The Origin of Quorum Systems

Marko Vukolić
IBM Research - Zurich
mvu@zurich.ibm.com

Abstract

A quorum system is a collection of subsets of nodes, called quorums, with the property that each pair of quorums have a non-empty intersection. Quorum systems are the key mathematical abstraction for ensuring consistency in fault-tolerant and highly available distributed computing. This paper is a brief overview of the evolution of quorum systems, with emphasis on their role in two fundamental applications: distributed storage and replication.

1 Introduction

The very etymology of the word *quorum* ('of whom', Latin genitive plural of *qui*, who), is an indication of its importance, as it denotes a selected group. Quorums have been used for centuries in, e.g., legislative terminology, to denote the number, usually a majority, of 'officers or members of a body that when duly assembled is legally competent to transact business' [2]. A historical example of such a 'business' is the process of *voting*. Intuitively, requiring majorities to reach decisions in a voting process is critical in preventing (obviously undesirable) inconsistencies and partitioning in a legislative process.

This historical use of quorums has arguably inspired their use in computer science. Namely, in distributed computing, quorums come in groups, forming *quorum systems*. Given a set of nodes, typically servers, a quorum system is a collections of subsets of nodes, called quorums, every two of which intersect. A set of majorities is both a fundamental and obvious example of a quorum system.

Moreover, following the historical path further, it should not be surprising that the *raison d'être* of quorum systems in distributed computing is to guarantee *consistency*. Here, the key property of quorum systems is that of non-empty pair-wise intersections. The other important aspect of quorums, namely that they are (typically strict) subsets of a set of nodes, relates to the goals of higher availability [56], better load balancing [37] and fault-tolerance [13] in distributed systems. The key idea here is that a client accessing a (replicated) service does not necessarily need to communicate with all the nodes, but only with the nodes belonging to some quorum, which is itself often a strict subset of nodes. This leads to relaxation of the load on nodes that reside outside of a quorum and/or enables tolerance of their failures, resulting in higher availability.

In this context, quorum systems have been used to implement a wide variety of distributed objects and services. Typical examples include replicated databases [26, 35, 62], mutual exclusion [6, 45], read/write registers [11, 47] and group communication [9, 18], to name only a very few.

A comprehensive survey of all the protocols and techniques that rely on the notion of quorums would probably require a dedicated book and is most certainly beyond the scope of this paper. On the other hand, the goal of this paper is to briefly overview the evolution of quorum systems in distributed computing literature, with particular emphasis on refinements of the original quorum notion. Specifically, our goal is to overview how the simple non-empty intersection property has evolved in time and to attempt to explain different quorum access methods of a given quorum system, in terms of different model assumptions and design goals. The key applications we have in mind are two fundamental ones that have significantly benefitted from the evolution of quorum systems: distributed storage and replication. In addition, we overview some of the main quorum systems measures such as load and availability.

The remainder of the paper is organized as follows. Section 2 introduces some of the definitions and terminology used in the paper. Section 3 overviews classical quorum systems as well as few fundamental measures typically used to evaluate quorum systems and protocols that use them. Then we describe three refinements of the classical quorum systems. Section 4 describes Byzantine quorum systems, which are designed to provide consistency in presence of Byzantine failures. Section 5 describes probabilistic quorum systems that probabilistically depart from classical quorum system to provide better availability. Section 6 presents refined quorum systems, which further refine classical and Byzantine quorum systems, with the goal of designing strongly consistent distributed protocols with optimal latency. Finally, Section 7 concludes the paper.

2 Preliminaries

Basics. Given a set S, a set system (or a hypergraph) $\mathbf{H}$ is a subset of the powerset of S, i.e., $\mathbf{H} \subseteq 2^S$. In other words, a set system is a set of subsets of S. We denote by $m(\mathbf{H})$ the minimal cardinality of some element in $\mathbf{H}$, $m(\mathbf{H}) = \min_{Q \in \mathbf{H}} |Q|$. *Strategy* σ is a probabilistic function that takes a (non-empty) set system $\mathbf{H}$ as input, and outputs some $Q \in \mathbf{H}$ with probability $\sigma_{\mathbf{H}}(Q)$, such that $\sum_{Q \in \mathbf{H}} \sigma_{\mathbf{H}}(Q) = 1$.

Registers. In this paper we also discuss different read/write storage [22] semantics implemented using different quorum systems. For completeness, we informally define these storage semantics here (precise definitions can be found in, e.g., [12, 41, 44]).

We consider the notions of *safe*, *regular* and *atomic* storage (also called read/write *register*) introduced by Lamport in [41]. All three semantics behave in the same consistent way in the absence of read/write concurrency (also called *contention*); in this case, a read returns the last value written. In the case of contention, a read in safe storage may return an arbitrary value, whereas in regular storage, it may return either the last value written prior to the read or some of the values written concurrently. Finally, atomic storage has the strongest semantics and provides the reader with an illusion of a sequential access to storage.

3 Classical Quorum Systems

In this section, we first provide some background and a basic definition of quorum systems. This is followed by exemplifying some of the classical quorum systems used in literature. Finally, we introduce some important quorum system measures.

