Probabilistic Quorums for Dynamic Systems

Ittai Abraham [*] Dahlia Malkhi [†]

Abstract

A quorum system is a set of sets such that every two sets in the quorum system intersect. Quorum systems may be used as a building block for performing updates and global queries on a distributed, shared information base. An ε-intersecting quorum system is a distribution on sets such that every two sets from the distribution intersect with probability $1 - \varepsilon$. This relaxation of consistency results in a dramatic improvement of the load balancing and resilience of quorum systems, making the approach especially attractive for scalable and dynamic settings.

In this paper we assume a dynamic model where nodes constantly join and leave the system. A quorum chosen at time s must evolve and transform as the system grows/shrinks in order to remain viable. For such a dynamic model, we introduce dynamic ε-intersecting quorum systems. A dynamic ε-intersecting quorum system ensures that in spite of arbitrary changes in the system population, any two evolved quorums intersect with probability $1 - \varepsilon$.

Contact Author:

Ittai Abraham
`ittaia@cs.huji.ac.il`
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem 91904 ISRAEL
Tl: +972-2-658-5351

Track: Regular presentation.

Student paper: Ittai Abraham is a full-time student.

[*]School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel. This research was supported in part by NDS PhD fellowship program. Email: ittaia@cs.huji.ac.il

[†]School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel. Email: dalia@cs.huji.ac.il

1 Introduction

Consider the following natural information-sharing problem. Participants wish to publicize certain data items, and be able to access the whole information base with search queries. In order to balance the load of updates and queries among the participants, quorums may be used for reading and for writing. Quorums enhance the load balance and availability, and provide flexibility in tuning between read costs and writes costs. However, quorum systems are defined over a universe known to all participants, requiring each process to maintain knowledge of all the other participants.

We aim for a scalable and dynamic information-sharing solution, in which maintaining global knowledge of the system configuration is prohibitive. More concretely, we allow each participant to maintain connections with, and even knowledge of, only a constant number of other members. This restriction stems both from our vision of having ubiquitous, low-memory devices participate in Internet resource sharing applications; and from the desire to keep the amount of state that needs to be updated at reconfiguration very low.

Our focus on scale and dynamism of information-sharing systems is supported by the stellar popularity of recent resource sharing applications like Gnutella. In these systems, users make available certain data items, like music clips or software, and the system supports global querying of the shared information. Gnutella supports global querying through a probabilistic depth-bounded multi-cast. This approach is effective, yet ad hoc.

We devise techniques for the information sharing problem, capable of dealing with high decentralization and dynamism exhibited in Internet-scale applications. Our approach is based on the *probabilistic quorum systems* (PQSs) of Malkhi et al. in [15]. We extend the treatment of PQSs to cope with scalability and high dynamism in the following ways. First, we allow each participant only partial knowledge of the full system, and avoid maintaining any global information of the system size and its constituents. To this end, we develop a theory of PQSs whose individual member selection probability is non-uniform. We demonstrate a realization of such a non-uniform PQS that is fully adapted for the dynamic and scalable settings we aim for. The second extension of PQSs we address is to evolve quorums as the system grows/shrinks in order for them to remain viable. We provide both a formal definition of quorum evolution and the algorithms to realize it.

We first remind the reader of PQSs and motivate their use. The PQSs of Malkhi et al. [15] are an attractive approach for sharing information in a large network. Using a PQS, each participant can disseminate new updates to shared data by contacting a subset (a probabilistic quorum) of $k\sqrt{n}$ processes chosen uniformly at random, where n is the size of the system and k is a reliability parameter. Likewise, participants query data from such quorums. Intuitively, analysis similar to the famous "birthday paradox" (e.g., see [4]) shows that each pair of update/query quorums intersect with probability $1 - e^{-k^2/2}$. The result is that with arbitrarily good probability a query obtains up to date information, and with a small calculated risk it might obtain stale data.

The benefit of the PQS approach is tremendous: Publicizing information and global querying are done each with only a $O(1/\sqrt{n})$ fraction of the participants. The same efficiency may be achieved in strict quorum systems. Independently and simultaneously to our work, Naor and Wieder [17] sought solutions for dynamic deterministic quorum systems. Compared with strict quorum systems, PQSs are natural for dynamic and non-structured environments. Additionally, PQSs maintain availability in face of as many as $O(n)$ faults. In deterministic approaches these two features are provably impossible to achieve simultaneously (see [18]). Indeed, PQSs have been employed in diverse and numerous settings. To name just a few deployments, PQSs were used for designing probabilistic distributed emulations of various shared objects [11, 12]; they were used for constructing persistent shared objects in the Phalanx and Fleet systems [13, 14]; and they were employed for maintaining tracking data in mobile ad-hoc networks [8].

Now we get to why we need new PQSs. The prevalent PQS construction [15] is not adequate precisely for the kind of scalable and dynamic settings for which PQSs are most beneficial. First, it requires global and precise knowledge of the system size 'n'. Second, it hinges on the ability of each participant to select other processes uniformly at random. And third, it does not indicate what should happen to a quorum that stores data at time s, as the system grows/shrinks.

Our goal is to maintain a PQS that copes with dynamism so that every pair of quorums intersect with the desired probability $1 - \varepsilon$ despite any number of system reconfigurations. To this end, we introduce the notion of *dynamic PQSs*. This notion captures the need for quorums to evolve as the system changes, and requires pairs of evolved quorums to intersect with probability $1 - \varepsilon$. We present a dynamic PQS construction that works without maintaining any global knowledge of the system size or its participants. Our approach incurs a reasonable price per system reconfiguration.

Technical approach: We first consider the problem of establishing probabilistic quorums in a setting in which full knowledge of the system by each member is not desirable. After that, we address quorum evolution.

Consider a system in which each process is linked to only a small number of other processes. Clearly, a quorum establishment operation initiated at some process must somehow walk the links in the system in search of unknown members. There are indeed some recent dynamic networks that support random walks. The works of [19, 5, 21] assume a-priori known bound on the network size, whereas we do not place any bound on network growth. A different approach is taken in the AntWalk system [20], that necessitates periodic, global "re-mixing" of the links of old members with those of the new processes that arrived. We consider the price of this approach too heavy for Internet wide applications.

The first step in our solution to avoid the globalization pitfall is to introduce a *non-uniform PQS* as follows. Let us have any probability distribution $p : S \to [0..1]$ over individual member selection. We define the *flat access strategy* $f(p, m)$ as the quorum selection distribution obtained by selecting m members according to p. We show that a quorum system with the access strategy $f(p, k\sqrt{n})$ is an $e^{-k^2/2}$-intersecting PQS.

