Scalable and Dynamic Quorum Systems*

Moni Naor[†‡] Udi Wieder [‡]

Abstract

We investigate issues related to the probe complexity of quorum systems and their implementation in a dynamic environment. Our contribution is twofold. The first regards the algorithmic complexity of finding a quorum in case of random failures. We show a tradeoff between the load of a quorum system and its probe complexity for non adaptive algorithms. We analyze the algorithmic probe complexity of the *Paths* quorum system suggested by Naor and Wool in [28], and present two optimal algorithms. The first is a non adaptive algorithm that matches our lower bound. The second is an adaptive algorithm with a probe complexity that is linear in the cardinality of the smallest quorum set. We supply a constant degree network in which these algorithms could be executed efficiently. Thus the *Paths* quorum system is shown to have good balance between many measures of quality. Our second contribution is presenting *Dynamic Paths* - a suggestion for a dynamic and scalable quorum system, which can operate in an environment where elements join and leave the system. The quorum system could be viewed as a dynamic adaptation of the *Paths* system, and therefore has low load high availability and good probe complexity. We show that it scales gracefully as the number of elements grows.

1 Introduction and Motivation

Quorum systems serve as a basic tool providing a uniform and reliable way to achieve coordination between processors in a distributed system. Quorum systems are defined as follows:

Definition 1. *Let U be a universe of n elements. A set system $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is said to be a* quorum system *over the universe U if $\forall i\ S_i \subseteq U$ and $\forall i, j\ S_i \cap S_j \neq \varnothing$. Each set S_i is referred to as a* quorum set *or simply as a* quorum.

Quorum systems have been used in the study of distributed control and management problems such as mutual exclusion (cf. [10],[33]), data replication protocols (cf. [10]) and secure access control ([27]). In many applications of quorum systems the underlying universe is associated with a network of processors, and a quorum is employed by accessing each of its elements. For example, in a typical implementation of mutual exclusion using quorum systems, processors request access to the critical section from all members of a quorum. A processor can enter its critical section only if it receives permission from all processors in a quorum. The intersection property guarantees the integrity of the mutual inclusion. In a typical application of data replication, the quorum sets are divided into reading quorums and writing quorums where each reading quorum intersects each writing quorum. When a data item is added to the system, it is written into all the members of a writing quorum. A data item is searched by querying all the members of a reading quorum. The intersection property guarantees the effectiveness of the search. We investigate two aspects of quorum systems:

*Research supported in part by the RAND/APX grant from the EU Program IST

†Incumbent of the Judith Kleeman Professorial Chair.

‡Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science. Rehovot 76100 Israel. {moni.naor, udi.wieder}@weizmann.ac.il

1. It is often assumed that processors can somehow find and communicate with one another. We analyze algorithms for finding quorum systems in a distributed network while taking into account the *network implementation*; i.e., the network and the quorum system should be compatible such that elements from the same quorum are connected to one another. We supply algorithms for finding a quorum set (even in the case of failures) and analyze their running time and communication complexity. In this setting non-adaptive algorithms are attractive since they can be executed in parallel.

2. The setting in which the quorum operates is often dynamic, and should accommodate changes in the quorum system over time. See for instance [21],[33]. We address the problem of designing a quorum system that is fit for a scalable and dynamic environment where processors leave and join at will. Abraham and Malkhi [3] address this problems when the intersection property is not guaranteed but rather occurs with high probability.

1.1 Scalable Dynamic Data Structures - P2P

Recently a new approach for construction of dynamic distributed data structures on overlay networks was suggested, which offers excellent scalability. The main motivation for this line of research comes from the rise in popularity of P2P application, therefore the attention was put on the construction of distributed hash tables (cf. [26],[23], [34], [32]). In these works an overlay network is built dynamically. Processors may fail (with some probability) and are allowed to join and leave. Each processor holds some data items. The construction in [26] for instance, guarantees that any data item could be found in logarithmic time, while imposing small load on every processor. In this paper we suggest quorum systems that operate in a dynamic peer-to-peer model. We combine techniques developed in these papers, mainly [26] and [32], with appropriate quorum systems, and provide the distributed algorithms for finding the quorums. We allow two types of events:

1. A Processor may temporarily fail (halt). The failure of a processor occurs with some fixed probability and is independent from failures of other processors in the network. It is desired that the probability that a live quorum is found be as high as possible.

2. Processors may wish to join the system or to leave it (a long term failure of a processor could be regarded as if the processor left the system). It is desired that the quorum sets be updated such that these processors are included/excluded from the system.

1.2 Measures of Quality

The metrics that measure the quality of a dynamic quorum system relate both to its *combinatorial* structure and to its capability of being implemented in a distributed network. The following metrics were analyzed by Naor and Wool in [28] and are used to measure the quality of static systems as well.

- `Load -` A strategy is a distribution over quorum sets, giving each quorum set an access probability (i.e., the probability by which it is accessed by the user). A strategy induces a load on each element, which is the sum of the probabilities of quorums it belongs to. This represents the fraction of the time an element is used. For a given quorum system $\mathcal{S}$, the load $\zeta(\mathcal{S})$ is the minimal load on the busiest element, minimizing over the strategies. The load measures the quality of the quorum system in the following sense: if the load is low, then each element is accessed rarely, thus it is free to perform other unrelated tasks. Let c be the cardinality of the smallest quorum set. Naor and Wool prove in [28] the following lemma:

  Lemma 2. *The load of a quorum system is always at least* $\max\{\frac{1}{c}, \frac{c}{n}\}$ *which implies that* $\zeta(\mathcal{S}) \geq \frac{1}{\sqrt{n}}$.

- `Availability -` Assuming that each element fails with probability p, what is the probability F_p, that the surviving elements do not contain any quorum? This failure probability measures how resilient the system is, and we would like F_p to be as small as possible.

The Load is especially important if the application of the quorum system involves replication of data, as was described in the previous section. In this case the load is proportional to the fraction of data each element has to hold, and therefore smaller load means that each processor needs to allocate a smaller amount of memory. The notion of availability is important when dealing with *temporary faults*. The most common strategy to deal with faults is to *bypass* them; i.e., find a quorum set for which all processors are alive. This introduces the following notion:

- `Algorithmic probe complexity` - The complexity of the algorithms for finding a quorum should be low. Even if all processors are alive the *network* should allow easy access to elements of the same quorum system. In case some elements fail, finding a live quorum set can be a difficult algorithmic task. Peleg and Wool analyzed in [30] the probe complexity of several quorum systems. They assume that an adversary decided which elements fail and analyzed the number of elements needed to be probed before either a living quorum is found or an evidence for the lack of it. They assume that each probing takes $O(1)$; i.e., they ignore the complexity caused by the implementation of the network. Hassin and Peleg extend these results in [14] to the case where each processor fails with some fixed probability . The *Algorithmic probe complexity* is the actual time and message complexity needed to find a live quorum. It is determined by the *network* and by the quorum system. A related term is the *Cost of Failures* introduced by Bazzi [7]. Given a network implementation and an algorithm for finding quorums, the cost for failures measures the average communication overhead caused by encountering a faulty processor.

The introduction of a dynamic environment requires another set of demands:

- `Integrity-` A new processor that joins the system, and a processor that leaves the system, should change the quorum sets. The integrity of the system should be preserved in two aspects: First the intersection property must hold. Bearden and Bianchini suggest in [8] a protocol for an online adjustment of quorum systems without compromising the integrity of the intersection property *during* the adaptation. It is necessary that the adaptations themselves do not corrupt the intersection property of the quorum system; i.e., that the intersection property holds after the adaptations took place. The second aspect is application oriented. Quorums that were used in the past (say for mutual exclusion) might not be legal quorum sets after the adaptation. It is necessary that when an adaptation occurs, the intersection guarantee that the quorum system supplies the application is not compromised.

- `Scalability-` The number of elements in the quorum system may increase over time. The increase in the size of the system should maintain the good qualities of it, i.e., it should decrease the load on each processor and increase the availability of the system. It is important that when the system scales the algorithmic probe complexity would remain low. Finally the Join and Leave operation should be applied with low time and message complexity.

