_i < HAPPY_SIZE then
14         probe initial rendezvous peers, called seed
           rendezvous;
15 until rendezvous service is stopped ;
```

used by JXTA-J2SE is slightly different, but the variation in behavior does not appear to be significant. Apart from the peer running the algorithm, two other peers are mainly involved: 1) the rendezvous peer whose peer ID immediately preceeds the local peer ID in the sorted list of peer IDs, noted as `lower_rdv` in the algorithm and 2) the rendezvous peer whose peer ID immediately follows the local peer ID in the same list, noted as `upper_rdv` in the algorithm. Theses two rendezvous peers, if present (peers at each end of the sorted list will have only one peer to probe), are more actively probed if the size of the peerview has not reached a configurable minimum threshold: `HAPPY_SIZE` (by default set to 4). A probe is a peerview message that contains a *rendezvous advertisement* describing the sender (let us call this peer A). In response to a probe, the receiver (also a rendezvous peer, let us call it B) returns its own rendezvous advertisement. In a separate message, peer B will also return a randomly chosen rendezvous advertisement for another rendezvous peer in his list (C). This second message is known as a *referral response*. This way, the initiator of the probe (A) may learn about a new rendezvous peer. However, before adding this new rendezvous advertisement in its local peerview, peer A will probe peer C, which will also send a referral response publishing the identity of yet another rendezvous peer (D), and so on. The default lifetime of rendezvous advertisements in the peerview is controlled via a constant (`PVE_EXPIRATION`, by default set to 20 minutes). This tunable constant is used at line 3 of Algorithm 1 to remove expired peerview entries.

3.3. The discovery protocol: publishing and discovering resources

An overview of the LC-DHT algorithm. The LC-DHT algorithm (for *Loosely-Consistent Distributed Hash Table*) defines a mechanism for publishing and discovering resources within a JXTA overlay. Each resource made available is described by an advertisement. Each type of advertisement defines a set of attributes by which instances of the advertisement are indexed. Peers maintain and publish attribute tables for their advertisements. An attribute table consists of tuples (index attribute, value), each of which is associated to a life duration and to the identity of the publishing peer. These attribute tables are published by the edge peers to their associated rendezvous peers. Rendezvous peers which receive such publication requests keep a copy of the tuples and then replicate them across the entire network of rendezvous by applying a hash function on each tuple, in order to determine on which rendezvous peer to replicate the table. Such a rendezvous peer designated by the LC-DHT algorithm as responsible for an advertisement is called *replica peer* for that advertisement. The hash function (detailed below) shows that the mapping of indexes to the appropriate replica peer is based on the use of the local peerview (we remind the reader that l_i is the size of the local peerview):

Function `ReplicaPeer(`*tuple*`)` applied by peer R_i member of $\mathbf{S}$ to find the replica peer for a given advertisement.

1 $hash =$ `SHA-1` $(tuple)$;
2 $pos =$ floor$(hash * \frac{l_i}{MAX_HASH})$;
3 Return peerview entry at position pos;

The hash is actually applied on a string obtained by concatenating the type of the advertisement, the name of the attribute used for indexing and its value. The maximum value the hash function of the LC-DHT can return is noted `MAX_HASH`.

Let us take an example to illustrate both publish and lookup operations for a resource (with $\mathbf{S} = \{R_i, i = 1..6\} \cup \{E_j, j = 1..2\}$). In our example, the resource is a peer represented by a peer advertisement `Adv` (so the peer type is `Peer`); let us assume that the index attribute is `Name` and its associated value is `Test`. The hash function will then be applied to the string: "PeerNameTest". Let us assume that the output hash value is 116, that `MAX_HASH` equals 200 and that the property (2) is satisfied ($l_i = g = 6$). Consequently, the Table 1 shows local peerviews of each R_i member of $\mathbf{S}$.

Peerview entry	0	1	2	3	4	5
Peer ID (R_i)	006 (R_1)	020 (R_2)	036 (R_3)	050 (R_4)	088 (R_5)	180 (R_6)

Table 1. Local peerview used by each R_i member of S for both publish and lookup operations of tuple (116, E1).

Publishing. The top side of Figure 2 shows the main steps for publishing this advertisement, which are described below.

