Harnessing the power of DHTs to build dynamic quorums in large-scale enterprise infrastructures *

Roberto Baldoni
Dipartimento di Informatica e Sistemistica, Sapienza Universitá di Roma
Via Ariosto 25, Rome, Italy
baldoni@dis.uniroma1.it

Ricardo Jiménez-Peris
Universidad Politécnica de Madrid (UPM), Campus de Montegancedo
Boadilla del Monte, Madrid, Spain
rjimenez@fi.upm.es

Marta Patiño-Martínez
Universidad Politécnica de Madrid (UPM), Campus de Montegancedo
Boadilla del Monte, Madrid, Spain
mpatino@fi.upm.es

Leonardo Querzoni
Dipartimento di Informatica e Sistemistica, Sapienza Universitá di Roma
Via Ariosto 25, Rome, Italy
querzoni@dis.uniroma1.it

Antonino Virigllito
Dipartimento di Informatica e Sistemistica, Sapienza Universitá di Roma
Via Ariosto 25, Rome, Italy
virgi@dis.uniroma1.it

ABSTRACT

Recently, enterprises owning a large IT hardware and software infrastructure have started looking at Peer-to-peer technologies as a mean both to reduce costs and to help their technical divisions to manage huge number of devices characterized by a high level of cooperation and a relatively low churn. Obtaining the complete and exclusive control of the system for maintenance or auditing purposes in these enterprise infrastructures is a fundamental operation to be implemented. In the context of classical distributed applications, quorum systems have been considered as a major building block for implementing many paradigms, from distributed mutual exclusion to data replication management. In this paper, we explore how to architect decentralized protocols implementing quorum systems in Distributed Hash Table based cooperative P2P networks. This paper introduces some design principles for both quorum systems and protocols using them that boost their scalability and performance. These design principles consist of a dynamic and decentralized selection of quorums and in the exposure and exploitation of internals of the DHT. As a third design principle it is also shown how to redesign quorum systems to enable efficient decentralization.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

Keywords

Quorum systems, distributed hash table, overlay networks

1. INTRODUCTION

Modern internet-scale applications have led to a surge in research for large-scale distributed systems able to tolerate the rate of failures and the inherent dynamics typical of such scenarios. Peer-to-peer (P2P) systems are considered to be an ideal substrate for such applications thanks to their self-organizing nature and their ability to evenly distribute the application load while leveraging the resources offered by each participant.

Recently it has been shown that P2P technologies can be successfully adopted in large-scale IT enterprise infrastructures to reduce the complexity of their management (e.g., [13], [5], [2]) and thus the cost of ownership. The enterprise setting is dramatically different from the inter-domain one: even though the number of peers can be considered of the same order of magnitude[1], peers belonging to the substrate of an enterprise infrastructure work in a managed environment and, thus, they are much more stable than those in internet-based applications. We can therefore expect these peers to join and leave the application gracefully, except in the advent of failures.

However, operations that have to be executed on the P2P substrate by enterprise applications are definitely more complex than those implemented by Internet-based applications (e.g., prioritize and processing SLA alerts based on business impact vs. file location in file-sharing application). These complex operations may require a peer, for example, to take the control of the entire system for auditing, monitoring or

*The work presented in this paper is an extended abstract version of [1] and it was partially supported by the EU Network of Excellence ReSIST (026764), EU Project CoMiFin (225407) and EU project SM4All (224332).

[1]Peers in a large scale enterprise infrastructure include any kind of devices and applications potentially reaching thus a number in the order of hundreds of thousands or millions [14].

maintenance purposes [3]. As remarked in [13], such deep differences between enterprise and Internet-scale settings in terms of system model and operations lead to the need of designing P2P technologies (e.g., algorithms and mechanisms) optimized for a specific environment while keeping generic the very basic P2P functionalities such as the capability of peers to self-organize into a connected overlay network and collectively route data.

In this paper, we focus on an important abstraction for building decentralized protocols, namely quorum systems. Quorum systems [9] are an important building block for the previously described innovative enterprise applications that need to execute complex operations based on distributed consistency of replicated data or on distributed mutual exclusion.