1.3 New Results and Paper Organization

The paper is divided into two parts. In the first part, we show a tradeoff between the load and the non-adaptive probe complexity of quorum systems (Section 2), thus proving a lower bound for non-adaptive probe complexity. In Section 3 we show a non-adaptive algorithm for finding a quorum in the *Paths* quorum system which is tight in that respect. We further show an adaptive algorithm for *Paths* with probe complexity

$O(\sqrt{n})$, which is optimal (up to constants). Thus combined with the results in [28] the Paths system is the first quorum system shown to have an excellent balance between many somewhat contradictory measures of quality. In the second part of the paper (Section 4) we present and analyze *Dynamic Paths*, a construction for a dynamic and scalable quorum system which could be viewed as a dynamic adaptation of the *Paths* system. To the best of our knowledge *Dynamic Paths* is the first scalable quorum system which is shown to have low load, high availability and good probe complexity. Thus it is an excellent candidate for an implementation of quorums in a dynamic distributed network.

2 Non Adaptive Algorithms vs. Load

A non adaptive algorithm for finding a live quorum is an algorithm which decides which elements to probe *before* it gains any knowledge as to which elements failed and which did not. Non-adaptive algorithms are important in the context of a distributed network since they are easy to implement in parallel. It might be worthwhile to 'pay' in a higher message complexity, and reduce the total time complexity of the algorithm. As an illustrative example consider a quorum system in which only $\sqrt{n}$ elements participate in quorum sets. Clearly querying only those $\sqrt{n}$ elements is sufficient to find a live quorum. The drawback of this approach is that the load on these elements would be high (Lemma 2 implies that it would be at least $n^{-\frac{1}{4}}$). In this section we show a tradeoff between the load of a quorum system and its probe complexity for non adaptive algorithms.

Theorem 3. *Let $\mathcal{S}$ be a quorum system over universe U with a load of $\zeta = \zeta(\mathcal{S})$. Assume that each element in U fails with some fixed probability $p < \frac{1}{2}$. Let $X \subseteq U$ be a predefined set of elements such that*

$$\Pr[X \textit{ contains a live quorum}] \geq \frac{1}{2}\,,$$

then

$$|X| \geq \frac{1}{2\log(1/p)+1} \cdot \frac{\log(1/4\zeta)}{\zeta}.$$

In particular if $\zeta(\mathcal{S})$ is $O(\frac{1}{\sqrt{n}})$ then, $|X|$ is $\Omega(\sqrt{n}\log n)$.

Proof. Let $\mathcal{S}_\mathcal{X}$ be all the quorum sets contained in X, i.e., $\mathcal{S}_\mathcal{X} = \{S | S \in \mathcal{S} \wedge S \subseteq X\}$. Let $\mathcal{R}$ be all the sets which are an intersection of X with a quorum; i.e.,

$$\mathcal{R} = \{R | R = S \cap X, S \in \mathcal{S}\}.$$

By the intersection property each set $R \in \mathcal{R}$ intersects all the sets in $\mathcal{S}_\mathcal{X}$. Therefore, if for a set $R \in \mathcal{R}$ *all* elements in R fail then X does not contain a live quorum. We show that $\mathcal{R}$ must contain many *disjoint* sets of small cardinality. Let f be a distribution over quorum sets which imposes the optimal load ζ, and let $a = |X|\zeta$. Distribution f induces a marginal distribution over the sets $R \in \mathcal{R}$ by taking $S \cap X$ for each sampled set S. Under this distribution, the expected size of R is at most a (i.e., $\mathbb{E}_f[|R|] \leq a$), otherwise the load on the elements of X would be higher than ζ. By Markov's inequality we have that with probability at least $\frac{1}{2}$ the sampled set is of size at most $2a$, so we have

$$\sum_{R:|R|\leq 2a} \Pr_f[R \text{ is sampled}] \geq \frac{1}{2} \tag{1}$$

On the other hand since the load induced by f is at most ζ we have

$$\forall x \in X \quad \sum_{R:x\in R} \Pr_f[R \text{ is sampled}] \leq \zeta \tag{2}$$

Next we show that inequalities (1) and (2) imply that $\mathcal{R}$ contains a collections $\mathcal{R}'$ of at least $\frac{1}{2\zeta}$ disjoint sets of size at most $2a$. To see this employ the following procedure: pick a set $Q \in \mathcal{R}$ such that $|Q| \leq 2a$ and put Q in $\mathcal{R}'$. Define $\mathcal{R}_Q \subset \mathcal{R}$ to be all the small sets in $\mathcal{R}$ which intersect Q, i.e., $\mathcal{R}_Q = \{R \in \mathcal{R} | R \cap Q \neq \varnothing \wedge |R| \leq 2a\}$. Now since Q has at most $2a$ elements then by inequality (2) we have

$$\sum_{R\in\mathcal{R}_Q} \Pr_f[R \text{ is sampled}] \leq 2a\zeta \tag{3}$$

Remove the sets $\mathcal{R}_Q$ from $\mathcal{R}$ and repeat the procedure by picking another set Q, until all the sets of cardinality $\leq 2a$ were removed. By inequalities (1) and (3) we can perform this procedure $\frac{1}{4a\zeta}$ times. Clearly all the sets Q chosen in this process are disjoint and of small cardinality. For each set $Q \in \mathcal{R}'$ the probability that all its elements fail is at least p^{2a}. Since the sets are disjoint, these events are mutually independent. In order for the probability of finding a live quorum to be at least $\frac{1}{2}$ we must have then that

$$\begin{aligned}
&(1-p^{2a})^{\frac{1}{4a\zeta}} \geq \frac{1}{2} \\
&\exp\left(-\frac{p^{2a}}{4a\zeta}\right) \geq e^{-1} \\
&2a \cdot \log(1/p) + \log a \geq \log(1/4\zeta) \\
&a \geq \frac{\log(1/4\zeta)}{2\log(1/p)+1}
\end{aligned}$$

Now since $|X| = \frac{a}{\zeta}$ this implies the theorem. □

Theorem 3 lower bounds the probe complexity of *non-adaptive* algorithms. The smaller the load is, the larger the probe complexity is. The Paths system has a load of $\Theta(\frac{1}{\sqrt{n}})$. The theorem implies that any non-adaptive algorithm would have to probe a predefined set of $\Theta(\sqrt{n}\log n)$ processors, in order to succeed with probability $\frac{1}{2}$. If the load is very large, say constant, then the bound given by Theorem 3 is $\Omega(1)$. In case of the Majority system, the bound is much worst than the trivial lower bound of $\frac{n}{2}$. This however is unavoidable since quorum systems with high load may have quorum sets of small cardinality, so any bound which uses the load alone will deteriorate when the load increases.

3 The Paths Quorum System

We recall the construction of the *Paths* system from [28]. We start with a precise definition of the grid we will be using.

Definition 4. *Let $G(\ell)$ be the subgrid of $\mathbb{Z}^2$ with vertex set $\{(v_1, v_2) \in \mathbb{Z}^2 : 0 \leq v_1 \leq \ell+1, 0 \leq v_2 \leq \ell\}$ and edge set consisting of all edges joining neighboring vertices except those joining vertices u, v with either $u_1 = v_1 = 0$ or $u_1 = v_1 = \ell+1$.*

Definition 5. *Let $G^*(\ell)$, the* dual *of $G(\ell)$ be the subgrid with vertex set $\{(v_1, v_2) + (\frac{1}{2}, \frac{1}{2}) : 0 \leq v_1 \leq \ell, -1 \leq v_2 \leq \ell\}$ and edge set consisting of all edges joining neighboring vertices except those joining vertices u, v with either $u_2 = v_2 = -\frac{1}{2}$ or $u_2 = v_2 = \ell + \frac{1}{2}$.*

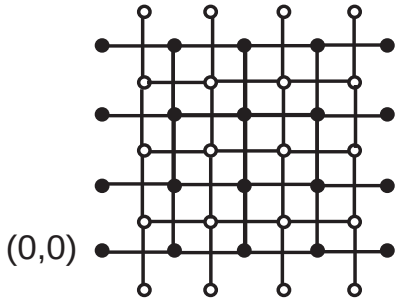

Figure 1: The grids $G(3)$ (thick lines) and $G^*(3)$ (thin lines).

Note that every edge $e \in G(\ell)$ has a dual edge $e^* \in G^*(\ell)$ which *crosses* it. We call such e and e^* a *dual pair of edges*. Note also that $G(\ell)$ and $G^*(\ell)$ are isomorphic. Both $G(\ell)$ and $G^*(\ell)$ contain $\ell^2 + (\ell+1)^2 = 2\ell^2 + 2\ell + 1$ edges.