It is left to show how to realize a process selection distribution p in dynamic settings and how to preserve it in evolving quorums. Our approach makes use of recent advances in overlay networks for peer-to-peer applications. Specifically, our design employs a dynamic routing graph based on the dynamic approximation of the de Bruijn network introduced in [1] [1]. This dynamic graph has w.h.p. a constant-degree, logarithmic routing complexity, and logarithmic mixing time. Thus, random walks are rapidly mixing to a stationary distribution p. This means that using a logarithmic number of messages, each process can find other participants with selection probability p.

The dynamic graph allows to estimate the size of the network n with a constant error factor. Using this estimation, the flat access strategy $f(p, k\sqrt{n})$ is approximated by performing roughly $k\sqrt{n}$ random walks of $\log(n)$ steps each. We obtain at any instant in time an $e^{-k^2/2}$-intersecting PQS. Accessing quorums is done in $\log(n)$ parallel time.

Finally, we need to describe how to evolve quorums as the system grows. We devise an evolution strategy that grows the quorums along with the system's growth automatically and distributively. We prove that our evolution technique keeps quorums sufficiently large, as well as maintains the

[1] The dynamic De Bruijn construction appeared independently also in [16, 6, 10].

individual member selection distribution p. The cost of our maintenance algorithm is w.h.p. a constant number of random walks per system reconfiguration. Each single walk incurs a logarithmic number of messages and a state change in one process.

To summarize, the results of our construction is a scalable information sharing mechanism based on dynamic PQSs. The construction maintains $1-\varepsilon$ intersection probability in any dynamic setting, without central coordination or global knowledge. The system achieves the following performance measures with high probability: The cost of a member addition (join) is a logarithmic number of messages, and a state-change in a constant number of members. Further, the advantage of using PQSs is stressed in the time required for accessing a quorum. Because $O(\sqrt{n})$ processes are chosen independently at random, quorum selection is done in $O(\log n)$ parallel time. Regardless of system growth, the load incurred by information maintenance and update processing on individual members is balanced up to a constant factor.

2 Problem Definition

We consider a (potentially infinite) universe W of possible processes. The system consists of a dynamic subset of processes taken from W that evolves over time as processes join and leave the system. For purposes of reasoning about the system, we use a logical discrete time-scale $T = \{0, 1, \dots\}$. At each time-step $i \in T$ there exists a set $U(i)$ of processes from W that are considered members of the system at that time. Each time-step i consists of a single event $e(i)$, which is one of the following: Either a process joins the system, or a process leaves the system. For each time step $t > 0$, the partial history of events uniquely determines the universe $U = U(t)$ consisting of all the processes that joined the system minus those that have left.

Focusing on a fixed time step $t > 0$ for now, we first recall the relevant definitions from [15]. A *set system* $\mathcal{Q}$ over a universe U is a set of subsets of U. A *(strict) quorum system* $\mathcal{Q}$ over a universe U is a set system over U such that for every $Q, Q' \in \mathcal{Q}$, $Q \cap Q' \neq \emptyset$. Each $Q \in \mathcal{Q}$ is called a *quorum*. An *access strategy* ac for a set system $\mathcal{Q}$ specifies a probability distribution on the elements of $\mathcal{Q}$. That is, $ac : \mathcal{Q} \to [0, 1]$ satisfies $\sum_{Q \in \mathcal{Q}} ac(Q) = 1$. We are now ready to state the definition of probabilistic quorum systems:

Definition 2.1 (ϵ-intersecting quorum system[15]) *Let $\mathcal{Q}$ be a set system, let ac be an access strategy for $\mathcal{Q}$, and let $0 < \epsilon < 1$ be given. The tuple $\langle \mathcal{Q}, ac \rangle$ is an* ϵ-intersecting quorum system *if $\Pr[Q \cap Q' \neq \emptyset] \geq 1 - \epsilon$, where the probability is taken with respect to the strategy ac.*

We now proceed to define time-evolving quorums. We first define an evolution strategy as follows:

Definition 2.2 (Evolution strategy) *For every $t \in T$, let $\mathcal{Q}(t)$ be a set system over the system $U(t)$. An* evolution strategy *ev_t specifies a probability distribution on the elements of $\mathcal{Q}(t)$ for each given element of $\mathcal{Q}(t-1)$. Formally, $ev_t : \mathcal{Q}(t-1) \times \mathcal{Q}(t) \to [0, 1]$ satisfies*

$$\forall Q' \in \mathcal{Q}(t-1) : \sum_{Q \in \mathcal{Q}(t)} ev_t(Q', Q) = 1 \ .$$

Thus, $ev_t(Q', Q)$ for $Q' \in \mathcal{Q}(t-1)$ and $Q \in \mathcal{Q}(t)$ indicates the probability that Q' evolves into Q.

The access strategies over $U(1), U(2), \ldots$ together with an evolution strategy determine the probability that a certain subset occurs as the evolution of any previously created quorum. The following definition captures this distribution:

Definition 2.3 (Evolving probability distribution) *For every time step $i \in T$, let $\langle \mathcal{Q}(i), ac_i \rangle$ be a probabilistic quorum system and ev_i be an evolution strategy. The* evolving probability distribution *$p_t^s : \mathcal{Q}(t) \to [0,1]$ for quorums created at time s that evolved up to time t, for $t \geq s$, is defined recursively as follows:*

$$\forall Q \in \mathcal{Q}(t) : p_t^s(Q) = \begin{cases} ac_s(Q) & t = s, \\ \displaystyle\sum_{Q' \in \mathcal{Q}(t-1)} p_{(t-1)}^s(Q') ev_t(Q', Q) & t > s. \end{cases} \tag{1}$$

Our goal is to devise a mechanism for maintaining ε-intersecting probabilistic quorums in each $U(t)$, and to evolve quorums that maintain information (such as updates to data) so that their evolution remains ε-intersecting with quorums in later time steps. Any two quorums created at times s and t will evolve in a manner that at any later time r, their intersection probability remains $1 - \varepsilon$. This is captured in the following definition:

Definition 2.4 (Dynamic ϵ-intersecting probabilistic quorum system) *For every time step $i > 0$, let $\langle \mathcal{Q}(i), ac_i \rangle$ be a probabilistic quorum system and ev_i be an evolution strategy. Let $0 < \epsilon < 1$ be given. Then $\langle \mathcal{Q}(i), ac_i, ev_i \rangle$ is a* dynamic ϵ-intersecting quorum system *if for all $r \geq s \geq t > 0, Q, Q' \in \mathcal{Q}(r)$:*

$$\Pr[Q \cap Q' \neq \emptyset] \ \geq \ 1 - \epsilon$$

where the probability is taken over the choices of Q and Q', distributed respectively according to $Q \sim p_r^s$ and $Q' \sim p_r^t$.