A quorum system over a set of sites (*system universe*) consists of a set of mutually intersecting subsets of sites (quorums). A large variety of quorum systems have been proposed in the literature (see [6] for a survey). In traditional distributed systems, where a site corresponds to a single host, when a quorum is requested on a site, it independently chooses a single quorum among those in the system, and then proceeds trying to obtain permission (the locks) from the corresponding sites. Therefore, all the quorum requesters must precisely know the membership of the system. This makes this strategy hardly adaptable to the P2P context, where the large size of the system makes unrealistic the assumption of global membership knowledge.

This paper presents a set of design principles for the conception, implementation and deployment of quorum systems in DHT-based enterprise infrastructures. A DHT can greatly help the realization of quorums in a large-scale setting: quorums can be defined over the key space, considering each key as a site, without any global knowledge on the number and identity of connected peers. However, simply reproducing the traditional quorum acquisition approach over the DHT does not lead to an efficient solution: if the quorum acquisition process is completely unaware of the key-to-peer mapping, routing can incur in a high overhead.

We propose three design principles for the implementation of efficient quorum systems over P2P architectures, namely *delegation*, *integration* and *flexibility*, and discuss their impact on quorum acquisition costs in terms of both application level messages and quorum formation latency.

These design principles are exercised in two quorum systems. The first one, named *farsighted*, is a novel quorum system that extends hierarchical quorum consensus [7] by offering higher flexibility in choosing quorums with the additional advantage of enabling smaller quorums. The second quorum system is the hierarchical grid quorum [7] incorporating two of the three design principles presented.

The paper is structured as follows: following this introduction, Section 2 will provide the reader with some background about DHTs and quorum systems; Section 3 introduces our general framework together with the three mentioned design principles and its specific implementations for the two considered quorum systems; Section 4 provides a brief insight about the performance of these implementations and, finally, Section 5 concludes the paper.

2. BACKGROUND

Quorum systems. A quorum system over a set of n sites, N, is defined as a set of subsets of sites, or quorums, with pair-wise non-empty intersection. More formally, a set system universe $S = \{S_1, S_2, ..S_n\}$ is a collection of subsets $S_i \subseteq N$. A *quorum system* defined over a set of sites N is a set system S that has the following *intersection property*: $\forall i, j \in \{1..n\}, S_i \cap S_j \neq \emptyset$. In a quorum system each subset S_i is called a *quorum*.

Many quorum systems have been proposed in the literature, but in this paper we consider two well-known systems, namely *farsighted* and *hierarchical grid*. The Farsighted quorum system [1] is a generalization of hierarchical majority [7]. In this quorum system sites are organized in a hierarchy represented as a complete tree with sites at its leaves. A quorum is obtained locking recursively a majority of children sites at each level starting from the root. More than one level at a time can be considered for site selection, thus increasing the selection flexibility of this approach.

Hierarchical grid (HGrid) [8] is a variant of the grid quorum [4] where sites are organized into a multi-level hierarchy, such that they reside on the leaves of this hierarchy, while other levels are represented by logical nodes. Each node at level i of the hierarchy (beside leaves) is defined by a rectangular $m \times n$ grid of nodes at level $i + 1$. A quorum consists of the union of a *full row* and a *row cover* obtained recursively on the hierarchy. A *full row* at level i is defined as the set of $(i + 1)$-level nodes all pertaining to a single row of the grid, while a *row cover* consists in a set of $(i+1)$-level nodes where each node pertains to a different row of the grid. For the sake of simplicity in this paper we only consider HGrid quorums with 2×2 grids at each level of the hierarchy.

P2P distributed hash tables. P2P Distributed Hash Tables (DHTs) are overlay networks based on the idea that messages, instead of being routed directly using physical peers' addresses ranging over a peer space N, can be routed using logical *key* identifiers, defined over a key space K. In *ring-based* DHTs, like Chord [12] or Pastry [11], the key space is a unidimensional circular space, while in *range-based* DHTs like CAN [10] the key space is an n-dimensional space.