Definition 6. *The* Paths quorum system *of order ℓ has $n = 2\ell^2+2\ell+1$ elements, and we identify an element in U with a dual pair of edges $e \in G(\ell)$ and $e^* \in G^*(\ell)$. A quorum in the system is a set of elements which contains (elements identified with) the edges of a left-right path in $G(\ell)$ and the edges of a top-bottom path in $G^*(\ell)$.*

The Paths quorum system of order 3 is depicted in Figure 1. The intersection property of the quorum system follows from the following fact:

Fact 7. *Every left-right path in $G(\ell)$ crosses every top-bottom path in $G^*(\ell)$.*

Naor and Wool proved that the load of the Paths quorum system is at most $\frac{2\sqrt{2}}{\sqrt{n}}$ (where $\frac{1}{\sqrt{n}}$ is best possible). Furthermore it is shown that if each processor fails with probability smaller than half, then the probability a live quorum exists is at least $1 - e^{-\Omega(\sqrt{n})}$.

3.1 Algorithmic Probe Complexity

The analysis of the algorithms we present is based on Theorem 8 bellow due to Menshikov [24] from Percolation Theory. Consider the infinite two dimensional grid $\mathbb{Z}_2$ and fix a vertex u. Define $S(k)$ to be the ball of radius k with u at its center, where the distance k is taken according to the grid L_1 metric. The set $\vartheta S(k)$ consists of the vertices in the boundary of the ball. Assume each edge fails with some fixed probability $p > \frac{1}{2}$. Note that since the failure probability is greater than $\frac{1}{2}$, we discuss the case in which *most* edges fail. Define A_k to be the event that there is a path of surviving edges between u and some vertex in $\vartheta S(k)$. The following is Menshikov's Theorem. A good reference for its proof could be found in Grimmett's book [12].

Theorem 8. *Let $\frac{1}{2} < p \leq 1$ be some failure probability, and let A_k be defined as above. There exists some positive constant $\psi(p)$ such that $\Pr[A_k] < e^{-\psi(p)k}$ for all k.*

Let $\overline{G}(\ell)$ be the dual grid of $G(\ell)$ (just like $G^*(\ell)$), however if an edge in $G(\ell)$ survives then its dual edge in $\overline{G}(\ell)$ fails and if an edge in $G(\ell)$ fails then its dual edge in $\overline{G}(\ell)$ survives. The graph $\overline{G}(\ell)$ is used for the analysis and is not a part of the construction itself. Now, since the failure probability in $G(\ell)$ is smaller than $\frac{1}{2}$, the failure probability in $\overline{G}(\ell)$ is greater than $\frac{1}{2}$, and we can use Theorem 8. Theorem 8

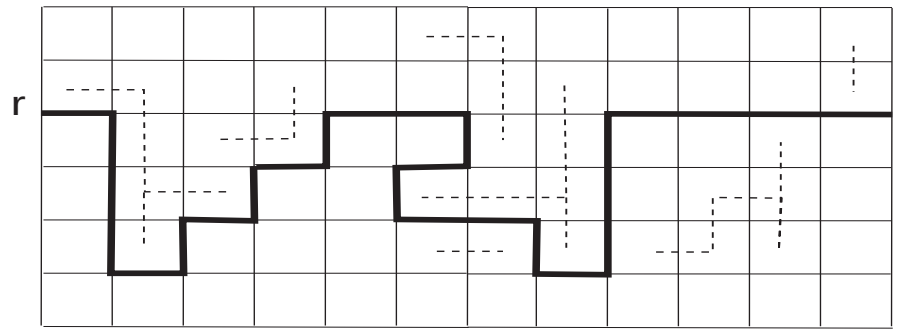

Figure 2: The dashed lines indicate the duals of failed edges. The bold line indicates a left-right path.

bounds the radius of a connected component of $\overline{G}(\ell)$. It states that the radius of a connected component has an exponential decay.

Corollary 9. *If each edge of $G(\ell)$ fails with probability $p < \frac{1}{2}$, there exists some constant $\delta = \delta(p)$ such that with high probability*[1] *every connected component of $\overline{G}(\ell)$ is contained in some ball of radius $\delta \log n$ (where the balls are defined by the metric of the grid before failures).*

Proof. Theorem 8 states that the probability that a ball of radius $\delta \log n$ centered at vertex u does not contain the component of u in $\overline{G}(\ell)$ is less than $e^{-\psi(p)\delta \log n}$. Set δ such that $\delta \cdot \psi(p) \geq 2$. Now for each vertex u this probability is less than $\frac{1}{n^2}$. When applying the union bound over all the n vertices we have: $\Pr[$ All components are contained in balls of radius $\leq \delta \log n]$ is at least $1 - \frac{1}{n}$.

□

3.1.1 A Non Adaptive Algorithm.

We show an algorithm that matches the lower bound of $\sqrt{n} \log n$ for non adaptive probes from Theorem 3. A left-right path in $G(\ell)$ must *avoid* all the components of surviving edges in $\overline{G}(\ell)$. See Figure 2. We describe a non-adaptive algorithm that finds a left-right path, when each element fails with probability $p < \frac{1}{2}$. The case of a top-bottom path is analogous. Choose a horizontal strip of width at least $2\delta \log n + 1$ (where δ is taken from Corollary 9) and examine all the edges. The algorithm tries to find a left-right crossing within the boundaries of this strip.

Claim 10. *If each element in the quorum system fails independently with probability $p < \frac{1}{2}$, then after probing non-adaptively $2\ell(2\delta \log n + 1) = \Theta(\sqrt{n} \log n)$ elements, the algorithm finds a quorum with high probability.*

Proof. Corollary 9 implies that there is no path in $\overline{G}(\ell)$ that crosses the strip top to bottom (otherwise this path is part of a component which can not be contained in a $\delta \log n$ radius ball). By Fact 7 this implies a left-right path in the strip. See Figure 2. □

Note that while the we showed that probing $O(\sqrt{n} \log n)$ elements is sufficient to succeed with high probability, Theorem 3 states that $\Omega(\sqrt{n} \log n)$ probes are necessary to succeed with merely probability $\frac{1}{2}$. As mentioned, since the algorithm is non adaptive it could be implemented in parallel. The actual running time of the algorithm depends on the implementation of the network.

[1]The term 'with high probability' (w.h.p) means with probability $1 - n^{-\epsilon}$ where ϵ is some positive constant.

The Load After Failures Naor and Wool show in [28] (Proposition 5.8) that the load of the Paths system is $\Theta(\frac{1}{\sqrt{n}})$ even after failures. In this section we present an efficient non-adaptive algorithm for picking a quorum which meets this bound w.h.p.

Lemma 11. *If each edge fails with probability $p < \frac{1}{2}$, then there exists a positive constant $\alpha = \alpha(p)$ such that in every strip of width $\alpha \log n$ there exists $\log n$ left-right paths that are edge disjoint.*

Proof. Denote by LR the event that there exists a left-right path in a strip of width $\alpha \log n$ (the constant α will be fixed later). Denote by LR_r the event that there are r *edge disjoint* left-right paths in the strip. Fix some p' such that $p < p' < \frac{1}{2}$. Proposition 5.8 in [28] uses a known result from percolation theory [6] in order to show the following:

$$\Pr_p(LR_r) \geq 1 - \left(\frac{1-p}{p'-p}\right)^r (1 - \Pr_{p'}(LR))$$

When $r = \log n$ we have that $\left(\frac{1-p}{p'-p}\right)^r$ is $O(n^k)$ for some constant k. Since $p' < \frac{1}{2}$, by Corollary 9 and Claim 10 we can choose α to be large enough so that $1 - \Pr_{p'}(LR) < n^{-(k+1)}$ and the Lemma follows. □

The strategy of picking a quorum is the following: First pick at *random* a strip and probe all its elements. Find the edge disjoint left-right paths and pick at random one of these paths.