1. The tuple is first sent by the edge peer E_1 to its associated rendezvous peer R_1, which can compute the appropriate replica peer for the advertisement. According to the *replica peer* function ($position = \lfloor \frac{116*6}{200} \rfloor = 3$), the tuple (116, E1) will get replicated onto the third peer in R_1's peerview, which is rendezvous peer R_4 (see Table 1). The tuple (116, E1) is also stored by R_1 to increase its availability for edge peers connected to the same rendezvous peer.
2. The tuple (116, E1) is therefore sent on the rendezvous peer R_4.

Discovery. The goal of the peer discovery protocol is to find resources within the group. It makes use of the tuples described above. Edge peers specify queries by indicating index attributes and desired values for those attributes. This is expressed as a discovery protocol message, which is propagated over a JXTA overlay according to the following steps (bottom side of Figure 2).

1. The message is first forwarded by the edge peer to its rendezvous peer via the resolver protocol (for rendezvous peers this step is not necessary as they act as their own rendezvous). In our example, edge peer E_2 is looking up for `Adv`: the discovery request is sent to R_2, which is E_2's rendezvous peer.
2. Rendezvous peer R_2 first checks if it has a local hash value of 116. In that case, R_2 can directly forwards the query to E_1 and the lookup process continues at step 4. If no local matching hash value if found, R_2 applies the same hash function as used for publication upon the query expression elements. Consequently, the query is forwarded to the resulting replica peer: R_4.
3. As R_4 is indeed the correct replica peer for this query, the query is forwarded to the edge peer that has published the `Adv` advertisement: E_1.
4. Finally, edge peer E_1 sends its advertisement to the requesting edge peer (E_2).

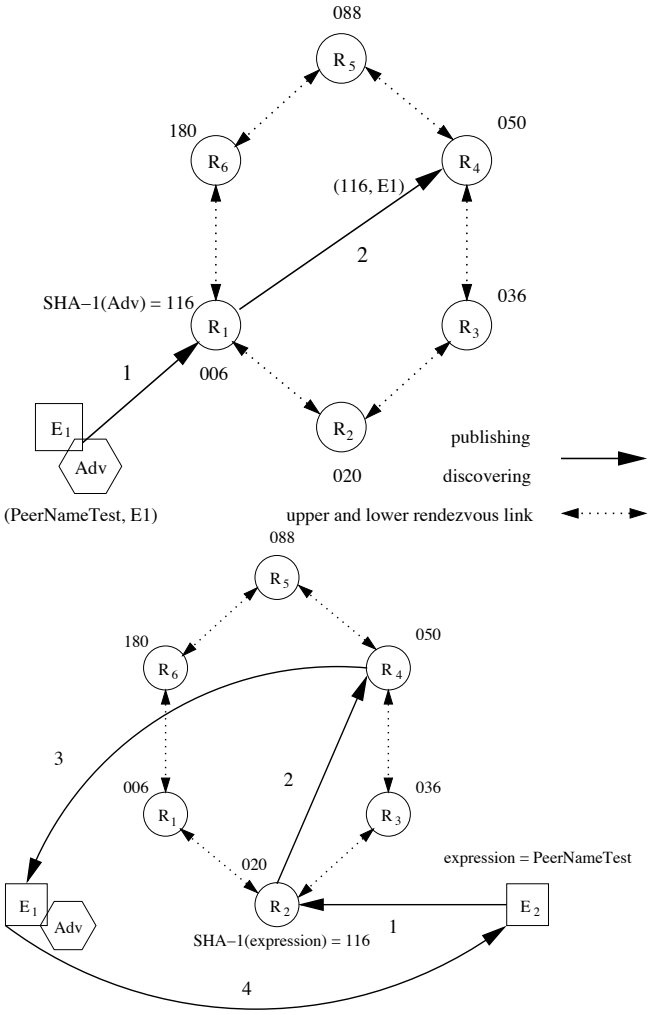

Figure 2. Main steps for publishing (top) and looking up (bottom) a resource advertisement over the network of rendezvous peers. In this example, the advertisement has a hash value of 116.