DHT protocols are usually based on a hash function that maps keys from the K to actual nodes in N. This function assigns each node and key a m-bit identifier. Each key k is then assigned through a deterministic strategy to one of the peers in the systems (e.g. the one with the closest identifier, or the one whose identifier immediately follows the key's identifier); each peer is responsible of the interval of keys assigned to it by this strategy. The system automatically maintains consistent key mappings in case of nodes joining and leaving the system and provides primitives to route messages targeted to key identifiers. Efficient message routing is obtained by leveraging local data structures, often called *finger tables*, that contain information about nodes whose identifiers are at exponential distance from the local one.

```
Function name: GetQuorum
Input: An interval of keys interval
Output: true or false

List ← ChooseStrategy(interval)
for each i ∈ List
        if ¬ Acquire(i)
                return false
return true
```

```
Function name: Acquire
Input: An interval of keys interval
Output: true or false

send GETQUORUM[interval] to FirstKeyOf(interval)
wait for message from FirstKeyOf(interval)
if message = ACK[interval]
        return true
else
        return false
```

```
Function name: handler for GETQUORUM messages
Input: An interval of keys interval and a key k that generated the message

if interval ⊆ MyKeyspace
        if ¬ IsLocked(interval)
                Lock(interval)
                send ACK[interval] to k
        else
                send NACK[interval] to k
else if GetQuorum(interval)
        send ACK[interval] to k
else
        send NACK[interval] to k
```

Figure 1: A general algorithm for the implementation of quorum systems on DHT-based P2P networks

Thanks to these tables message routing can be realized in a number of application-level hops that is logarithmic with respect to the system population.

3. QUORUM SYSTEMS FOR DHTS

In this section we analyze the issues related to the implementation of different quorum systems on top of a DHT. A DHT provides upper-level applications with a virtual key space that is completely decoupled from the actual set of nodes implementing it. Intuitively, this virtual space can be used to build a quorum system, by considering each key in the DHT as one of the sites constituting the quorums. Therefore, differently from classical distributed systems, distinct sites can be mapped by the DHT to a same peer. We first introduce a general algorithm whose sole purpose is to represent a common implementation framework for all the types of quorum system we will consider. Then, we propose some general strategies that can be exploited for their implementation. Finally, we will show how the generic algorithm can be instantiated to implement the quorum systems briefly introduced in Section 2, and how this can be done exploiting the cited strategies. To simplify the presentation of the technical details, from now on we will refer to Chord [12] as the reference DHT. Nevertheless, all the principles and techniques introduced in this paper could be simply adapted to other DHT-based P2P systems.

3.1 A General Algorithm

The algorithm in Figure 1 is a generic canvas for the acquisition of quorums over DHT-based networks[2]. It embeds several functions, namely *GetQuorum*, *Acquire*, and *ChooseStrategy*, as well as the handler for messages exchanged between nodes. The canvas also embeds other functions whose meaning is intuitive. A peer willing to gain complete control of the system would start the acquisition of a quorum by invoking the *GetQuorum* function and passing it, as a parameter, the whole key space.

Key to this canvas is the *ChooseStrategy* function which implements the logic related to a specific quorum system (i.e., hierarchical grid, hierarchical majority, farsighted, etc.). In other words, only the *ChooseStrategy* must be changed in order to implement a different quorum system.

The proposed general algorithm enables a large variety of implementation strategies. As a trivial solution, the *ChooseStrategy* function can return subintervals composed by a single key. This corresponds to a *centralized* strategy, where the requester directly contacts all the keys forming the quorum. The centralized approach has an evident drawback: as each single peer is unaware of the mapping between keys and peers, it will send a different message to each single key even if most of such messages will be routed to the same peer. This inefficient behavior has a strong negative impact on the number of messages generated to acquire each single quorum.

Through the general algorithm we can achieve a completely decentralized approach in the quorum formation. If the *ChooseStrategy* implementation returns a set of subintervals, each subinterval will lead to a recursive call to *getQuorum*, until the subinterval is a subset of the key space of a peer. In other words we introduce a mechanism of *delegation* to delegate different parts of the quorum acquisition to different peers. The advantage of this technique is that large intervals are progressively split into smaller subintervals, that are more likely to be completely contained in a peer's own interval and that can be acquired with a single message.

The selection of key intervals in *ChooseStrategy* can be done considering preferred intervals for delegation as those that will be delegated to peers contained in the finger table of the

[2]Note that the proposed protocol, in general, will not deal with failures: a failure of a peer could leave a subset of keys locked, and thus reduce the overall availability of the quorum system. Various techniques can be adopted to recover from this state; we strongly suggest the reader to refer to [1] for further details on this aspect.