Corollary 12. *If $p < \frac{1}{2}$ then the load imposed on the elements by the strategy described above is $\Theta(\frac{1}{\sqrt{n}})$.*

Proof. Given a node u, the probability that node u belongs to the randomly chosen strip is $\Theta(\frac{\log n}{\sqrt{n}})$. Lemma 11 implies that given that u is in the strip, the probability it belongs to the chosen quorum set is at most $\Theta(\frac{1}{\log n})$. As the second event is conditioned upon the first, we may multiply the probabilities and deduce that the load imposed by the strategy is $\Theta(\frac{1}{\sqrt{n}})$ even after failures. □

3.1.2 An Adaptive Algorithm for Paths

Adaptive algorithms can do better than non adaptive ones. Hassin and Peleg presented in [14] a lower bound of $\frac{c}{1-p} + O(1)$ on the expected probe complexity, when c is the cardinality of the smallest quorum set. They proved that for some quorum systems this bound is tight. We note that it is proved in [28] that $\zeta(\mathcal{S}) \geq \max\{\frac{1}{c}, \frac{c}{n}\}$ therefore this lower bound is at best linear in the inverse of the load. In the following we present an adaptive algorithm for the Paths quorum system which needs only $\Theta(\sqrt{n})$ probes. It is optimal in the sense that every quorum system with optimal load must have $c = \Omega(\sqrt{n})$. Various adaptive algorithms for quorums were analyzed by Hassin and Peleg [14]. The only quorum system with a better probe complexity is the Crumbling Walls system. This system however suffers from high load. Bazzi presented in [7] the Triangle Lattice quorum system, which resembles Paths, and an adaptive algorithm for finding quorums in case of failures. Bazzi proves that its cost of failures (i.e., the communication overhead due to encountering a failed processor) is constant. Our algorithm, is an adaptation of Bazzi's algorithm to the grid, and therefore Bazzi's analysis applies in our case and shows the following: let Q be the set of processors probed by the algorithm until a quorum was found. Let $F \subset Q$ be the subset of which that failed, then there exists a constant α such that $|F| \leq \frac{|Q|}{\alpha}$. This result does not bound the *total number* of probed processors. Our goal in this section is to show that *with high probability*, the total number of processors probed is $O(\sqrt{n})$.

We start with a formal description of the algorithm, as before it is sufficient to show how to find a left-right path. The algorithm is a variant of a DFS-search with a specified strategy for picking the next edge to

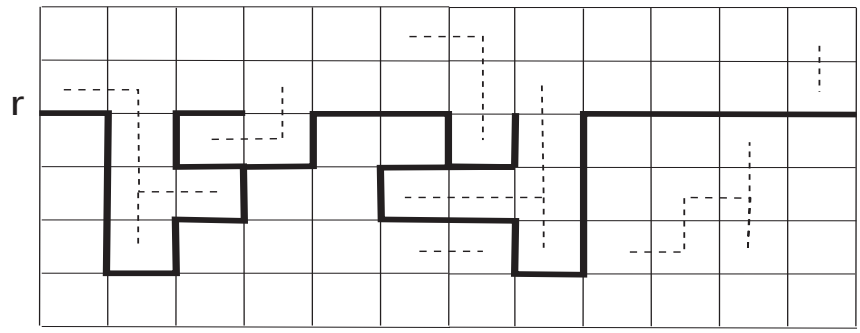

Figure 3: The dashed lines indicate the duals of failed edges. The bold line indicates a possible path taken by the adaptive algorithm.

probe. We say that a path *circumvents* a component of $\overline{G}(\ell)$ if it travels along the surface of it; i.e, it travels along edges, the duals of which are adjacent to the component and not part of it. The algorithm would try to find a path which is a straight left-right line. Whenever a component of $\overline{G}(\ell)$ is encountered it would be circumvented. More formally:

1. Choose r at random $1 \leq r \leq \ell$. The search begins at edge r of the left column, and aims to travel along row r.

2. Go to the right. When a failed edge is encountered *circumvent* the component until row r is reached again. If $r \leq \frac{1}{2}\ell$ from above, otherwise from below. Return to row r.

The path in bold presented in Figure 3 demonstrates a possible path of the DFS search. The path taken by the algorithm needs to circumvent a component of $\overline{G}(\ell)$ only if it contains the dual of an edge in row r. For each failed edge e of row r define C_e to be the number of edges in the component of $\overline{G}(\ell)$ that contains the dual of e. If e did not fail then $C_e = 0$. The number C_e is an upper bound on the length of the circumvention the path had to take in order to avoid the failed edge e.

Observation 13. *The length of the path taken by the algorithm is at most $\ell + \sum C_e$ where the sum is taken over the edges of row r.*

Theorem 14. *The probe complexity of the algorithm is $\Theta(\ell) = \Theta(\sqrt{n})$ with high probability (where the probability is taken over the occurrence of faults).*

Proof. Assume that the random starting point selected in Step (1) of the algorithm is a starting point of some left-right path of the grid. By Lemma 11 we know that the probability of this is constant. Thus we repeat the procedure above, until a good starting point is found. We need to show that all the circumventions taken in Step (2), i.e., $\sum C_e$, accumulate to no more than $\Theta(\ell)$. Fix some edge e on row r, and let vertex u belong to its dual edge. Let C_u be the number of edges in the component of u in $\overline{G}(\ell)$. Let A_u denote the diameter of that component. Since the vertex u is adjacent to the dual of e it holds that $C_u \geq C_e$. The grid topology implies that if $C_u \geq k$ then $A_u \geq \frac{1}{2}\sqrt{k}$. We have (by Theorem 8)

$$\Pr[C_u \geq k] \leq \Pr[A_u \geq \tfrac{1}{2}\sqrt{k}] \leq e^{-\psi(p)\sqrt{k}} \quad \text{for some } \psi(p) > 0 \tag{4}$$

$$E[C_u] = \mu \leq \sum_{k=1}^{\infty} k \cdot e^{-\psi(p)\sqrt{k}} = O(1) \tag{5}$$

The algorithm may need to avoid at most ℓ components of $\overline{G}(\ell)$. By linearity of expectation the expected probe complexity of the algorithm is $\Theta(\ell)$. To show that this sum is $\Theta(\ell)$ with high probability we need a slightly different argument. Divide the grid into $(\frac{\ell}{\delta \log n})$ vertical strips each of width $\delta \log n$, where δ is taken from Corollary 9. Each strip is wide enough such that w.h.p it is wider than any component of $\overline{G}(\ell)$. Assume this high probability event occurs. Define $X_i \stackrel{def}{=} \sum C_e$ where the sum is taken over edges of row r and strip i. The length of the path the algorithm took is at most $\ell + \sum X_i$.

Lemma 15. *$E[X_i] \leq \mu\delta \log n$ and w.h.p for all i we have $X_i \leq 2\delta^2 \log^2 n$.*

Proof. The width of the strip is $\delta \log n$ and the expected size of each component is μ, therefore by linearity of expectation $E[X_i] \leq \mu\delta \log n$. By Corollary 9 we know that w.h.p all the components are contained in a $\delta \log n$ radius ball. Therefore w.h.p all the components are confined into a rectangle of area $2\delta^2 \log^2 n$. which proves the second claim. □

Define $I_\sigma = \{1 \leq i \leq \frac{\ell}{\delta \log n} : i \mod 3 = \sigma\}, \sigma \in \{0, 1, 2\}$.

Lemma 16. *Conditioned on the event that all the components are of diameter $O(\log n)$, which by Corollary 9 occurs with high probability, the set $\{X_i\}_{i \in I_\sigma}$ consists of independent random variables.*

Proof. If all components are of small diameter, then every connected component of $\overline{G}(\ell)$ belongs to at most two strips. Therefore X_i depends only upon the probes of edges in strips $i-1, i, i+1$. This means that X_i, X_{i+3} are independent. □

By using the appropriate Chernoff Hoefding bound (cf. [11] page 17) we have

$$\Pr\left[\sum_{I_\sigma} X_i - \sum_{I_\sigma} E[X_i] \geq t|I_\sigma|\right] \leq 2\exp\left(-\frac{2t^2|I_\sigma|}{(2\delta^2 \log^2 n)^2}\right)$$

Since $|I_\sigma|$ is in the order of $\frac{\sqrt{n}}{\log n}$, the probability that there is a large deviation decays exponentially fast. In particular setting t to be some large enough constant implies that $\Pr[\sum_{I_\sigma} X_i > \Theta(\ell)] \leq \frac{1}{n^2}$ for $\sigma \in \{0, 1, 2\}$. Now we apply the union bound over the high probability events of Corollary 9 and Lemmas 15,16, which means that with high probability the probe complexity of the algorithm is $\Theta(\ell) = \Theta(\sqrt{n})$. This concludes the proof of Theorem 14. □

Network implementation In order to calculate the *actual* running time and message complexity of these algorithms we need to take into consideration the topology and implementation of the network over which the quorum system is defined. The most natural network topology to consider is that of $G(\ell), G^*(\ell)$ themselves. Each processor is associated with a pair of dual edges, and is connected to the processors that are associated with edges that are adjacent to its own edges. In other words, the topology of the network is the *line graph* of the two dimensional grid. In Figure 4 the thick solid edges belong to the line graph of $G(\ell)$, the dotted edges belong to the line graph of $G^*(\ell)$ and the diagonal edges belong to both. A quorum set therefore is composed of processors (nodes) that form a left-right path using the solid horizontal, vertical and diagonal edges and a left-right path using the dotted and diagonal lines. In this implementation the message complexity and the time complexity of the adaptive algorithm are indeed $\Theta(\ell)$. The non-adaptive algorithm can probe its chosen strip in parallel, and achieve a running time of $\Theta(\ell)$ and a message complexity of $\Theta(\sqrt{n} \log n)$. Other data structures that are implemented on the network might support the implementation of the quorum system. For instance if the network implements a DHT (such as the one presented in [26]) then the DHT could be used for probing the strip in parallel and the time complexity would reduce to $\Theta(\log n)$ with an extra logarithmic factor in the message complexity.