Link with the peerview protocol. When publishing or looking up for advertisements, local peerviews of rendezvous peers are used to forward messages to replica peers. However, if rendezvous peers have different local peerviews they will route messages to different destination rendezvous peers: this is due to the fact that the replica peer is computed based on the rank of the rendezvous peers in the local peerviews. If local peerviews differ on the rendezvous peer performing the publication and on the rendezvous peer in charge of performing the corresponding discovery query, the replica peer computed at publication will not match the one computed during the discovery process. However, upon failing to find a resource on a replica peer, a backup mechanism is used: the query will be forwarded to the upper and lower rendezvous peers, which may store the resource. The query is said to *walk* the whole peerview in both directions: towards upper values in the hash space starting from the `upper_rdv` rendezvous peer, and towards lower values in the hash space starting from the `lower_rdv` rendezvous peer.

Complexity. On an overlay gathering n nodes, classical DHTs have a complexity in $O(log\ n)$ for publishing resources[1], whereas LC-DHT have a complexity in $O(1)$ (2 messages in the worst case). However for discovery, a LC-DHT based approach does not offer the same $O(log\ n)$ query guarantee that other DHT systems provide. Current implementations of JXTA provide a $O(r)$ query guarantee, where r is the number of rendezvous peers. The worst case corresponds to the use of the walk mechanism upon failure of the hash-based discovery (due to high churn), more precisely when the rendezvous peer responsible for the searched advertisement is diametrically opposed in the peerview of the rendezvous peer computed by the hash function. Therefore the peerview must be walked and the number of hops used is then $O(r)$. In practice, with the help of advertisement replication, this number of hops is much less than r. Finally, note that if local peerviews of all rendezvous peers of a given group satisfy the property (2), the complexity is only in $O(1)$ (actually 4 messages in the worst case, as illustrated on the bottom side of Figure 2).

The LC-DHT approach avoids the expensive traffic (and, often more importantly, latency overhead) required by classical DHTs to maintain consistency. On the other hand, if rendezvous peers are volatile, peerviews may change, and computed replica peers may not be correct, as it would be theoretically the case with classical DHTs. However, studies of these classical DHTs showed that their maintenance mechanisms are unable to cope with churns of a few tens of minutes [18], which most peers of existing systems follow according to the same paper. Note that JXTA edge peers periodically push tuples of updated or new indexes to their rendezvous peers (by default every 30 seconds). However, this is only done if advertisements have changed or have been explicitly republished by applications. Consequently, this may lead to the computation of new replica peers, to allow future discovery requests to complete correctly. On the other hand, edge peers also publish their tuples whenever they connect to a new rendezvous peer.

The drawback of this LC-DHT approach is therefore that it is possible to deliver incomplete results in the case of highly inconsistent peerviews. This explains the name given to this approach: *loosely-consistent DHT*, whose aim is to cope with highly-dynamic peer-to-peer networks.

[1] With respect to the number of required hops.

4. Scalability evaluation of JXTA-C protocols

Experimental setup. Nodes used for the experiments mainly consist of machines using dual or quadri 2.2 GHz AMD Opteron, dual 900 MHz Intel Itanium2, outfitted with up to 4 GB of RAM each, and running a 2.6 version Linux kernel; the hardware network layer used is a Giga Ethernet (1 Gb/s) network. All 9 sites of the Grid'5000 testbed were used: Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia and Toulouse. Tests were executed using JXTA-C version 2.3 (released the 15th of December 2005), used and configured to use TCP as the underlying transport protocol. JXTA-C benchmarks are compiled using `gcc` 4.0 with the `-O2` level of optimization. Finally, for the deployment of JXTA overlays, we used the generic deployment tool ADAGE [15]. In order to do this, we developed a JXTA plug-in for ADAGE so that overlays can be described in a concise manner, and generation of configuration files for JXTA automated. We performed two different kind of tests, as detailed below.

4.1. Evaluation of the peerview protocol

Benchmark description. The goal of this benchmark is to measure the time it takes for the peerview protocol to make the LC-DHT consistent across rendezvous peers, that is to say the time for the property (2) to be satisfied. Each time a rdv peer is added to, or removed from the local peerview of a rendezvous peer, the elapsed time since the beginning of the test is logged, as well as the type of event. We performed benchmarks using an increasing number of rendezvous peers, with different logical topologies (chain or tree) and for different values of the `PVE_EXPIRATION` constant. As in Section 3.1, we note r the number of rendezvous peers in a peer group (noted $\mathbf{S}$ as introduced in the same Section) and l the size of the local peerview of a rendezvous peer. Finally, note that in our benchmark $\mathbf{S}$ is only made of rendezvous peers ($\mathbf{S} = \{R_i, i = 1..r\}$).