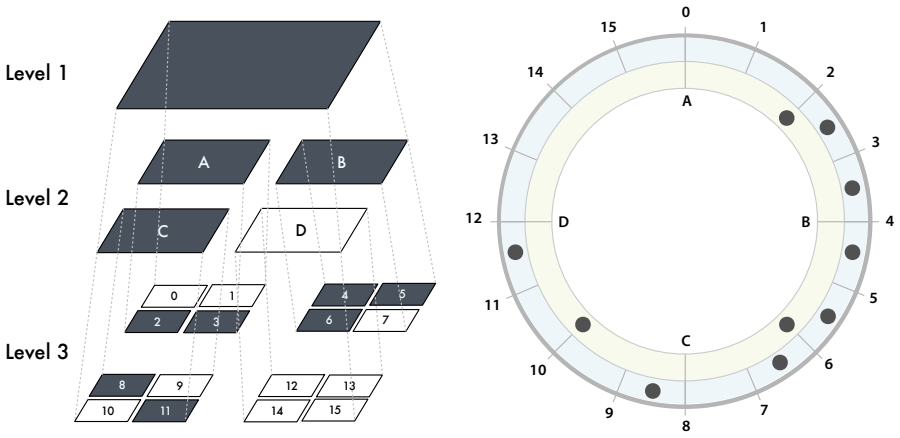

Figure 2: Example of a hierarchical grid quorum built over a 16-key Chord ring.

current peer. We refer to this new implementation strategy as a *integration* mechanism. The main goal of this mechanism is to select targets of the delegation such that, the routing mechanism of the DHT will be forced to make almost only single-hop routing steps, thus greatly reducing routing overhead. Implementations exploiting this mechanism will be referred to as *integrated* protocols, in contrast with protocols that are completely oblivious of the underlying DHT routing mechanism, namely *layered* protocols.

3.2 Hierarchical grid

To implement HGrid on top of a ring-based DHT we need to find out a functional way to map the hierarchical bi-dimensional grid structure on top of the uni-dimensional keyspace. The keyspace is first partitioned into four intervals that are mapped to the four cells of a 2×2 grid, such that keys from 0 to $2^{m-2}-1$ are mapped to cell $(1,1)$ (row 1 and column 1) of the grid, keys from 2^{m-2} to $2 \cdot 2^{m-2}-1$ are mapped to cell $(1,2)$, keys from $2 \cdot 2^{m-2}$ to $3 \cdot 2^{m-2}-1$ are mapped to cell $(2,1)$, and keys from $3 \cdot 2^{m-2}$ to $4 \cdot 2^{m-2}-1 = 2^m-1$ are mapped to cell $(2,2)$; this grid represents the first level of the hierarchy. Grids in lower levels of the hierarchy are obtained applying the same subdivision to the corresponding subintervals of the keyspace, until we reach a level where each single key is mapped to a different cell of the grid.

This mapping is shared among all peers in order to guarantee that the intersection property holds for any quorum chosen starting from any key. Figure 2 gives an example on how a three-level HGrid with 16 cells at the lower level can be mapped onto a 16-key Chord ring;

A centralized algorithm that builds HGrid quorums can be instantiated on the generic algorithm of Section 3.1 by implementing a *ChooseStrategy* function that acts locally by choosing recursively on the hierarchy a full row and a row cover, and then returns the corresponding keys. In the following we denote such an algorithm as *centralized HGrid*.

The same algorithm can be simply transformed in the corresponding decentralized version. The requesting peer starts two parallel actions to obtain a full row and a row cover on the root level grid. Each of these actions returns the key intervals corresponding to the selected cells. Choices at lower levels of the grid hierarchy are then delegated to those peers that are responsible for the first key of the chosen intervals. In the following we denote such an algorithm as *decentralized HGrid*.