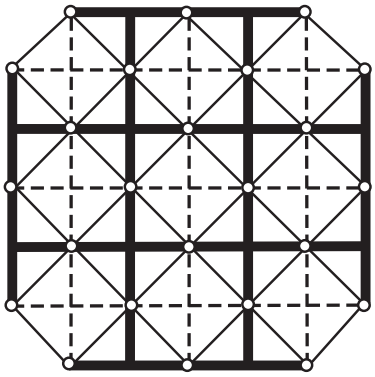

Figure 4: The line graph of a 4×4 grid and its dual.

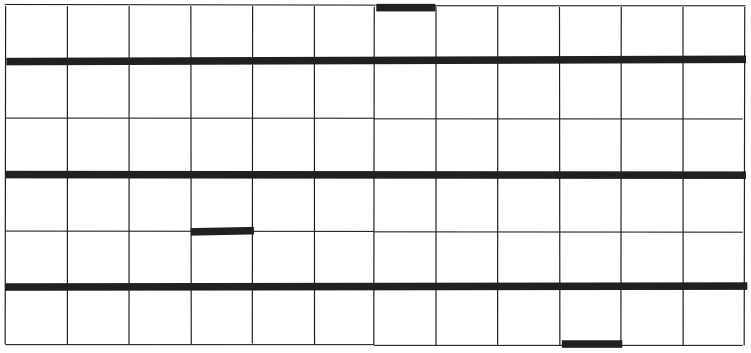

Figure 5: A possible sample from the distribution over inputs. The bolded edges are closed. The third row is an open path.

Worst case model Assume an adversary is given the possibility to crash a constant fraction of the elements. It is easy to see that an adversary can 'kill' all the short paths, and leave only paths of length $\Omega(\ell^2) = \Omega(n)$. However an adversary may force any algorithm (even probabilistic) to probe $\Omega(n)$ elements, even if we are guaranteed that there exists a short left-right path. We sketch the proof using Yao's minimax principle (cf. [25]). We need to supply a distribution of the *inputs* such that every deterministic algorithm would need to probe an expected $\Omega(n)$ elements. The distribution over inputs is as follows:

1. Kill every line of even index.
2. From the remaining lines choose at random one which would remain alive.
3. Kill each remaining line by choosing at random one element from it and deleting it.

An example of a possible input is seen in Figure 5, where the third row from the top is the only surviving row. Now every *deterministic* algorithm needs to find the line that survived. Every such algorithm will need to probe $\Omega(\ell)$ lines, each of these lines should be probed $\Omega(\ell)$ times. All in all every deterministic algorithm would probe on expectation $\Omega(\ell^2)$ edges. We conclude that for every algorithm (deterministic or randomized) there is an input, for which the expected probe complexity of the algorithm is $\Omega(n)$. Peleg and Wool analyze in [30] the probe complexity of several quorums under the model of adversarial deletion.

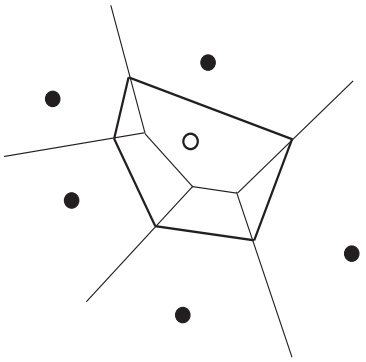

Figure 6: Addition of a new generator.

They show several lower bounds, all of which turn to be $\Omega(\ell)$ in the Paths system. Note that even though the algorithmic probe complexity is high, the *cost of failures* (as were defining by Bazzi [7]) is a constant.

4 The Dynamic Paths Quorum System

In this section we suggest a quorum system that operates in a dynamic model, where processors may join and leave. The applications of quorum systems in a dynamic setting were considered in a wide range of papers cf. [1],[15],[17]. Previous constructions of dynamic quorums focused on *implementing* quorum systems in a dynamic environment and designing algorithms that allowed a group of processors to *form* a new quorum in a consistent way (cf. [16],[19],[31], [13],[20]). We focus on the *combinatorial* properties of dynamic quorums. Our goal is to design dynamic quorums that enjoy low load, high availability, low probe complexity and that scale gracefully in respect to these parameters. The good properties of the Paths quorum system motivates us to design a *dynamic version* of the Paths system. The main idea is to substitute the grid with the *continuous* unit square $[0,1) \times [0,1) \subset \mathbb{R}^2$. The unit square is then decomposed into cells, where each processor is associated with a cell. The entrance and exit of a processor dynamically changes the decomposition. The decomposition of the square into the cells is done via Voronoi Diagrams. Our technique is similar to the one presented in [26] for building DHT's. Stojmenović and Peña [35] suggest a location based dynamic quorum system for use in ad-hoc wireless networks. The system is composed of North-South and West-East paths which are constructed dynamically according to the physical location of the nodes. Our work differs by assigning *virtual* coordinates to processors, thus using the quorum system in a more general setting. We then provide a rigorous analysis in which the combinatorial properties (load, availability, integrity) of the system are analyzed.

4.1 Dynamic Voronoi Diagrams

Definition 17 (planar ordinary Voronoi diagram). *Given a finite number (at least* 2*) of distinct points in the Euclidean plane, we associate all locations in that space with the closest member(s) of the point set with respect to the Euclidean distance. The result is a tessellation of the plane into a set of regions associated with members of the point set. We call this tessellation the* planar ordinary Voronoi diagram *generated by the point set, the points are sometimes referred to as* generators *and the regions constituting the Voronoi diagram* Voronoi cells*. The dual triangulated graph is called the* Delaunay triangulation*.*

See Okabe *et al* [29] for a thorough overview of Voronoi diagrams and their applications. Given an

existing Voronoi diagram, the entrance of a new generator and the exit of an existing one affects only the cells adjacent to the location of the generator. Therefore a Voronoi diagram can be maintained by a distributed algorithm, in which every cell is calculated separately and *locally*. The time and memory needed to compute a single Voronoi cell is $\Theta(d)$ when d is the number of neighbors the cell has; i.e., the degree of the generator in the Delaunay tessellation. See Figure 6 for a demonstration of an insertion of a new generator. It is well known that the average degree of a Voronoi cell is 6. It follows that if the generators of a Voronoi diagram are entered in random order, then the average of d is at most 6 as well. In the worst case d might be as high as $n-1$.

4.1.1 The Join/Leave operations

Processors are associated with generators of a Voronoi diagram. Each processor holds its own location on the plane and the location of its neighbors in the Delaunay triangulation. A processor that wishes to join the system does the following:

1. Choose a location x in the unit square (typically x would be chosen randomly and uniformly from $[0,1) \times [0,1)$).
2. Find the processor whose cell contains x. Learn the location of its neighbors.
3. Calculate the boundaries of the new Voronoi cell and inform the neighbors so that they can update their tables.

Before analyzing the algorithm we show the properties of a Voronoi diagram in which the location of each generator was chosen randomly and uniformly. We show that with high probably the Voronoi diagram decomposes the square into more or less equal cells.

Theorem 18. *If the location of each generator of the Voronoi diagram was chosen uniformly and randomly in* $[0,1) \times [0,1)$ *then with high probability the following holds:*

1. *The area of the largest Voronoi cell is at most* $O(\frac{\log n}{n})$.
2. *The number of neighbors each Voronoi cell (the maximum degree of the Delaunay graph) is* $O(\log n)$.
3. *The projection of each Voronoi cell on the axis lines is at most* $O(\sqrt{\log n/n})$.