Scalability of the peerview protocol. The top side of Figure 3 shows the evolution of l according to r. Both chains (r equals to 10, 45, 50, 80, 160, 580) and trees (160, 220, 338) topologies have been tested, revealing that this initial parameter has no significant influence on the peerview behavior. For a same experiment, the value l of each rendezvous peer belonging to $\mathbf{S}$ evolves in the same way. When $r \geq 45$, the property (2) is not satisfied. Moreover, only three experiments (r equals to 10, 45 and 50) reach the maximal possible value for $\mathbf{S}$. This maximal possible value, noted m, is equal to $r - 1$[2]. Finally, the experiment with $n = 580$ enables to clearly distinguish three phases in the setup on the local peerview of a rendezvous peer.

1. An increase of l. This phase lasts as long as the lifetime duration of the advertisements of rendezvous peers (controlled by the constant `PVE_EXPIRATION`, see Section 3.2), which is 20 minutes for the given experiment.

2. A decrease of l. This phase starts at the time `PVE_EXPIRATION`. If after this time the rendezvous associated to its entry in the peerview is not probed by the algorithm of the peerview, the entry is removed from it.

3. A fluctuation of l around a given value, depending of the value of n. In our case, l is around 300. In the worst case, this phase starts at a time around twice the value of `PVE_EXPIRATION` in minutes (experiment with $r = 580$).

The bottom side of Figure 3 shows the distribution of adding and removal events (respectively depicted by rhombus and cross) of rendezvous peers in the local peerview of a rendezvous peer (where $r = 580$). More precisely, on the y axis is shown the number of a given rendezvous peer[3] of the experiment. Results show that almost all rendezvous peers are getting in touch with all other rendezvous peers. The maximal given number of the last known rendezvous peer is indeed 577, that is to say 2 rendezvous peers were not discovered. This occurs 117 minutes after the beginning of the experiment. Two phases can also be observed for this curve:

1. First, only adding events of rendezvous peers in the peerview occur. This phase lasts `PVE_EXPIRATION` (20 minutes in our experiment). It starts with a *curve of adding events* in local peerviews, as shown by the first curve, made of rhombus, on the bottom side of Figure 3.

[2]Our measurement excludes the local rendezvous peer from the size of the peerview.

[3]For each new rendezvous peer added in the peerview, a number is given to the rendezvous peer starting from 1.

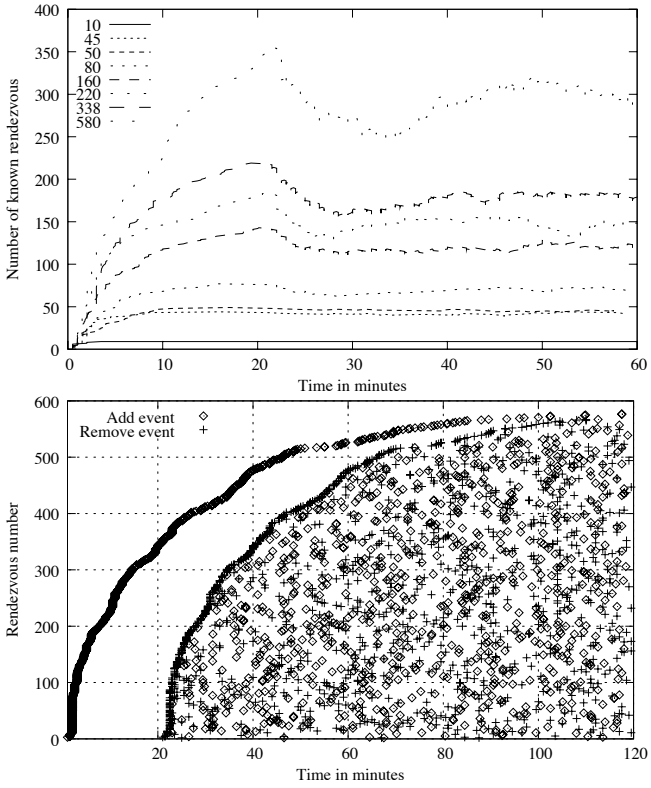

Figure 3. Evolution of the size l of the peerview of a rendezvous peer according to the total number r of rendezvous peers in the system (top). Distribution of add and remove events of rendezvous peers in the peerview of a rendezvous-peer (bottom).