2.1 Performance measures

Driven by our goal to maintain quorums in very large and dynamic environments, such as Internet-wide peer-to-peer applications, we identify the following four performance measures. First, we have the *complexity of handling join/leave* events, measured in terms of messages and the number of processes that must incur a state-change. Second, we consider the *complexity of accessing a quorum*, measured both in messages and in (parallel) time. These two measures reflect the complexity incurred by linking processes in the system to each other and of the searching over the links. Our goal is to keep the join/leave message complexity logarithmic, and the number of state-changes constant per reconfiguration. We strive to maintain a logarithmic quorum access time.

Additionally, we consider two traditional measures that were defined to assess the quality of probabilistic quorum systems [18, 15]: The *load* inflicted on processes is the fraction of total updates/queries they must receive. The degree of *resilience* is the amount of failures tolerable by the service. In Appendix B we recall the precise definition of these measures. Our goals with respect to the latter two measures are to preserve the good performance of PQSs in static settings. Specifically, we wish for the load to be $O(1/\sqrt{n})$ and the resilience to be $O(n)$.

3 Non-uniform Probabilistic Quorum Systems

In this section, we extend the treatment of probabilistic quorum systems of [15] to constructions that employ non-uniform member selection.

Let S be a system containing n members (e.g., $S = U(t)$ for some $t > 0$). Let $p(s)$ be any distribution over the members $s \in S$. We first define a flat non-uniform selection strategy that chooses members according to p until a certain count is reached.

DEFINITION 3.1 (Flat access strategy) *The* flat access strategy $f(p,m) : \mathcal{Q}_m \rightarrow [0,1]$ *as follows: for* $Q \in \mathcal{Q}_m$, $f(p,m)(Q)$ *equals the probability of obtaining the set* Q *by repeatedly choosing* m *times (with repetitions) from the universe* S *using the distribution* p.

The flat strategy $f(p,m)$ strictly generalizes the known access strategy for PQSs in which m members are chosen uniformly at random (simply put $p(s) = 1/n$ for all $s \in S$). In the Lemma below, we obtain a generalized probabilistic quorum system with non-uniform member selection.

Lemma 3.1 *The construction* $\langle 2^S, f(p, k\sqrt{n})\rangle$ *is an* $(e^{-k^2/2})$*-intersecting quorum system.*

Proof: Consider two sets $Q, Q' \sim f(p, k\sqrt{n})$. For every $s \in S$ denote an indicator variable x_s that equals 1 if $s \in Q \cap Q'$, and equals 0 otherwise. Thus, $E[\sum_{s\in S} x_s] = k^2 np^2(s)$. Now consider a uniformly distributed random variable p that has value $p(s)$ for all $s \in S$. Since for any random variable it holds that $E(p^2) \geq E(p)^2$ we have $\sum_{s\in S} p^2(s)\frac{1}{n} \geq \left(\sum_{s\in S} p(s)\frac{1}{n}\right)^2$. Combining the above: $E(\sum_{s\in S} x_s) = k^2 n \sum_{s\in S} p^2(s) \geq k^2$. Using Chernoff bounds we have proven:$Pr[Q \cap Q' = \emptyset] = Pr[\sum_{s\in S} x_s = 0] < e^{-k^2/2}$. □

Interestingly, the flat access strategy is overly conservative in the following sense. Generally, a quorum selection strategy with non-uniform member selection distribution need not necessarily have a fixed quorum size. Intuitively, this is because "heavier" members (that are chosen with a higher probability) are more likely to occur in the intersection among pairs of quorums. An example might clarify this point: Suppose that some member $s \in S$ has $p(s) = 1/2$. Clearly, if s belongs to a quorum, then the probability of intersecting with any other quorum is at least a half, even if quorums have only one element each. In the general case, the total number of selected members could therefore depend on their combined weight. We encountered a difficulty in obtaining such a "weighted" access strategy, namely that the likelihood that a member is included in a quorum depends on the ordering of sampling. We are currently still investigating whether there is a way to implement a non-uniform variable-size quorum access strategy along these lines.

Finally, note that implementing $f(p, k\sqrt{n})$ requires global knowledge of n, which is difficult in a dynamic setting. The remaining of this paper is devoted to approximating f, i.e., we show how to (roughly) maintain a non-uniform flat quorum access strategy and how to evolve quorums, over a dynamic system.

4 Non-uniform Probabilistic Quorums in Dynamic Systems

4.1 The dynamic graph

A key component in the construction is a dynamic routing graph among the processes. The graph allows processes to search for other processes during quorum selection. Denote $G(t) = \langle V(t), E(t)\rangle$

a directed graph representing the system at time t as follows. $V(t)$ is the set of processes $U(t)$ at time point $t > 0$. There is a directed edge $(u, v) \in E(t)$ if u knows v and can communicate directly with it. Henceforth, we refer to system participants as processes or as nodes interchangeably.

Driven by the need to maintain the goals stated above in Section 2.1, we wish to maintain a dynamic graph $G(t)$ with the following properties: (1) Small constant degree (so as to maintain constant join/leave complexity). (2) Logarithmic routing complexity (so as to maintain a reasonable quorum selection cost). (3) Rapid mixing time, so that we can maintain a fixed individual selection distribution using a small number of steps from each node.

We choose to employ for $G(t)$ a routing graph that approximates a de Bruijn routing graph. In the de Bruijn [3], a node $\langle a_1, \ldots, a_k \rangle$ has an edge to the two nodes $\langle a_2, \ldots, a_k, 0/1 \rangle$ (shift, then set the last bit). We employ a dynamic approximation of the De Bruijn graph that was introduced in [1]. This dynamic graph has w.h.p. a constant-degree, logarithmic routing complexity, and logarithmic mixing time.

The dynamic graph is constructed dynamically as follows. We assume that initially, G_1 has two members that bootstrap the system, whose id's are 0 and 1. The graph linking is a *dynamic de Bruijn* linking, defined as follows:

DEFINITION 4.1 (Dynamic de Bruijn linking) *We say that a graph has a* dynamic de Bruijn *linking if each node whose id is $\langle a_1, \ldots, a_k \rangle$ has an edge to each node whose id is $\langle a_2, \ldots, a_k \rangle$ or whose id is a prefix thereof, or whose id has any postfix in addition to it.*

Joining and the leaving of members is done as follows:

Join: When a node u joins the system, it chooses some member node v and "splits" it. That is, let $\hat{v} = \langle a_1, \ldots, a_k \rangle$ be the identifier v has before the split. Then u uniformly chooses $i \in \{0, 1\}$, obtains identifier $u.id = \langle a_1, \ldots, a_k, i \rangle$ and v changes its identifier to $v.id = \langle a_1, \ldots, a_k, (1 - i) \rangle$. The links to and from v and u are updated so as to maintain the dynamic de Bruijn linking.