Proof. Divide the square into $\frac{n}{\log n}$ squares of size $\sqrt{\frac{\log n}{n}} \times \sqrt{\frac{\log n}{n}}$. Now model the process as putting n balls in $\frac{n}{\lg n}$ bins. It is well known that when n balls are put uniformly at random into $\frac{n}{\log n}$ bins, then w.h.p every bin contains $\Theta(\log n)$ balls. Assume this high probability event occurs and each small square contains $\Theta(\log n)$ balls. Fix a generator x_i. A simple geometric argument demonstrated in Figure 7 shows that all the neighbors of x_i must lie within the 25 squares that compose the 5×5 grid which surrounds the square of x_i. This asserts claims $(1), (3)$. Since each square contains $O(\log n)$ generators the number of neighbors of x_i is also bounded by $O(\log n)$. □

Since the computation of a Voronoi cell is a local operation, Step (3) of the Join algorithm takes $O(d)$ time and memory, where d is the degree of the Voronoi cell in the Delaunay graph. The average degree is 6 and Theorem 18 assures that w.h.p all degrees are at most $O(\log n)$. Step (2) of the the algorithms requires locating the processor whose cell contains the point x. The complexity of Step (2) depends upon the topology of the network and the search options it provides. If the topology of the network is that of the Delaunay graph, then the processor holding x could be found by a greedy algorithm along the geometry of the Voronoi diagram; i.e., the query moves along the Delaunay edges in a greedy way to the direction of x.

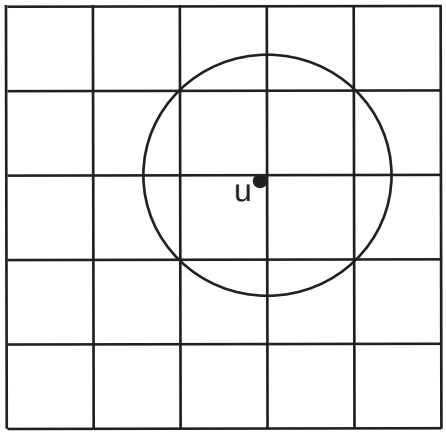

Figure 7: If each square contains at least one vertex, then the cell generated by u is contained in the circle.

Thus the time complexity and the message complexity of Step (2) are $O(\sqrt{n})$. A similar approach is taken in CAN [32]. Additional structure of the network may reduce the complexity of Step (2). The Distance Halving DHT suggested in ([26]) is implemented using the same Voronoi diagram and therefore requires low overhead. Using it Step (2) could be performed in $O(\log n)$ time and $O(\log n)$ messages. The interface of a DHT allows searching for a processor whose cell contains a certain point, without knowing a-priori the processor's i.d.

The Leave operation is done similarly. When a processors wishes to leave the system, it informs its neighbors which in turn divide and redistribute the area of its cell among themselves.

4.2 The Quorum System

In the Dynamic Paths quorum system, a quorum set is the union of (elements identified with) the vertices (generators) that form a left-right path and a top-bottom path in the Delaunay graph.

Load We upper bound the load by analysing a specific distribution over quorum sets: choose at random two points (x, y) in the interval $[0, 1)$. Now the pick the quorum set that is composed from all the cells that intersect the horizontal line x and the vertical line y. An example of a quorum set is depicted in Figure 10. The bound on the projection of a cell in Theorem 18 implies that with high probability the load imposed by this strategy is at most $\Theta(\frac{\sqrt{\log n}}{\sqrt{n}})$.

Availability Basic results in percolation theory imply that if the locations of the generators were picked uniformly at random, and the failure probability is strictly less than half, then with probability that tends to 1 (as $n \to \infty$) there exists a left-right path. This could be proved using planar duality in the same manner in which the critical probability of edge percolation is proven. A rough outline of the argument is as follows: If there is no left-right crossing, then there must be a top-bottom crossing of *failed* Voronoi cells. The procedure of creating the Voronoi diagram is symmetric and imposes the same probability over a top-bottom and a left-right crossing. Therefore if the failure probability is less than $\frac{1}{2}$ we expect a crossing to exist. Currently an analysis of the actual probability of the crossing, (i.e., the rate in which the probability of a crossing converges to 1) is unknown.

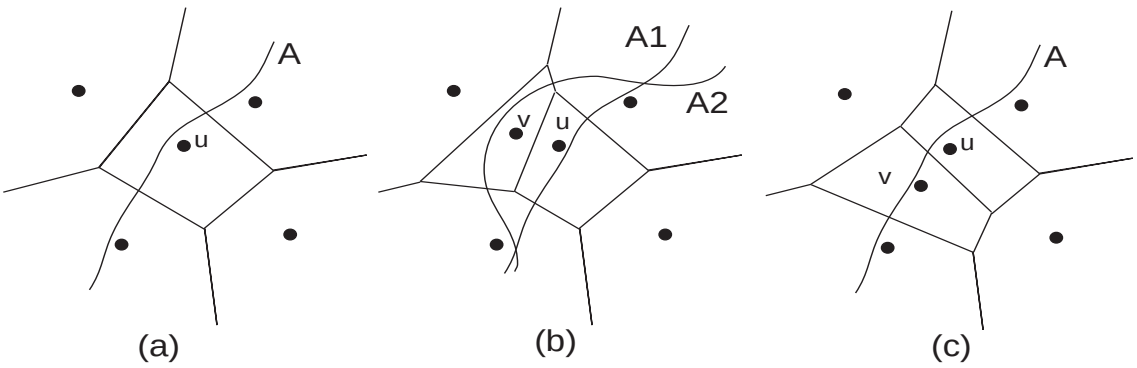

Figure 8: When adding a processor either the load of quorum A is split between quorums A_1, A_2 as seen in (b), or A grows, as seen in (c).

Integrity It is necessary that processors save some information about the quorum sets that were used. A quorum set is associated with a path. Every time a quorum is used, a processor that participates in the quorum should remember the identity of the processors before and after it in the path. When a new processor joins the system either the quorum set grows or the load should be divided evenly between the new quorums. Figure 8 demonstrates the process. Figure (a) shows the Voronoi diagram before the entrance of v. Figure (c) demonstrates the case where v is added to quorum A. Figure (b) shows the case where the responsibilities of quorum A (represented by the line in bold) should now be split between quorums A_1, A_2. If for instance the application of the quorum system is mutual exclusion, and quorum A is currently active, then processors u, v should decide among themselves which one of them remains active, and inform their neighbors. If the quorum system is used for replication of data, then the procedure is slightly more delicate. Each data item is associated with a quorum set. Processors u, v should divide among themselves the data items that were previously associated with quorum A, and of course inform their neighbors.

Algorithmic probe complexity The algorithms described in Section 3 have obvious analogs in the Dynamic Paths system. In order to prove that the probe complexity of the non-adaptive and adaptive algorithms is $\Theta(\sqrt{n}\log n), \Theta(\sqrt{n})$ respectively we need an analog for Theorem 8; i.e., we need that for a small failure probability, the radius of a component of *failed* cells would decay in sub-exponential rate. Unfortunately such a theorem is yet unknown, yet prominent researchers in the field (e.g. [9]) conjecture that it is true. If indeed the conjecture is true then the performance of the algorithms could be analyzed in the same manner as in Section 3 and the Dynamic Paths quorum system enjoys excellent probe complexity.

4.3 A Balanced Voronoi Diagram

The reason some of the parameters of *Dynamic Paths* are not as good as the parameters of *Paths* is that when each processor chooses its location randomly, some of the Voronoi cells are quite big. The load of the system is proportional to the size of the projection of the cells over the axis lines. Theorem 18 bounds the size of the projection (and therefore the load) by $O(\sqrt{\frac{\log n}{n}})$, in the case where the location of the processors is chosen uniformly and randomly in $[0,1) \times [0,1)$. Furthermore the existence of large cells makes the analysis of the availability and probe complexity of the system very difficult. A more sophisticated and coordinated procedure for choosing the location upon entrance may reduce the size of the largest cell and create a *balanced* Voronoi diagram. A *balanced Voronoi Diagram* is a diagram in which every Voronoi cell

is contained in a square of area $\Theta(\frac{1}{n})$. One such procedure is the following: upon entrance a processor chooses at random $\log n$ points and chooses its location to be inside the largest cell it encounters. An easy alteration of Theorem 10 in [26] shows that this procedure guarantees that as long as there are no deletions each cell would be contained in a square of area $\Theta(\frac{1}{n})$. This approach however cannot deal with random or worst case deletions of processors. In order to handle deletion some sort of balancing mechanism must be introduced. Balancing mechanism for the one dimensional case were introduced in [23], [26] and [2].