2. Both adding and removal events of rendezvous peers in the peerview are observed. This phase starts at time `PVE_EXPIRATION` and lasts till the end of the experiment. It starts with a *curve of removal events* in local peerviews, as shown by the second curve, made of crosses, on the bottom side of Figure 3.

Figure 3 (bottom side) explains the third phase in the setup of the peerview of a rendezvous peer. The fluctuation of the value m corresponds indeed to the adding and removal of entries in the peerview, which occur in this second and last phase. This indicate the incapacity of the peerview protocol to probe all the entries of the peerview, in a time shorter than the value of the constant `PVE_EXPIRATION`. Moreover, this figure also explains why the value of m culminates in time to the value of the constant `PVE_EXPIRATION`. After this time, the curve of adding events counterbalances the curve of removal events, thereby explaining the phase 3 in the setup of a local peerview.

Discussion. This benchmark allows to characterize the behavior of the algorithm of the peerview protocol of JXTA-C. It tries to give a reliable answer to the following question: how many rendezvous peers can be deployed in a JXTA group? This question is frequently asked in the mailing lists of JXTA and until now answered without any proof. The peerview algorithm was believed to work well for values of n as high as 100 or 150. However, our evaluation shows that even for n greater than 45, the property (2), which expresses the goal of the peerview protocol, is not satisfied. Consequently, with parameters of the peerview protocol set to *default values*, a JXTA group made of 45 rendezvous peers is not able to optimally organize itself. Therefore, the most efficient routing of requests is not guaranteed, leading to an increase in the number of hops needed for a discovery request for example (see Section 3.3).

A solution is to modify the value of the constant `PVE_EXPIRATION`, as shown by the following experiment. Figure 4 shows the evolution of the value of m on a rendezvous peer (with $r = 50$), according to two different values for the constant `PVE_EXPIRATION`. By changing this constant to a time greater than the duration of the experiment (60 minutes in our case), l reaches its maximum possible value: $r - 1$, which in our case is 49. In Property (2), t_1 is therefore equal to 17 minutes. Another solution to support a higher number of rendezvous peers is to decrease the interval of time between the iterations of the peerview algorithm loop (constant `PEERVIEW_INTERVAL`, see Section 3.2). Note that both parameters can be tuned at the same time.

In all cases, a compromise must be reached between freshness (and thereby reliability of information in the peerview) on one side and bandwidth consumption on the other side. The freshness of information decreases when the value of the constant `PVE_EXPIRATION` increases, whereas the bandwidth consumption increases whenever the value of the constant `PEERVIEW_INTERVAL` increases.

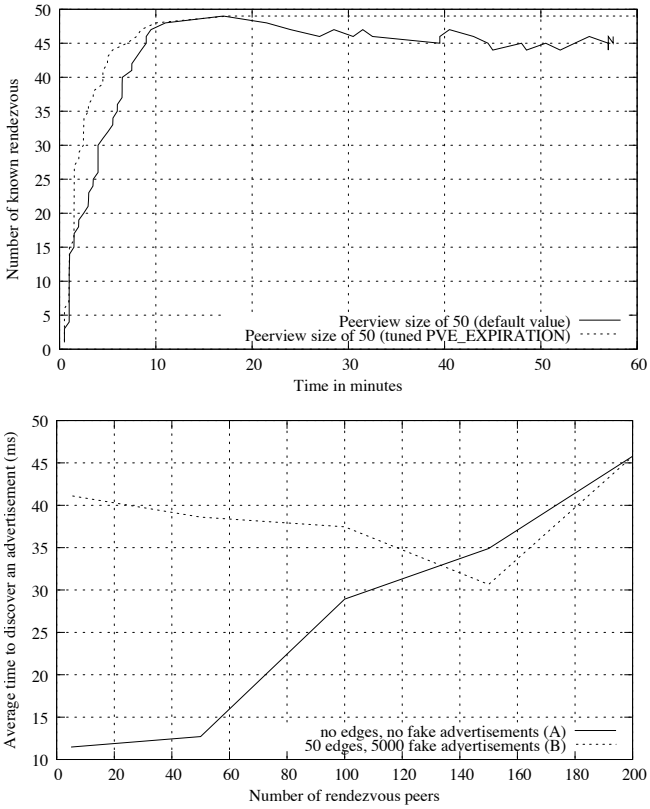

Figure 4. Evolution of the size l of the peerview of a rendezvous peer, for a P2P system made of 50 rendezvous peers, according to the value of the parameter `PVE_EXPIRATION` (top). Time t to discover an advertisement depending on the number of rendezvous peers r, edge peers and fake advertisements f (bottom).