Leave: When a node u leaves the system, it finds a pair of 'twin' nodes $\langle a_1, \ldots, a_k, 0 \rangle$, $\langle a_1, \ldots, a_k, 1 \rangle$. If u is not already one of them, it uniformly chooses $i \in \{0, 1\}$, swaps with $\langle a_1, \ldots, a_k, i \rangle$, and $\langle a_1, \ldots, a_k, i \rangle$ leaves.

When a twin $\langle a_1, \ldots, a_k, i \rangle$ leaves, its twin $\langle a_1, \ldots, a_k, (1-i) \rangle$ changes its identifier to $\langle a_1, \ldots, a_k \rangle$. The links to and from $\langle a_1, \ldots, a_k \rangle$ are updated so as to maintain the dynamic de Bruijn linking.

DEFINITION 4.2 (Level) *For a node v with id $\langle a_1, \ldots, a_k \rangle$. Define its* level *as $\ell(v) = k$.*

DEFINITION 4.3 (Global gap) *The* global gap *of a graph $G(t)$ is $\max_{v,u \in V(t)} |\ell(v) - \ell(u)|$.*

For a more general definition of these dynamic graphs, and an analysis of their properties see [1]. Techniques for maintaining w.h.p. constant-bound on the global gap in dynamic graphs such as $G(t)$ are presented in [1] with logarithmic per join/leave cost. In [16] techniques are presented for maintaining a global gap of 2 with linear cost per join/leave. From here on, we assume that w.h.p. a constant bound C on the global gap is maintained.

If the global gap is small, then a node can estimate the size of the network by examining its own level. This is stated in the following lemma:

Lemma 4.1 *Let $G(t)$ be a dynamic de Bruijn graph with global gap C. Then for all $u \in V(t)$: $2^{\ell(u)-C} \leq |V(t)| \leq 2^{\ell(u)+C}$.*

4.2 Quorum selection

For a node u to establish a read or a write quorum, it initiates $k\sqrt{2^{\ell(u)+2C}}$ random walk messages. When a node u initiates a random walk it creates a message M with a hop-count $\ell(u)$, an id $u.id$, and appends any payload A to it, i.e., $M = \langle \ell(u), u.id, A \rangle$. Each node (including u) that receives a message $\langle j, id, A \rangle$ with a non zero hop-count $j > 0$, forwards a message $M' = \langle j-1, id, A \rangle$, randomly to one of its outgoing edges. If $(u, v) \in E$ then the probability that u forwards the message to v is:

$$\Pr[u \text{ forwards to } v] = \frac{1}{2^{\max\{\ell(v)-\ell(u)+1,1\}}} \tag{2}$$

By induction on the splits and merges of the dynamic graph it is clear that the above function (Equation 2) is a well defined probability function.

We call the node that receives a message with hop-count 0 the destination of the message. As a practical matter, it should be clear that a destination node opens the message payload and executes any operation associated with it, such as updating data or responding to a query. These matters are specific to the application, and are left outside the scope of this paper.

4.3 Analysis of quorum selection

Let $G(t)$ be a dynamic graph on n nodes. Recall the probability distribution of message forwarding as defined in Section 4.2, Equation 2. We represent this distribution using a weighted adjacency $(n \times n)$-matrix $M(t)$ as follows:

$$m_{v,u} = \Pr[u \text{ forwards to } v] = \begin{cases} \frac{1}{2^{\max\{\ell(v)-\ell(u)+1,1\}}} & (u, v) \in E(t), \\ 0 & \text{otherwise.} \end{cases}$$

The main result we pursue, namely, that our construction realizes a non-uniform PQS, stems from the two propositions below. Due to space limitations, their proofs are deferred to an appendix. We first explicitly state the stationary distribution on the dynamic graph, and then prove that the weighed random walk algorithm makes a perfect sampling of this distribution.

Theorem 1 *The stationary distribution of $M(t)$ is the vector x, such that $\forall v \in V(t)$, $x_v = \frac{1}{2^{\ell(v)}}$*

For every $t > 0$, denote $x(t)$ as the stationary distribution on $M(t)$. We now show that the random walk algorithm described in Section 4.2 chooses nodes according to $x(t)$.

Theorem 2 *The mixing time of a random walk on $M(t)$ starting from a node of level k is k.*

The theorems above together imply that our graph maintenance algorithm together with our random walk quorum selection strategy implement a non-uniform selection strategy over the members of $V(t)$, where the probability of choosing $v \in V(t)$ is $1/2^{\ell(v)}$.

As an immediate consequence of the propositions above, we have our main theorem as follows:

Theorem 3 *For a system S on a dynamic graph with global gap C, the quorum selection strategy as described above forms a e^{-k^2}-intersecting probabilistic quorum system.*

Proof: The theorem follows from the fact that by assumption, each quorum access includes at least $k\sqrt{2^{\ell(u)+2C}} \geq k\sqrt{n}$ independent selections, each one done according to the distribution $x(t)$. □

5 Quorum Evolution

In this section we describe the evolution algorithm for maintaining *dynamic ε-intersecting quorum systems*. For such a construction, quorums need to evolve along with the growth of the system in order to maintain their intersection properties. This property must be maintained in spite of any execution sequence of join and leave events that may be given by an adversary.

One trivial solution would be to choose new quorums instead of the old ones each time the network's size multiplies. Such a solution has a major drawback, it requires a global overhaul operation that may affect all the system at once. Even if we consider amortized costs, this process requires to change the state of $\sqrt{|V|}$ nodes for some join events. In contrast, our evolution scheme w.h.p. resorts only to local increments for each join or leave event, each causing only a constant number of nodes to change their state.

The intuition for our algorithm comes from the following simple case. Suppose the network is totally balanced, i.e., all nodes have the same level m, and a quorum with $2^{m/2}$ data entries is randomly chosen. Further assume that after a series of join events, the network's size multiplies by 4 and all nodes have level $m+2$. Our evolution algorithm works as follows in this simple scenario. Each time a node splits, each data entry stored on the split node randomly chooses which sibling to move to. In addition, if the node that splits has an even level then each of its data entries also creates one more duplicate data entry and randomly assigns it to a new node. Thus the number of data entries doubles from $2^{m/2}$ to $2^{(m+2)/2}$ and each data entry is randomly distributed on the network.

Our evolution algorithm simulates this behavior on approximately balanced networks. Thus, its success relies on the fact that the global gap of the dynamic graph w.h.p. is at most C. In order to avoid fractions, we set the bound C on the global gap to be an even number.

5.1 Informal description of the evolution algorithm

Recall that a join (respectively, leave) event translates to a split (respectively, merge) operation on the dynamic graph. We now explain how the random walk algorithm is enhanced, and what actions are taken when a split or a merge operation occurs.