The Integrated version of the hierarchical grid quorum system (*integrated HGrid*) algorithm is an enhanced version of HGrid that exploits the knowledge contained at the DHT routing layer (i.e., finger table and the interval of keys the peer is responsible for). More specifically, *ChooseStrategy* of HGrid tries to select those cells in the grids that contain at least one finger. Only fingers pointing to peers that, given their position in the keyspace, could be able to obtain the desired keys are considered by the algorithm. When no finger is available to obtain all the required keys in a grid, then the algorithm proceeds with random choices.

Figure 2 shows an example of a hierarchical grid quorum obtained on a 16-cell grid structured as a three level hierarchy. Grey cells on the left side indicate those cells that have been selected by the algorithm, and dotted keys on the right side the corresponding locked keys in the Chord ring.

3.3 Hierarchical majority and Farsighted

The hierarchical majority quorum system (HMaj) consists in a hierarchical organization of the sites into a tree of degree d, in which sites (keys) are the leaves. Although the optimal quorum size for HMaj is obtained with a tree of degree 3, we will use $d = 4$ to simplify the matching between key intervals defined by the tree and the key space size, $2^{2 \cdot x} = 2^m$.

In HMaj a quorum is formed recursively by selecting a majority of children at the first level and then selecting recursively a majority in each of the selected children until the leaves are reached. Let us point out that a child at a certain level of the hierarchy is directly mapped to a subinterval of the key space. Therefore, when a majority of children are chosen, it is actually being decided which underlying intervals of the key space are being selected as shown in Figure 3 for a 16-key Chord ring. In this figure black dots represent a HMaj quorum.

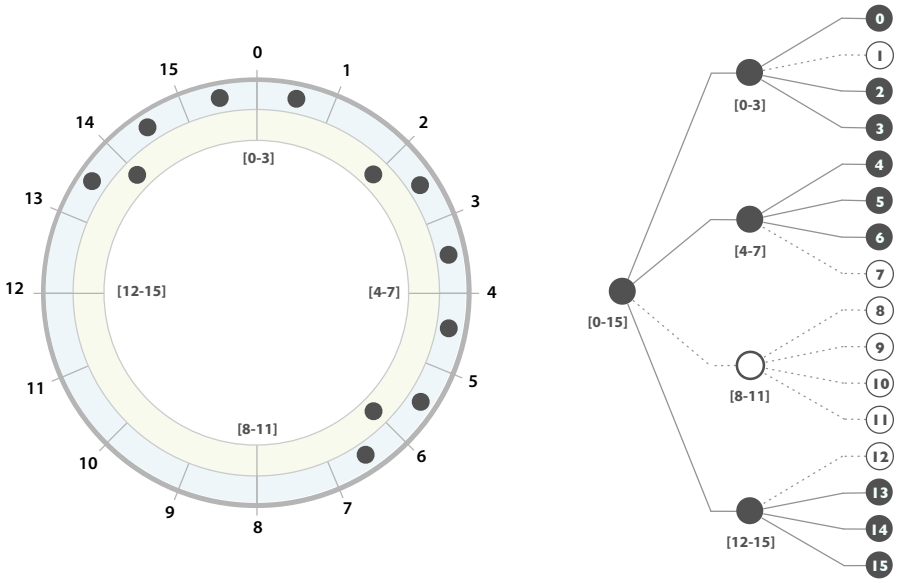

Figure 3: A decentralized hierarchical majority quorum built over 16-key Chord ring

A centralized approach for the implementation of HMaj would consist in selecting one quorum at the requester peer and obtaining a lock for each site in the selected quorum. The *ChooseStrategy* function that implements this approach is straightforward: it simply recursively walks down the hierarchy choosing each time three children out of four until the leaves level is reached. Each site (key) selected at this final stage is returned by the function as a different interval. We denote such algorithm as *centralized HMaj*.

HMaj quorums can be also implemented by computing quorum sites in a decentralized manner. In the following we refer to such algorithm as *decentralized HMaj*. As in centralized HMaj, the requester in decentralized HMaj selects three children (key subintervals) out of four, but only for a single level of the hierarchy. Then, a message with each selected interval is sent to the first key of that interval. The peer responsible for that key will walk down one more step in the hierarchy. The decentralized recursion stops when a peer receives a message containing an interval which is completely contained in the portion of the key space the peer is responsible for. Therefore, the implementation of *ChooseStrategy* for decentralized HMaj boils down to the selection from the given interval of three randomly chosen subintervals out of four.