4.2. Evaluation of the discovery protocol

Benchmark description. The goal of this benchmark is to evaluate the time t needed for an edge peer to retrieve an advertisement. In our setup, a network of r rendezvous peers is deployed to route discovery messages. One edge peer (called *publisher*) connects to this network and publishes a specific advertisement that is then searched by another edge peer (called *searcher*). All measurements are calculated based on 100 consecutive queries, each of them followed by a flush of the local *searcher* cache, in order to avoid cache speedup. Publishing and searching jobs delay their execution time after that local peerviews of rendezvous peer entered their phase 3 (see explanations of Figure 3). A first set of experiments involves a *publisher*, a *searcher* and an increasing number of rendezvous peers (configuration A). No edge peers are attached to rendezvous peers but the *publisher* and the *searcher* ($\mathbf{S} = \{R_i, i = 1..r\} \cup \{E_j, j = 1..2\}$). The second set of experiments extends the first one by adding edge peers to some rendezvous peers (configuration B). More precisely, 50 edge peers will connect to 5 rendezvous peers amongst the r available ($\mathbf{S} = \{R_i, i = 1..r\} \cup \{E_j, j = 1..52\}$). Each peer of this class, called *noisers*, publish a specified number of random advertisements f, called *fake advertisement*, to its rendezvous peer. The goal of this second experiment is to stress the discovery protocol by running it in more realistic conditions, where applications publish multiple advertisements concurrently. This will allow us to measure the impact of concurrency on the discovery time.

Scalability of the discovery protocol. The bottom side of Figure 4 shows the average time t needed to discover a given advertisement, according to the number of rendezvous peers r. For each value of r we measure the discovery time with (B) and without (A) extra "noise" produced by 50 *noiser* edges (with $f = 100$). The (A) curve shows that adding up to 50 rendezvous peers does not significantly increase the discovery time: t remains around 12 ms. From 50 to 200 rendezvous peers, the discovery time grows linearly. This is explained by the behavior described above for the peerview protocol: when using such a large number of rendezvous peers, Property (2), which is related to the stabilization of the peerview, cannot be satisfied. Therefore, the local peerviews of rendezvous peers are not consistent, forcing the LC-DHT algorithm to *walk* the global peerview. This represents a linear cost with respect to the number of rendezvous peers r, as stated in the complexity discussion of Section 3.3. The (B) curve shows the impact of the "noise" on the discovery time, while a total of 5,000 *fake advertisements* are published by the 50 *noisers*. The maximum overhead is measured for $r = 5$ (30 ms), i.e. when *noisers* are attached to each rendezvous peer of the network. Then, for values of r up to 150, this overhead slightly decreases. This can be explained by the load balancing of publishing queries onto different replica peers, as we use the same number of edge peers e (always equal to 50). From $r = 150$ to $r = 200$, publishing *fake advertisements* no longer influence the discovery time.

Discussion. As seen in Section 2, DHT algorithms are usually benchmarked using the number of hops required to perform a lookup. In addition to such an analysis of the discovery protocol (see Section 3.3), we could measure and explain the latency of discovery requests. Such results enable to tune applications based on this protocol depending on the network size (r) and on the number of published advertisements. Therefore, they help finding an answer to the following question: when should the application re-launch a discovery query?