We divide the levels of the graph into phases of size C, all the levels $(i-1)C+1, \ldots, iC$ belong to phase iC. When a node in phase iC wants to establish a quorum, it sends $k2^{(i+1)C/2}$ random walk messages. Each such message also contains the *phase* of the sender which is iC.

When two nodes are merged, the data entries are copied to the parent node. If the parent node is later split, we want each data entry to go to the sibling it originally came from. Otherwise, the distribution of the data entry's location will be dependent on the execution sequence. Thus, each data entry also stores all the random choices it has made as a sequence of random bits called *dest*. When an entry is first created, *dest* is set to the id of the node that the data entry is in.

When a node of level i is split into two nodes of level $i+1$, there are two possibilities: Either $|dest| \geq i+1$ and the data entry moves according to the (i+1)th bit of *dest*. Otherwise, the data entry randomly chooses which one of the two sibling to move to, and it records this decision by adding the appropriate bit to *dest*.

The number of data entries is increased only when a data entry is split on a node whose level is a multiple of C. If a data entry with phase iC is involved in a split operation on a node with level iC then $2^{C/2}-1$ new data entries with phase $(i+1)C$ are created. These data entries are randomly distributed using the random walk algorithm.

Additionally, whenever a random walk message from a node in phase jC arrives to a node u with phase $(j+1)C$, we simulate as if the message first arrived at an ancestor node of level jC that is a prefix of u, and later this ancestor node had undergone some split operations. Thus, if a phase $(j+1)C$ node receives a message with hop count 0 initiated by a node in phase jC then, in addition to storing the data entry, the node also creates $2^{2/C}-1$ new data entries with phase $(j+1)C$. This simulation technique is recursively expanded to cases where a node in phase $(j+\ell)C$ receives a message initiated by a node in phase jC.

5.2 Evolution algorithm

Enhanced random walk: Denote $phase(i) = C\lceil i/C \rceil$. When a node u initiates a random walk it creates a message M with a hop-count $\ell(u)$, phase $phase(\ell(u))$, id $u.id$, and payload A to it, i.e., $M = \langle \ell(u), phase(\ell(u)), u.id, A \rangle$. Each node that receives a message $\langle j, ph, id, A \rangle$ with a non zero hop-count $j > 0$, forwards a message $M' = \langle j-1, ph, id, A \rangle$, randomly to one of its outgoing edges v with probability $\Pr[u \text{ forwards to } v] = \frac{1}{2^{\max\{\ell(v)-\ell(u)+1,1\}}}$.

Nodes store information as a *data entry* of the form $(dest, ph, id, A)$, where $dest$ is a sequence of bits that describes the location of the entry, ph is the phase, id is the identity of the quorum initiator, and A is the payload.

When node w receives a message $M = \langle 0, ph, id, A \rangle$ it stores the data entry $(w.id, ph, id, A)$. If $phase(\ell(w)) > ph$ then for every i such that $\lceil ph/C \rceil < i \leq \lceil \ell(w)/C \rceil$, w sends $2^{C/2}-1$ messages of the form $\langle \ell(w), iC, id, A \rangle$.

Create: A node u creates a quorum by initiating $k2^{(phase(\ell(u))+C)/2}$ enhanced random walk messages.

Split: Suppose node u wants to enter the system, and $v = \langle a_1, \ldots, a_k \rangle$ is the node to be split into nodes $\langle a_1, \ldots, a_k, 0 \rangle$ and $\langle a_1, \ldots, a_k, 1 \rangle$. For every data entry (d, ph, id, A) held in v do the following. If $|d| \geq k+1$ then store (d, ph, id, A) at node $\langle a_1, \ldots, a_k, dest_{k+1} \rangle$ where $dest_i$ is the ith bit of $dest$. Otherwise, if $|d| < k+1$ then with uniform probability choose $i \in \{0,1\}$ and send to node $\langle a_1, \ldots, a_k, i \rangle$ the message $\langle 0, ph, id, A \rangle$. Node $\langle a_1, \ldots, a_k, i \rangle$ will handle this message using the enhanced random walk algorithm (in particular, if the split has crossed a phase boundary, it will generate $2^{C/2}-1$ new data replicas).

Merge: Suppose node u wants to leave the system, and the twin nodes $\langle a_1, \ldots, a_k, 0 \rangle$, $\langle a_1, \ldots, a_k, 1 \rangle$ are the nodes that merge into node $v = \langle a_1, \ldots, a_k \rangle$. If u and one of the twins swap their ids then they also swap the data entries that they hold. After the swap, the merged node $v = \langle a_1, \ldots, a_k \rangle$ copies all the data entries that the nodes with ids $\langle a_1, \ldots, a_k, 0 \rangle, \langle a_1, \ldots, a_k, 1 \rangle$ held.

5.3 Analysis of quorum evolution

Given a network $G(t)$ on n nodes, we seek to show that the evolved quorum's distribution is at least as good as the flat access scheme $f(x(t), k\sqrt{n})$. So we must show a set of data entries that are independently distributed, whose size is at least $k\sqrt{n}$. Note that the existence of some of the data entries is dependent on the execution history. Therefore, it is not true that all data entries are independently distributed. However, we use a more delicate argument in which we analyze the size of a subset of the data entries whose existence is independent of the execution sequence.

The main result we pursue is that a non-uniform PQS is maintained despite any system reconfiguration, and is given in the Theorem below. The following two lemmas are crucial for proving it. Their proofs are in the appendix.

Lemma 5.1 *For any time t, data entry D, the distribution of D's location on $V(t)$ is $x(t)$.*

DEFINITION 5.1 *Denote $L(t)$ as the lowest phase on $G(t)$, $L(t) = \min_{v \in V(t)} phase(\ell(v))$.*

Lemma 5.2 *Let $t > 0$, and let the dynamic graph $G(t)$ have global gap C. Consider any quorum initiated by a node u at phase i with payload A. If $L(t) \geq i$ then the number of data entries of the form (d, ph, u, A) such that $ph \leq L(t)$ is exactly $k2^{((L(t)+C)/2}$.*

Theorem 4 *On dynamic networks with global gap C, the evolution algorithm maintains a dynamic $e^{k^2/2}$-intersecting quorum system.*

Proof: By Lemma 5.1 the locations of all data entries of all quorums are distributed by $x(t)$, the stationary distribution of $M(t)$. Consider a quorum initiated at a phase i node. If $L(t) < i$ then the initial $k2^{(i+C)/2} \geq k\sqrt{|V(t)|}$ data entries suffice. If $L(t) \geq i$ then by Lemma 5.2 every quorum has $k2^{(L(t)+C)/2}$ entries whose existence is independent of the execution history. Since the network has global gap of C, then $k2^{(L(t)+C)/2} \geq k\sqrt{|V(t)|}$. Thus at any time t, the evolving probability distribution p_t^r of the above subset of data entries of any quorum, for any establishment time r, is a flat access strategy $f(x(t), m)$ in which $m \geq k\sqrt{|V(t)|}$. By Lemma 3.1 this access scheme forms an $e^{k^2/2}$-intersecting quorum system as required. □

Our construction implements, for any history of events, access strategies and an evolution strategy that maintains the evolving probability distribution p_t^r as a flat access strategy on $V(t)$ using the distribution $x(t)$ with more than $\sqrt{|V(t)|}$ independent choices. Thus, at any time t, all quorums (both newly established and evolved) are ε-intersecting.