For example, let us assume that key 0 is the requester key (Figure 3). The first iteration of the algorithm occurs in *ChooseStrategy* at the peer responsible for key 0 and works on interval $[0-15]$. *ChooseStrategy* subdivides the interval into four subintervals and chooses three of them: $[0-3]$, $[4-7]$ and $[12-15]$. The next iteration occurs in parallel on the peers responsible for the first key of each subinterval.

Farsighted is a novel quorum system [1] that has been designed for efficient decentralized and integrated quorum acquisition in enterprise P2P networks. It can be seen as a generalization of hierarchical majority. Farsighted is based on the same logical tree as hierarchical majority, but instead of considering a single tree level in the recursive quorum formation it looks at more than one level (hence its name) in order to form a quorum. In its basic form, it looks at two consecutive levels of the tree. By looking at two consecutive levels, a much richer set of alternatives to form quorums become possible. Let us consider a quaternary tree to simplify the exposition. Hierarchical majority allows choosing 3 children from the first level and then 3 children out of these 3. This would be represented by all the permutations of the pattern (3,3,3,0) (termed pattern sets). Farsighted allows more flexibility: it allows, for instance, also the pattern sets (3,2,2,2) and (3,3,2,0) or (4,1,1,1) and (4,3,2,0). Interestingly, some of the allowed pattern sets result in quorum systems with lower load than hierarchical majority (e.g. (4,1,1,1)).

The mutual intersection condition for a combination of patterns for a tree of degree d to yield a valid quorum system is that for each pair of patterns $(f_1, f_2, \ldots, f_d)$ and $(g_1, g_2, \ldots, g_d)$, $\exists k \in [1..d]$ such that $f_k + g_k > d$. The condition for mutual intersection can be extended to a pattern set (all the permutations of pattern). A pattern set satisfies the mutual intersection property if, once sorted in increasing and decreasing lexicographical order, it guarantees the aforementioned condition. For instance, the pattern set (4,1,1,1) satisfies the intersection condition since (4,1,1,1) and (1,1,1,4) results in 4+1>d=4 in both the first and last positions. The condition can also be extended to different valid pattern sets. Given two valid pattern sets they can be used in combination if sorted in opposite lexicographical orders they satisfy the above condition. For instance, for the pattern sets (4,1,1,1) and (4,3,2,0), sorted in opposite lexicographical orders yield (4,1,1,1) and (0,2,3,4)), and in the fourth position 1+4=5>d=4, so this means that all patterns in these pattern sets satisfy the intersection property. Thanks to the richer set of pattern sets, it becomes possible for farsighted to choose which is the pattern more

appropriate in a decentralized and integrated quorum system formation protocol. This allows selecting patterns that lead to interval delegation toward easily reachable fingers, resulting in reduced load due to DHT routing. More details about farsighted, and how it can be implemented through our general algorithm can be found at [1].

4. EVALUATION

Quorum systems have been traditionally compared in terms of availability, load [9] and the quorum acquisition cost, i.e. the number of messages needed to lock all sites forming a quorum.

While load and availability only depend on combinatorial properties of the quorum structure, acquisition cost is related to the number of messages required to contact all the sites forming a quorum. Quorums built upon a P2P routing mechanism, as the one provided by DHTs, must be evaluated considering also the cost incurred by the routing mechanism to bring every message to the intended destination. Message routing in DHTs exploits local data structures (*finger tables*) at each peer to route messages between any two peers with an upper bound of $O(log(n))$ application-level hops in a system with n peers.

In [1] we report an extensive simulation-based evaluation of both farsighted and hierarchical grid algorithms implemented on a Chord [12] DHT using the three design principles introduced in Section 3: delegation, integration and flexibility. The study compares the performance of the various implementations of the two quorum systems in terms of (i) percentage of peers containing locked keys after a quorum acquisition is finished, (ii) quorum acquisition cost in terms of application-level messages, (iii) depth of the multicast tree generated during the quorum acquisition, (iv) role of peers (i.e. percentage of peers with locked keys, delegating peers or DHT routers), (v) distribution of messages among peers and (vi) quorum availability after a peer failure. The main outcomes of this evaluation are:

- a huge number of messages for the quorum acquisition can be saved just applying the delegation design principle, resulting in better performance for both the quorum systems in terms of quorum acquisition costs. This is a consequence of the fact that algorithms designed with this principle build a multicast tree over the DHT for locking sets of keys during the quorum acquisition instead of contacting each site with a single point-to-point message;

- the adoption of the integration design principle let algorithm implementations use the DHT data structures to take key selection decisions during the quorum formation that greatly help reducing the depth of the multicast tree. The reduction of this metric has a direct impact on the latency experienced during each quorum acquisition;

- the flexibility introduced by the farsighted quorum system with respect to simpler hierarchical majority quorum systems is clearly reflected in its improved performance in all aspects.

5. CONCLUSIONS

P2P technologies are becoming pervasive in new application contexts different from the classical internet-based file sharing, such as service management in data centers, distributed data storage and retrieval or trust-less distributed file systems for large enterprises. These new contexts have dramatically different characteristics and requirements (e.g., long vs. short peer lifetime, managed environments vs. unmanaged ones, cooperative vs. uncooperative peers) and they have to support much more complex operations (e.g., managing contractual SLA data vs. file location). As remarked in the introduction, such deep differences between enterprise and Internet-scale settings lead to the need of designing P2P technologies (e.g., algorithms and mechanisms) optimized for a specific environment while keeping generic the very basic P2P functionalities — such as the capability of peers to self-organize into an overlay network and collectively route data.

In this paper we focussed our study on three design principles, namely integration, delegation and flexibility, that should guide the construction of quorum systems and protocols in DHT-based enterprise infrastructures. These design principles have been exercised in two quorums systems, farsighted and hierarchical grid. For each quorum system three versions have been implemented and evaluated: centralized, decentralized and integrated.

A simulation-based study shown that traditional centralized approaches for acquiring quorums are not scalable in a P2P context due to the huge number of messages generated. Decentralization by itself helps improving considerably the amount of load imposed on the system by a quorum acquisition. Moreover, integration effectively diminishes the number of routing peers needed during the quorum formation. Integration, mixed with quorum flexibility in the farsighted quorum system, is also able to reduce substantially the quorum acquisition latency introduced by decentralization.

6. REFERENCES

[1] R. Baldoni, R. R. Jiménez-Peris, M. Patiño-Martínez, L. Querzoni, and A. Virgillito. Dynamic quorums for dht-based enterprise infrastructures. *Journal of Parallel and Distributed Computing*, 68(9):1235–1249, 9 2008.

[2] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 34–43, New York, NY, USA, 2000. ACM.

[3] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *OSDI*, pages 335–350. USENIX Association, 2006.

[4] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. In *Proceedings of the 6th International Conference on Data Engineering (ICDE)*, pages 438–445. IEEE Computer Society, 1990.

[5] G. De Candia, D. Hastorun, M. Jampani,

G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 205–220, New York, NY, USA, 2007. ACM.

[6] R. Jiménez-Peris, M. Patiño-Martínez, G. Alonso, and B. Kemme. Are quorums an alternative for data replication ? *ACM Transactions on Database Systems*, 28(3):257–294, 2003.

[7] A. Kumar. Hierarchical quorum consensus: A new algorithm for managing replicated data. *IEEE Transactions on Computers*, 40(9):996–1004, 1991.

[8] A. Kumar and S. Y. Cheung. A high availability $\sqrt{N}$ hierarchical grid algorithm for replicated data. *Information Processing Letters*, 40(6):311–316, 1991.

[9] M. Naor and A. Wool. The load, capacity, and availability of quorum systems. *SIAM Journal of Computing*, 27(2):423–447, 1998.

[10] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In J. Crowcroft and M. Hofmann, editors, *Networked Group Communication*, volume 2233 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2001.

[11] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 12-16 November 2001.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 149–160, 2001.

[13] C. Tang, R. N. Chang, and E. So. A distributed service management infrastructure for enterprise data centers based on peer-to-peer technology. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 52–59. IEEE Computer Society, 2006.

[14] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici. A scalable application placement controller for enterprise data centers. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors, *Proceedings of the 16th international conference on World Wide Web (WWW)*, pages 331–340. ACM, 2007.