In a balanced diagram the projection of each cell on the axis lines is $O(\frac{1}{\sqrt{n}})$, therefore the load of the quorum system would be optimal. Balancing the Voronoi diagram enables us to analyze the availability and probe complexity of the quorum system. As mentioned before, we conjecture that the availability and probe complexity of the quorum system based on random entrance is indeed similar to that of Paths. However if the diagram is balanced and each Voronoi cell is contained in square of area $\Theta(\frac{1}{n})$ then we can prove our claims. Intuitively if the Voronoi diagram is balanced then 'it looks like a grid' and therefore theorems that are correct for the grid should apply for the diagram. The technique we use follows this intuition, though it is rather delicate. We use domination by product measures as shown by Liggett *et al* in [18]. We need some definitions from probability theory. In the following we define the necessary definitions and sketch the idea of the proof. A good exposition of the notions we use appears in Grimmett's book[12]. The discussion below follows it.

4.3.1 Domination by Product Measures

We begin by defining stochastic domination in our context. Say we have a finite set S and a state space $\Omega = \{0,1\}^S$. The set S may be the set of edges in a two dimensional grid and Ω the set of configurations when some of the edges fail. Given $\omega_1, \omega_2 \in \Omega$ we say that $\omega_1 \leq \omega_2$ if $\forall s \in S\ \omega_1(s) \leq \omega_2(s)$. In our case $\omega_1(s) \leq \omega_2(s)$ if all the surviving edges in ω_1 have also survived in ω_2.

Given a function $f : \Omega \to \mathbb{R}$ we say that f is *increasing* if

$$\omega_1 \leq \omega_2 \ \ \Rightarrow f(\omega_1) \leq f(\omega_2).$$

For instance the function that assigns the value 1 to a configuration that contains a left-right crossing and 0 otherwise, is an *increasing function*.

Now, given two probability measures on Ω, μ and ν we shall say that μ *stochastically dominates* ν - and write $\mu \succeq \nu$ - if for any increasing function f we have $E_\mu(f) \geq E_\nu(f)$. This is a very strong condition which amounts to saying that in every possible way, μ puts more mass on bigger elements of Ω than ν does. In case that f is defined as above, it means that the probability there exists a left-right path is larger in μ than it is in ν. A canonical example for domination is the following: Assume we have a two dimensional grid. Denote by π_p the product measure with probability p, i.e., the case in which each edge fails independently with probability $1-p$. It is intuitive (though requires proof) that $\pi_{p_1} \succeq \pi_{p_2}$, when $p_1 \geq p_2$.

The analysis of *Paths* used bounds on increasing events on the *product measure* over the grid. Our approach would be to show that the process of randomly failing cells in a balanced Voronoi diagram *dominates* a product measure on the grid, thus lower bounding the probability there exists a left-right path in the Voronoi diagram.

Let T be a balanced Voronoi diagram with n generators and assume that each cell survives with probability $p > \frac{1}{2}$ and fails with probability $1-p$, independently from all other cells. Now construct a $\sqrt{n} \times \sqrt{n}$ grid called G *on top* of the Voronoi diagram, as shown in Figure 9. We say that an edge $e \in G$ failed iff it intersects a failed cell of T. Let X_e be the indicator of the state of e (i.e., $X_e = 1$ iff e survived). Now $\Pr[X_e = 1]$ is exactly p to the power of the number of cells it intersects. However since T is balanced, we know that this power is bounded by some constant, therefore there exists some $p' < p$ independent of n, such that for all $e \in G$, $\Pr[X_e = 1] \geq p'$. Assume that p was large enough such that $p' > \frac{1}{2}$.

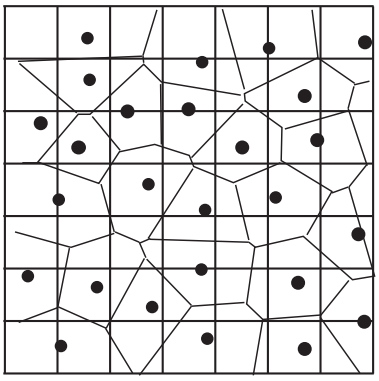

Figure 9: The grid G is put on top of the diagram T.

Observation 19. *If there exists a left-right crossing of survived edges in G then there exists a left-right crossing of survived Voronoi cells in T (i.e., a crossing in the Delaunay graph).*

Since $p' \geq \frac{1}{2}$ one is tempted to use known results from percolation theory that show that the probability of a crossing is very high, as was used in [28] to prove the availability of *Paths* and as was used in this paper to prove the low probe complexity of *Paths*. The problem is that the random variables $\{X_e\}_{e \in G}$ are *not mutually independent*. In particular, if two edges are contained in the same cell in T, then the state of both of them is determined by the state of that cell. The key observation is that since T is balanced, X_e is independent from all but *a constant number* of other edges. Let μ be the probability measure thus defined on $\{X_e\}_{e \in G}$. Liggett *et al* show in [18] that in this case μ *dominates* the product measure over the edges of G for some other value $r' \leq p'$. Theorem 1.3 in [18] could be stated in our case as follows:

Theorem 20. *Let μ be some probability measure over the set of configurations of the edges of G. Assume that each edge in G survives with probability at least p', and that the state of each edge is dependent on the state of at most k other edges for some constant k. Then there exists some r' which is a function of p', k and independent of n such that $\mu \succeq \pi_{r'}$. Furthermore by increasing p', r' could be made arbitrarily close to* 1.

Intuitively speaking Theorem 20 states that if we have a two dimensional grid, and each edge fails 'almost' independently from all other edges, then by reducing the failure probability, we may think as if each edge failed independently. Note that the existence of a left-right path in the grid is an increasing event. The diameter of a connected component in the dual graph (which is bounded in Theorem 8) is also an increasing function. Theorem 20 implies that by reducing the failure probability, we may use these theorems to bound those random variables in the balanced Voronoi diagram.

Denote by $G_\mu(p')$ the random graph induced by $\{X_e\}_{e \in G}$. Denote by $G_\pi(r')$ the random graph induced by the product measure with probability r'.

Corollary 21. *Let p' be close enough to* 1. *There exists some $r' \leq p'$ independent from n, such that the probability there exists a crossing in $G_\mu(p')$ is at least the probability there exists a crossing in $G_\pi(r')$, and the probability a component of the dual $G_\mu(p')$ is of diameter k, is at most the probability a component of the dual of $G_\pi(r')$ is diameter k. Furthermore by increasing p', r' could be made arbitrarily close to* 1.

Corollary 21 is directly used to analyze the availability and probe complexity of *Dynamic Paths*:

Theorem 22. *Let T be a balanced Voronoi diagram, and let $\mathcal{S}$ be the Dynamic Paths quorum system derived by it. Then the load of the system $\zeta(\mathcal{S})$ is $O(\frac{1}{\sqrt{n}})$. There exists some $\frac{1}{2} < p_c < 1$ such that for $p_c < p < 1$, if each processor fails independently with probability $1 - p$ then the following hold:*

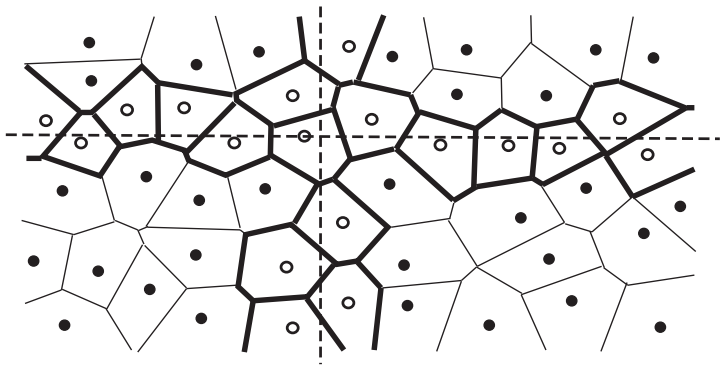

Figure 10: An example of a quorum on a Voronoi diagram. The cells that belong to the quorum set are the ones that intersect the dashed lines.