6 Performance Analysis

Our protocols hinge on the network balancing algorithms we employ, e.g., from [1], and on their ability to maintain the bound C on the global level gap. We note that the network construction of [1] incurs a constant number of state-changes per join/leave and a logarithmic number of messages. It maintains the global gap bound C w.h.p. Below, the analysis stipulates that the global gap C is maintained, and calculates additional costs incurred by our algorithm.

Join/leave complexity. When a new process joins the system, it may cause a split of a node whose level is a multiple of C. In that case, we allocate a constant number of new data entries, that incur a constant number of random walks. Thus, the message cost is $O(\log(n))$ and the number of processes incurring a change in their state is constant. Leave events generate one message and a state change to one process.

Quorum access complexity. When selecting a quorum, we initiate $O(\sqrt{n})$ random walks in parallel. The parallel time is $O(\log(n))$, and the total number of messages is $O(\sqrt{n}\log(n))$.

Load and Resilience. These measures are defined and analyzed in Appendix B.

7 Discussion

In this paper we assumed the read-write ratio to be roughly equal. It is possible to extend the techniques of this paper to differentiate between read-quorums and write-quorums, and achieve better performance. Given any read-write ratio, instead of having all operations select $cn^{1/2}$ nodes, read operations select cn^{α} nodes, and write operations select $cn^{1-\alpha}$ nodes for some predetermined $0 < \alpha < 1$.

We presented a system with a constant $1 - \varepsilon$ intersection probability. In the AntWalk system [20], $\sqrt{n \log n}$ processes are randomly chosen thus leading to intersection with high probability. Our quorum selection and evolution algorithm can be modified along similar lines to achieve a high probability dynamic intersecting quorum system.

Our analysis is sketched in a model in which changes are sequential. While we believe our construction to be efficient in much stronger settings, where a large number of changes may occur simultaneously, it is currently an open problem to provide a rigorous analysis.

The fault tolerance analysis concerns the robustness of the data which the system stores against $O(n)$ failures. While the data will not be lost due to such catastrophic failure, clearly our constant degree network, which is used to access the data, may disconnect. Network partitioning can be reduced by robustifing the network through link replication. But unless each node has $O(n)$ links, $O(n)$ failures will disconnect any network. Once the network is partitioned, the problem of rediscovering the network's nodes is addressed in [9, 2]. When the network is reconnected, the dynamic de-Bruijn can be reconstructed. After recovering from this catastrophic failure, the system will maintain consistency, since the information itself was not lost.

References

[1] I. Abraham, B. Awerbuch, Y. Azar, Y. Bartal, D. Malkhi and E. Pavlov. A Generic Scheme for Building Overlay Networks in Adversarial Scenarios. In *International Parallel and Distributed Processing Symposium* (IPDPS 2003), April 2003, Nice, France.

[2] I. Abraham, D. Dolev. Asynchronous Resource Discovery. In proceedings of *the 22nd ACM Symposium on Principles of Distributed Computing* (PODC 2003). June 2003.

[3] N. G. de Bruijn. A combinatorial problem, Konink. Nederl. Akad. Wetersh. Verh. Afd. Naturk. Eerste Reelss, A49 (1946), pp. 758-764.

[4] W. Feller. *An Introduction to Probability Theory and Its Applications, volume 1*. John Wiley & Sons, New York, 3rd edition, 1967.

[5] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[6] P. Fraigniaud and P. Gauron. The Content-Addressable Network D2B. Technical Report 1349, LRI, Univ. Paris-Sud, France, January 2003.

[7] `http://gnutella.wego.com`.

[8] Z. J. Haas and B. Liang. Ad hoc mobility management with randomized database groups. In *Proceedings of the IEEE Internation Conference on Communications*, June 1999.

[9] M. Harchol-Balter, T. Leighton, and D. Lewin. Resource Discovery in Distributed Networks. In *Proc. 15th ACM Symp. on Principles of Distributed Computing*, May 1999, pp. 229-237.

[10] F. Kaashoek and D. R. Karger. Koorde: A Simple Degree-optimal Hash Table. In *2nd International Workshop on Peer-to-Peer Systems* (IPTPS '03), February 2003, Berkeley, CA.

[11] H. Lee and J. L. Welch. Applications of Probabilistic Quorums to Iterative Algorithms. In *Proceedings of 21st International Conference on Distributed Computing Systems* (ICDCS-21), Pages 21-28. April, 2001.

[12] H. Lee and J. L. Welch. Randomized Shared Queues. Brief announcement in *Twentieth ACM Symposium on Principles of Distributed Computing* (PODC 2001).

[13] D. Malkhi, M. Reiter. Secure and Scalable Replication in Phalanx. Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems (SRDS '98), October 1998, Purdue University, West Lafayette, Indiana, pages 51–60.

[14] D. Malkhi and M. Reiter. An Architecture for Survivable Coordination in Large Distributed Systems. IEEE Transactions on Knowledge and Data Engineering, 12(2):187–202, April 2000.

[15] D. Malkhi, M. Reiter, A. Wool and R. Wright. Probabilistic quorum systems. The Information and Computation Journal 170(2):184–206, November 2001.

[16] M. Naor and U. Weider. Novel architectures for P2P applications: the continuous-discrete approach. In proceedings pf *Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2003)*, June 2003.

[17] M. Naor and U. Wieder. Scalable and Dynamic Quorum Systems. In proceedings of *the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, June 2003.

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

[19] G. Pandurangan, P. Raghavan and E. Upfal. Building low-diameter p2p networks. In *Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, 2001.

[20] D. Rataczjak. Decentralized Dynamic Networks. M. Eng. Thesis Proposal, MIT, May 2000.

[21] J. Saia, A. Fiat, S. Gribble, A. Karlin, and S. Saroiu. Dynamically Fault-Tolerant Content Addressable Networks, In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, March 2002, Cambridge, MA USA

A Deferred Proofs

A.1 Proofs for Section 4.3

Proof: (of Theorem 1) For this proof, we make use of the following technical lemma. Intuitively, it states each node in $G(t)$ is pointed to by edges whose total weight is proportional to its own level. The precise sense in which this holds is stated by the lemma.