5. Conclusion

The interesting features of the P2P model have made it attractive for both the academic and industrial world. Several generic P2P frameworks are now available to developers wishing to use this model for their applications. However, very few experimental evaluations of these P2P libraries have been reported, especially at large scales.

In this paper, we focus on a de facto standard of P2P programming: JXTA specifications. We provide a detailed analysis of the protocol used to manage a JXTA overlay, namely the peerview protocol. We also perform an analysis of the protocol used to lookup for resources: the discovery protocol. The theoretical behavior of both protocols is described and multi-site experimental tests are reported, using the French Grid'5000 testbed with various JXTA-C overlay configurations. The goal of these benchmarks is to answer a common and unanswered question on the JXTA mailing lists: how many rendezvous peers are supported by JXTA in a given group? Our results show that with default values for parameters of the peerview protocol, the goal of the algorithm is not achieved, even with as few as 45 rendezvous peers. However, parameter tuning makes it possible to reach larger configurations in terms of number of rendezvous peers. For the discovery protocol, we show that discovery time is rather smaller, provided that all rendezvous peers satisfy a given property. These results give developers a better view of the scalability of JXTA protocols.

Nevertheless, considering all the factors explored in this paper, this research is not an exhaustive evaluation of the scalability of JXTA protocols. In particular, no volatility was introduced during the experiments. For instance, it would be interesting to evaluate the behaviour of fall-back mechanism used for resource discovery under high volatility. Further experiments should also evaluate the mechanisms used by JXTA-C to address complex queries, such as range queries.

References

[1] M. Amoretti, G. Conte, M. Reggiani, and F. Zanichelli. Service Discovery in a Grid-based Peer-to-Peer Architecture. In *International Workshop on e-Business and Model Based IT Systems Design*, Saint Petersburg, Russia, Apr. 2004.

[2] G. Antoniu, L. Bougé, and M. Jan. JuxMem: An Adaptive Supportive Platform for Data Sharing on the Grid. *Scalable Computing: Practice and Experience*, 6(3):45–55, September 2005.

[3] G. Antoniu, P. Hatcher, M. Jan, and D. A. Noblet. Performance Evaluation of JXTA Communication Layers. In *Proc. Workshop on Global and Peer-to-Peer Computing (GP2PC 2005)*, pages 251–258, Cardiff, UK, May 2005. Held in conjunction with the 5th IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid 2005).

[4] G. Antoniu, M. Jan, and D. A. Noblet. Enabling the P2P JXTA Platform for High-Performance Networking Grid Infrastructures. In *Proc. of the first Intl. Conf. on High Performance Computing and Communications (HPCC '05)*, number 3726 in Lect. Notes in Comp. Science, pages 429–440, Sorrento, Italy, September 2005. Springer.

[5] S. Baehni, P. T. Eugster, and R. Guerraoui. OS Support for P2P Programming: a Case for TPS. In *22nd International Conference on Distributed Computing Systems (ICDCS '02)*, pages 355–362, Vienna, Austria, July 2002. IEEE Computer Society.

[6] F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, N. Melab, G. Mornet,

R. Namyst, P. Primet, and O. Richard. Grid'5000: A Large Scale, Reconfigurable, Controlable and Monitorable Grid Platform. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing (Grid '05)*, pages 99–106, Seattle, WA, USA, Nov. 2005.

[7] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, July 2003.

[8] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a DHT for Low Latency and High Throughput. In *Proc. of the 1st Symposium on Networked Systems Design and Implementation (NSDI 2004)*, pages 85–98, San Francisco, CA, USA, Mar. 2004. USENIX.

[9] N. Drost, R. van Nieuwpoort, and H. E. Bal. Simple Locality-Aware Co-allocation in Peer-to-Peer Supercomputing. In *Proceedings of the 6th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid '06)*, May 2006.

[10] I. Foster and A. Iamnitchi. On Death, Taxes, and the Convergence on Peer-to-Peer and Grid Computing. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, number 2735 in Lect. Notes in Comp. Science, Berkeley, CA, USA, Feb. 2003. Springer.