1. *The probability a live quorum set exists is* $1 - e^{-\Omega(\sqrt{n})}$
2. *The non-adaptive algorithmic probe complexity is* $O(\sqrt{n}\log n)$ *w.h.p.*
3. *The adaptive algorithmic probe complexity is* $O(\sqrt{n})$ *w.h.p.*

4.4 A simpler Quorum System:

A possible simplification of the Dynamic Paths system is the following: Define a quorum set to be all the (elements identified with) cells that intersect the same horizontal and vertical line (see Figure 10). This quorum system is a dynamic adaptation of a quorum system suggested by Maekawa [22]. A slight improvement was suggested by Agrawal *et al* in [5] where instead of looking at horizontal and vertical lines, they examine diagonal lines that resemble the paths of billiard balls. Theorem 18 implies that the load of these quorum systems is $\Theta(\sqrt{\frac{\log n}{\sqrt{n}}})$. The integrity of these systems could be maintained by associating each quorum set with the numeric value of the vertical and horizontal lines, thus the implementation is simpler. The main drawback of these systems is their low availability. If each processor fails with probability $\Theta(\frac{\log n}{\sqrt{n}})$, then with high probability no quorum set survives.

5 Conclusion and Open Questions

The main open problem is to improve the load of the *Dynamic Paths* quorum system so that it matches the load of *Paths*. The load of *Dynamic Paths* is determined by the size of the projection of cells over the axis lines. The Join algorithms as we described it guarantees that the projection of all cells is at most $O(\frac{\sqrt{\log n}}{\sqrt{n}})$. It is interesting to find other (perhaps more sophisticated) Join algorithms that guarantee a better load. A deterministic Join algorithms that guarantees excellent load in the worst case is presented in [26]. A random algorithm appears in [2]. These algorithms operate in the *one dimensional* universe, i.e when processors are located along a line. It would be interesting to find a two dimensional analog to that algorithm. Some work in this direction was done in [4], however they considered splitting the plain into rectangles (as in CAN) and not a Voronoi diagram.

A better understanding of percolation theory over Voronoi diagrams would improve the analysis of the algorithms. In particular it is important to bound the probability of a diameter k component in a percolation with $p < \frac{1}{2}$. A 'Menshikov style' theorem of this sort that states that this probability is exponentially small in

k, would imply a $\Theta(\log n\sqrt{n})$ algorithmic probe complexity for *Dyanmic Paths* even for the simple random Join algorithm.

Conclusion The Paths quorum system is shown to have excellent adaptive and non-adaptive probing algorithms. It was previously known that the Paths system has optimal load and availability, thus the Paths system offers excellent balance between different quality measures. This makes Paths a natural candidate for an adaptation into a dynamic setting. A general technique for designing scalable dynamic data structures is presented in [26]. Applying this technique results with the *Dynamic Paths* quorum system which is scalable and operates in a dynamic setting. *Dynamic Paths* maintains the good qualities of the Paths system. Its low load, high availability and simple probing algorithms makes it an excellent candidate for an implementation of dynamic quorums.

Acknowledgments

We gratefuly thank Itai Benjamini for pointing out some relevant theorems in probability and percolation theory, and Dahlia Malkhi for useful discussions. We thank the anonymous referees for many helpful suggestions and references.

References

[1] Amr El Abbadi, Dale Skeen, and Flaviu Cristian. An efficient, fault-tolerant protocal for replicated data management. In *Proceedings of the 5th ACM SIGACT/SIGMOD Conference on Principles of Database Systems*, pages 215–229, 1985.

[2] Ittai Abraham, Baruch Awerbuch, Yossi Azar, Yair Bartal, Dahlia Malkhi, and Elan Pavlov. A generic scheme for building overlay networks in adversarial scenarios. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, page 40, April 2003.

[3] Ittai Abraham and Dahlia Malkhi. Probabilistic quorums for dynamic systmes. In *Proceedings of the 17th International Symposium on Distributed Computing (DISC)*, pages 60–74, 2003.

[4] Micah Adler, Eran Halperin, Richard M. Karp, and Vijay V. Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, pages 575–584, 2003.

[5] Divyakant Agrawal, Omer Egecioglu, and Amr El Abbadi. Billiard quorums on the grid. *Information Processing Letters*, 64(1):9–16, 1997.

[6] Michael Aizenman, Jennifer Chayes, Lincoln Chayes, Jurg Frohlich, and L. Russo. On a sharp transition from area low to perimeter low in a system of random surfaces. *Comm. Mathematical Physics*, (92):19–69, 1983.

[7] Rida A. Bazzi. Planar quorums. In *Distributed Algorithms, 10th International Workshop, WDAG '96*, volume 1151 of *Lecture Notes in Computer Science*, pages 251–268, Bologna, Italy, 9–11 October 1996. Springer.

[8] Mark Bearden and Jr. Ronald P. Bianchini. A fault-tolerant algorithm for decentralized on-line quorum adaptation. In *Symposium on Fault-Tolerant Computing*, pages 262–271, 1998.

[9] Itai Benjamini. Private communication.

[10] Hector Garcia-Molina and Daniel Barbara. How to assign votes in a distributed system. *Journal of the Association for Computing Machinery*, 32(4):841–855, October 1985.

[11] Oded Goldreich. *Randomized Methods in Computation - Lecture Notes*. `http://www.wisdom.weizmann.ac.il/~oded/rnd.html`, 2001.

[12] Geoffrey Grimmett. *Percolation*. Springer-Verlag, 1989.

[13] Zygmunt J. Haas and Ben Liang. Ad hoc mobility management with uniform quorum systems. *IEEE/ACM Transactions on Networking*, 7(2):228–240, 1999.

[14] Yehuda Hassin and David Peleg. Average probe complexity in quorum systems. In *20th ACM Symposium on Principles of Distributed Computing (PODC)*, 2001.

[15] Maurice Herlihy. Dynamic quorum adjustment for partitioned data. *ACM Transactions on Database Systems (TODS)*, 12:170–194, 1987.

[16] Sushil Jajodia and David Mutchler. Dynamic voting algorithms for maintaining the consistency of a replicated database. *ACM Transactions on Database Systems*, 15(2):230–280, June 1990.

[17] Goutham Karumanchi, Srinivasan Muralidharan, and Ravi Prakash. Information dissemination in partitionable mobile ad hoc networks. In *Proceedings of IEEE Symposium on Reliable Distributed Systems*, pages 4–13, 1999.

[18] Thomas L. Liggett, Roberto H. Schonmann, and Alan M. Stacey. Domination by product measures. *The Annals of Probability*, 25(1):71–95, January 1997.

[19] Esti Yeger Lotem, Idit Keidar, and Danny Dolev. Dynamic voting for consistent primary components. In *Symposium on Principles of Distributed Computing (PODC)*, pages 63–71, 1997.

[20] Nancy Lynch and Alexander Shvartsman. Rambo: A reconfigurable atomic memory service for dynamic networks. In *Proceedings of the 16th International Symposium on Distributed Computing*, pages 173–190, 2002.

[21] Nancy A. Lynch and Alexander A. Shvartsman. Robust emulation of shared memory using dynamic quorum-acknowledged broadcasts. In *Symposium on Fault-Tolerant Computing*, pages 272–281, 1997.

[22] Mamoru Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3(2):145–159, May 1985.

[23] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *ACM Conf. on Principles of Distributed Computing (PODC)*, pages 183–192, 2002.

[24] Mikhail V. Menshikov. Coincidence of critical points in percolation problems. *Soviet Mathematics Doklady*, 33:856–859, 1986.

[25] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[26] Moni Naor and Udi Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 50–59, 2003.

[27] Moni Naor and Avishai Wool. Access control and signatures via quorum secret sharing. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):909–922, 1998.

[28] Moni Naor and Avishai Wool. The load, capacity, and availability of quorum systems. *SIAM Journal on Computing*, 27(2):423–447, 1998.

[29] Atsyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations — Concepts and Applications of Voronoi Diagrams*. Wiley, Chichester, second edition, 2000.

[30] David Peleg and Avishai Wool. How to be an efficient snoop, or the probe complexity of quorum systems. *SIAM Journal on Discrete Mathematics*, 15(3):416–433, August 2002.

[31] Roberto De Prisco, Alan Fekete, Nancy A. Lynch, and Alexander A. Shvartsman. A dynamic view-oriented group communication service. In *Symposium on Principles of Distributed Computing (PODC)*, pages 227–236, 1998.

[32] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In *Proc ACM SIGCOMM*, pages 161–172, 2001.

[33] Beverly A. Sanders. The information structure of distributed mutual exclusion algorithms. *ACM Transactions on Computer Systems*, 5(3):284–299, August 1987.

[34] Ian Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[35] Ivan Stojmenović and Pedro E. V. Pena. A scalable quorum based location update scheme for routing in ad hoc wireless networks. Technical Report TR-99-09, SITE, University of Ottawa, September 1999.