Lemma A.1 *Let $v \in G(t)$ be a node whose id is $v.id = \langle a_1, a_2, \ldots, a_k \rangle$. Denote by $N_0(v)$ the nodes in $G(t)$ whose id's match $\langle 0, a_2, \ldots, a_k \rangle$, or are a prefix of it, or have a postfix added to it. (Similarly, denote by $N_1(v)$ the nodes that match $\langle 1, a_2, \ldots, a_k \rangle$, its prefix or postfix.) Then*

$$\sum_{u \in N_i(v)} m_{v,u} \frac{1}{2^{\ell(u)}} = \frac{1}{2^{\ell(v)+1}} .$$

Proof: (of Lemma A.1) By our graph construction, there are two cases to consider. The first one is $|N_0(v)| = 1$. In this case, denote $N_0(v) = \{w\}$, and it follows that $\ell(w) \leq \ell(v)$. We therefore have:

$$m_{v,w} \frac{1}{2^{\ell(w)}} = \frac{1}{2^{\ell(v)-\ell(w)+1}} \frac{1}{2^{\ell(w)}} = \frac{1}{2^{\ell(v)+1}} .$$

The second case is $|N_0(v)| > 1$. Then $\forall w \in N_0(v) : \ell(w) > \ell(v)$, and the nodes $w \in N_0(v)$ have the form $w.id = \langle 0, a_2, \ldots, a_k \rangle$, or are a prefix of it or have a postfix appended to it. By a trivial induction on split and merge operations, we have $\sum_{w \in N_0(v)} \frac{1}{2^{\ell(w)}} = \frac{1}{2^{\ell(v)}}$. Thus:

$$\sum_{w \in N_0(v)} m_{v,w} \frac{1}{2^{\ell(w)}} = \sum_{w \in N_0(v)} \frac{1}{2} \frac{1}{2^{\ell(w)}} = \frac{1}{2} \sum_{w \in N_0(v)} \frac{1}{2^{\ell(w)}} = \frac{1}{2^{\ell(v)+1}} .$$

By symmetry, the analogous statement on $N_1(v)$ also holds. □

Now we get back to the main proof to Theorem 1. Showing $\sum_{v \in V(t)} x_v = 1$ is trivially done by induction on the series of node additions and removals.

For every $v \in V(t)$, we will show that $\sum_{u \in V(t)} m_{v,u} x_u = x_v$. Suppose $v = \langle a_1, a_2, \ldots, a_k \rangle$. In our graph, nodes that have directed links to v are those in $N_0(v)$ and in $N_1(v)$. The sets $N_0(v)$ and $N_1(v)$ are disjoint by definition. Thus, using Lemma A.1, we have:

$$\sum_{u \in V(t)} m_{v,u} x_u = \sum_{w \in N_0(v) \text{ or } w \in N_1(v)} m_{v,w} x_w = \frac{1}{2^{\ell(v)+1}} + \frac{1}{2^{\ell(v)+1}} = \frac{1}{2^{\ell(v)}} = x_v \ .$$

□

Proof: (of Theorem 2) The proof makes use of the following Lemma.

Lemma A.2 *All the bits of a random walk message with hop count* 0 *are independently uniformly distributed.*

Proof: (of Lemma A.2) We show by induction that after i hops, the walk reaches a node whose bits beyond the first $(k - i)$ bits are selected independently uniformly at random.

Let v be a starting node whose level is k. Denote v's id by $v.id = \langle a_1, \ldots, a_k \rangle$. The first hop must go to a node whose id matches $\langle a_2, \ldots, a_k, r_1 \rangle$, or a prefix thereof, or with a postfix appended. and where r_1 is chosen to be $0/1$ with uniform probability. In case the destination node has a postfix appended, the postfix bits are chosen uniformly at random by our construction, since every split operation divides the weight of an edge pointing to the split node into half. Thus, we have the induction basis.

For the induction step, suppose that after $i - 1$ hops, the walk reaches a node $\langle a_i, \ldots, a_k, \sigma \rangle$, or a prefix thereof, where σ is a sequence of randomly chosen bits. Then at hop i we move to a node whose id matches $\langle a_{i+1}, \ldots, a_k, \sigma, r_i \rangle$, or a prefix thereof, or with a random postfix added to it, and where r_i is chosen to be 0 or 1 with uniform probability. By the same argument as above, in case the destination node has a postfix appended, the postfix bits are chosen uniformly at random.

Thus when the hop count reaches zero, the bits of the target are all random, independent and uniformly selected. □

We also note the following simple combinatorial fact:

Fact A.1 *For $k \leq j$, given a fixed sequence of bits $A = \langle a_1, \ldots, a_k \rangle$, and a sequence $B = \langle b_1, \ldots, b_j \rangle$ of bits each independently chosen with uniform probability then*

$$Pr[A \textit{ is a prefix of } B] = \frac{1}{2^k}$$

Now we get back to the main proof of Theorem 2. A random walk message starting at a level k node will walk k steps until its hop count reaches 0. By Lemma A.2 all its bits are independently uniformly chosen. Thus for $v \in V(t)$ the probability that the random walk reaches v by Fact A.1 is $2^{-\ell(v)}$.

□

A.2 Proofs for Section 5.3

Proof: (of Lemma 5.1) For every data item $D = (dest, ph, id, A)$ we prove the following by induction: *dest* is a sequence of bits that are independently and uniformly distributed and D is stored in the node whose id is a prefix of *dest*.

When a data entry is created it is stored at a node v chosen by the random walk algorithm and *dest* is set to v. By Lemma A.2 all the bits of v are independently uniformly distributed. Thus the induction base holds.

Now assume at time t entry D is stored in node $v = \langle a_1, \ldots, a_k \rangle$. Suppose the next event $e(t+1)$ is a leave that causes a merge operation on v and its twin. This will cause D to be stored in $\langle a_1, \ldots, a_{k-1} \rangle$ which still remains a prefix of *dest*.

Suppose $e(t+1)$ is a join that causes a split operation on v. If $|dest| \geq k+1$ then D moves to $\langle a_1, \ldots, a_k, dest_{k+1} \rangle$ and the claim holds. If *dest* has only k bits then the evolution algorithm independently with uniform probability chooses which sibling to move to and thus the new destination maintains the induction hypothesis.

Therefore, the *dest* sequence is i.i.d. and the data entry resides in a node which is a prefix of *dest*. Thus by Fact A.1 and Theorem 1, the location of D at time t has a distribution $x(t)$ □

Proof: (of Lemma 5.2) The proof is by induction. When a quorum is established by a node of phase i, it creates $2^{(i+C)/2}$ data entries with phase i. Since the gap is at most C, i is either $L(t)$ or $L(t)+C$. If $i = L(t)$ then the induction base holds. If $i = L(t)+C$ then the base holds vacuously, since $L(t) < i$.