[11] E. Halepovic and R. Deters. The Cost of Using JXTA. In *3rd International Conference on Peer-to-Peer Computing (P2P '03)*, pages 160–167, Linköping, Sweden, Sept. 2003. IEEE Computer Society.

[12] E. Halepovic and R. Deters. JXTA Performance Study. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '03)*, pages 149–154, Victoria, B.C., Canada, Aug. 2003. IEEE Computer Society.

[13] E. Halepovic, R. Deters, and B. Traversat. Performance Evaluation of JXTA Rendezvous. In *International Symposium on Distributed Objects and Applications (DOA '04)*, pages 1125–1142, Agia Napa, Cyprus, Oct. 2004. Springer.

[14] M. Junginger and Y. Lee. The Multi-Ring Topology - High-Performance Group Communication in Peer-to-Peer Networks. In *2nd International Conference on Peer-to-Peer Computing (P2P '02)*, pages 49–56, Linköping, Sweden, Sept. 2002. IEEE Computer Society.

[15] S. Lacour, C. Pérez, and T. Priol. Generic application description model: Toward automatic deployment of applications on computational grids. In *6th IEEE/ACM International Workshop on Grid Computing (Grid2005)*, Seattle, WA, USA, November 2005. Springer.

[16] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In *Proc. 24th Conf. on the IEEE Computer and Communications Societies (INFOCOM 2005)*, pages 225–236. IEEE Computer Society, Mar. 2005.

[17] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *Proc. of the 21th annual symposium on Principles of distributed computing (PODC '02)*, pages 233–242, Monterey, CA, USA, 2002. ACM Press.

[18] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in DHT. Technical Report UCB/CSD-03-1299, Univeristy of California, Computer Science Division (EECS), Berkeley, CA, USA, Dec. 2003.

[19] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–250, Heidelberg, Germany, Nov. 2001. Springer.

[20] J.-M. Seigneur, G. Biegel, and C. D. Jensen. P2P with JXTA-Java pipes. In *2nd international Conference on Principles and Practice of Programming in Java (PPPJ '03)*, pages 207–212, Kilkenny City, Ireland, 2003. Computer Science Press, Inc.

[21] K. Shudo, Y. Tanaka, and S. Sekiguchi. P3: P2P-based Middleware Enabling Transfer and Aggregation of Computational Resources. In *Proc. Workshop on Global and Peer-to-Peer Computing (GP2PC 2005)*, pages 259–266, Cardiff, UK, May 2005. Held in conjunction with the 5th IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid 2005).

[22] D. Stutzbach and R. Rejaie. Improving lookup performance over a widely-deployed DHT. In *Proc. 25th Conf. on Computer Communications (INFOCOMM 2006)*, Barcelona, Spain, Apr. 2006. IEEE Computer Society.

[23] D. Talia and P. Trunfio. Toward a Synergy Between P2P and Grids. *IEEE Internet Computing*, 7(4):94–96, 2003.

[24] N. Théodoloz. DHT-based Routing and Discovery in JXTA. Master's thesis, School of Computer and Communication Sciences, Feb. 2004.

[25] B. Traversat, M. Abdelaziz, D. Doolin, M. Duigou, J.-C. Hugly, and E. Pouyoul. Project JXTA-C: Enabling a Web of Things. In *36th Annual Hawaii International Conference on System Sciences (HICSS '03)*, page 282b, Big Island, Hawaii, Jan. 2003. IEEE Computer Society.

[26] B. Traversat, M. Abdelaziz, and E. Pouyoul. Project JXTA: A Loosely-Consistent DHT Rendezvous Walker. `http://www.jxta.org/docs/jxta-dht.pdf`, Mar. 2003.

[27] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.-C. Hugly, E. Pouyoul, and B. Yeager. Project JXTA 2.0 Super-Peer Virtual Network. `http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf`, May 2003.

[28] Bamboo. `http://bamboo-dht.org/`.

[29] FreePastry. `http://freepastry.rice.edu/`.

[30] JXTA Company Spotlight. `http://www.jxta.org/companies/companyarchive.html`.

[31] JXTA University Spotlight. `http://www.jxta.org/universities/universityarchive.html`.

[32] JXTA specification project. `http://spec.jxta.org/`.