If $e(t+1)$ is a join that causes $L(t+1) = L(t)+C$ then all of the data entries with phase at most $L(t)$ must have been spilt on a node of level $L(t)$ (or split on a lower level node that simulates this split). If $t+1$ is the first time that $L(t+1) \geq phase(id)$, and the induction hypothesis was vacuously true before $t+1$, then the phase of the quorum establisher was $L(t)+C$, thus $k2^{((L(t)+2C)/2}$ data entries of phase $l(t)+C$ exist in the network as required.

Otherwise, by the induction hypothesis there are $k2^{(L(t)+C)/2}$ data entries of phase at most $L(t)$. Each one creates $2^{C/2}-1$ more data entries of phase $L(t)+C$. Together, there are $k2^{((L(t)+2C)/2}$ entries of phase at most $L(t)+C$. □

B Measures and Analysis of Probabilistic Quorum Systems

B.1 Measures

Traditionally, three measures were defined to assess the quality of quorum systems: the load, the fault tolerance, and the failure probability of the system. In this section, we briefly recount the definition of these measures, and then apply these measures to our dynamic probabilistic quorum construction.

Load The load of a quorum system, defined by Naor and Wool [18], captures the probability of accessing the busiest server. Load is a measure of efficiency. All other things being equal, systems with lower load can process more requests than those with higher load.

DEFINITION B.1 (Load) *Let w be a strategy for a set system $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ over a universe U. For a server $u \in U$, the* load *induced by w on u is $l_w(u) = \sum_{Q_i \ni u} w(Q_i)$. The* load *induced by a strategy w on $\mathcal{Q}$ is $L_w(\mathcal{Q}) = \max_{u \in U}\{l_w(u)\}$.*

Let $\langle \mathcal{Q}, w \rangle$ be an ε-intersecting quorum system. Then the load *of $\langle \mathcal{Q}, w \rangle$ is $L(\langle \mathcal{Q}, w \rangle) = L_w(\mathcal{Q})$.*

It is known that for any quorum system $\mathcal{Q}$ over n servers, $L(\mathcal{Q}) \geq \max\{\frac{1}{c(\mathcal{Q})}, \frac{c(\mathcal{Q})}{n}\}$ where $c(\mathcal{Q})$ is the size of the smallest quorum in $\mathcal{Q}$ [18]. In particular, this implies that for any quorum system $\mathcal{Q}$, $L(\mathcal{Q}) \geq 1/\sqrt{n}$.

Fault tolerance The fault tolerance and failure probability measures for probabilistic quorum systems are taken from [15].

Definition B.2 (δ-high quality quorums) *Let $\langle \mathcal{Q}, w \rangle$ be an ε-intersecting quorum system, and let $0 \leq \delta \leq 1$ be given. The set of* δ-high quality quorums *of $\langle \mathcal{Q}, w \rangle$ is*

$$\mathcal{R} = \{Q \in \mathcal{Q} : \Pr[Q \cap Q' \neq \emptyset] \geq 1 - \delta\},$$

where $Q' \in \mathcal{Q}$ is chosen according to w.

Definition B.3 (High quality quorums) *Let $\langle \mathcal{Q}, w \rangle$ be an ε-intersecting quorum system. Then the* high quality quorums *of $\langle \mathcal{Q}, w \rangle$ are the $\sqrt{\varepsilon}$-high quality quorums of $\langle \mathcal{Q}, w \rangle$.*

Definition B.4 (Fault tolerance) *Let $\langle \mathcal{Q}, w \rangle$ be an ε-intersecting quorum system. Let $\mathcal{R}$ be the set of high quality quorums of $\langle \mathcal{Q}, w \rangle$, and let $\mathcal{S} = \{S : S \cap Q \neq \emptyset$ for all $Q \in \mathcal{R}\}$. Then the* fault tolerance *$A(\langle \mathcal{Q}, w \rangle)$ is $\min_{S \in \mathcal{S}} |S|$.*

Failure probability

Definition B.5 (Failure probability) *Let $\langle \mathcal{Q}, w \rangle$ be an ε-intersecting quorum system, and let $\mathcal{R}$ be the set of high quality quorums of $\langle \mathcal{Q}, w \rangle$. The* failure probability *$F_p(\langle \mathcal{Q}, w \rangle)$ is the probability that every $Q \in \mathcal{R}$ contains at least one crashed server, under the assumption that each server in U crashes independently with probability p.*

B.2 Analysis

We now analyze our PQS according to these measures. At any time $t > 0$, let the system $S = U(t)$ have n processes.

Lemma B.1 *The load on a process v during quorum selection at time t is $O(1/\sqrt{n})$.*

Proof: This lemma stems from the fact that we have $\log(n) - C \leq \ell(s) \leq \log(n) + C$. Note that $L(t) \leq \log(n) + C$. The size of any quorum established at time t is at most $2^{(phase(\log(n)+C)+C)/2} \leq 2^2 C\sqrt{n}$. Thus, the probability of s being selected for any quorum is at most

$$1 - \left(1 - \frac{1}{2^{\ell(s)}}\right)^{\sqrt{n}2^2C} \leq 1 - \left(1 - \frac{1}{2^{\log(n)-C}}\right)^{\sqrt{n}2^2C} \leq \frac{\sqrt{n}2^2C}{2^{\log(n)-C}} = \frac{2^{3C}}{\sqrt{n}} .$$

□

As the system grows, the load above continues to hold. If the system dramatically diminishes, the relative fraction of data entries could grow, causing high load on processes. Naturally, a practical system must deploy garbage collection mechanisms in order to preserve resources. The discussion of garbage collection is left outside the scope of this paper.

For the availability analysis, note that all quorums of size $k2^C\sqrt{n}$ are high quality.

Lemma B.2 *The fault tolerance is* $n - k2^C\sqrt{n} + 1 = \Omega(n)$.

Proof: Because only $k2^C\sqrt{n}$ processes need be available in order for some (high quality) quorum to be available, the fault tolerance is $n - k2^C\sqrt{n} + 1 = \Omega(n)$. □

Lemma B.3 *The failure probability* $F_p = e^{-\Omega(n)}$.

Proof: Let p denote the independent failure probability of processes. For the system to fail, at least $n - k2^C\sqrt{n} + 1$ processes must fail. By a standard analysis of threshold-resilience using a Chernoff bound (see [18]), the failure probability can be bounded by the following:

$$F_p = \Pr(\#\text{fail} > n - k2^C\sqrt{n}) \leq e^{-2n\left(\frac{k2^C}{\sqrt{n}}+\delta\right)^2} = e^{-\Omega(n)} \ ,$$

for all $p \leq 1 - 2\frac{k2^C}{\sqrt{n}} - \delta$. This failure probability is optimal [18]. □