3.1 Background

In 1979, Thomas proposed [62] a majority approach to solving consensus [44] to maintain concurrency control over multiple copies of a replicated database. This paper, along with that of Gifford [26], has marked the dawn of quorum systems in distributed computing. Perhaps unsurprisingly, both papers used quorums in the context of voting. In short, Thomas used a majority voting scheme in which database copies vote on the acceptability of update requests. To write data to the database, the writer would timestamp the data and write it to a majority of servers. Then, to read the data, the reader would contact a (possibly different) majority, and return the data having the highest timestamp.

In this scheme, the majority-intersection property guarantees that the reader will obtain the latest value. This property is critical in preserving consistency in presence of potential network partitions.

Quorum systems can be defined in a more general context, refining the concept of majorities to allow arbitrary quorum sizes while maintaining the requirement for non-empty pair-wise quorum intersections. The basic definition we use in this paper is a variation of the definition given in [25]:

Definition 1 (Quorum System)**.** *Given a set* $S = \{s_1, s_2 \dots s_n\}$ *(*$n \geq 1$*), a set system* **QS** *is a quorum system over* S*, if and only if*

(Intersection) $\forall Q_1, Q_2 \in \mathbf{QS} : Q_1 \cap Q_2 \neq \emptyset$*.*

Elements of a quorum system are simply called *quorums*. When S is understood, we omit it for simplicity.

It is worth noting that the definition of Garcia-Molina and Barbara [25] differs somewhat, as it defines *coteries* (i.e., 'exclusive groups' [1]) in place of quorum systems. Coteries (see also, e.g., [38]) can be seen as minimal quorum systems, with the minimality property stating that there are no two quorums in a coterie such that one is the strict subset of the other. In this paper, unless stated otherwise, we will consider a more general, non-minimal notion of a quorum system, as defined in Definition 1.

In contrast to the work of Thomas, Gifford [26] introduced a weighted voted scheme and was also the first to refine the concept of majority-based quorums by separating the notions of read and write quorums. In this fundamental refinement, the Intersection property is relaxed so as not to require all quorums to intersect. Gifford separates quorums into two classes: *read* and *write* quorums, and requires only quorums belonging to different classes to intersect. In principle, this refinement with its distinctions between read and write quorums can be applied to any quorum system. Therefore, for simplicity and unless stated otherwise, the quorum systems surveyed in this paper will not account for the read/write distinction.

For a comprehensive survey of early approaches to consistency using quorum systems, the reader is referred to [24]. Classical quorum systems have also been extensively used in the context of storage simulations in message-passing systems. The seminal example of such is the ABD atomic storage simulation by Attiya, Bar-Noy and Dolev [11]. The original majority-based ABD simulation along with the survey of the subsequent work were discussed in a recent article by Attiya [10].

In the following, we briefly instantiate classical quorum systems through few classical examples (beyond majorities).

3.2 Examples

Singleton. The simplest quorum system is the one containing a singleton: **Singl** $= \{\{s_i\}\}$, for some $s_i \in S$.

Finite projective planes (FPP). If set S contains $n = k^2 + k + 1$ nodes, where k is a prime power, then a finite projective plane of order k is a quorum system, in which every quorum has exactly $k + 1 = O(\sqrt{n})$ nodes, every node is contained in exactly $k + 1$ quorums and every two quorums intersect in exactly one node. This quorum system was used by Maekawa [45] in his mutual exclusion algorithm.

The simplest example of an FPP quorum system is given by Fano plane, an order 2 FPP over a set of $n = 7$ nodes (see Fig. 1).

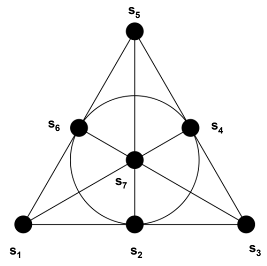

Figure 1: Fano plane (an order 2 FPP).

Grid. Assume $|S| = k^2$, for some integer k and nodes arranged in a square matrix (i.e., a grid). Then, a set of subsets of S of the form $Q_{i,j}$, such that each $Q_{i,j}$ contains all elements in row i and column j ($1 \leq i, j \leq k$) forms the quorum system over S. Such a quorum system has $k^2 = n$ quorums, each of size $2k - 1 = O(\sqrt{n})$ and every quorum intersects with every other quorum in at least 2 nodes.

To minimize the size of quorum intersection we can construct a slightly different quorum system containing $k = \sqrt{n}$ quorums Q_i ($1 \leq i \leq k$) such that Q_i contains all nodes from row i and exactly one node from each row $j > i$ [46]. It is not difficult to see that the quorum size in this quorum system (we refer to as the Grid) remains $O(\sqrt{n})$, whereas the size of pair-wise intersections among quorums is 1. The first grid-like quorum system was used in the replication protocol of Cheung et al [21].

B-Grid. A generalized grid-like quorum system, called B-Grid, was proposed by Naor and Wool [56]. This quorum system assumes a rectangular grid of R rows and c columns, such that rows are grouped into b bands of r rows ($R = br$), where band j ($1 \leq j \leq b$) contains rows $(j - 1)r + 1 \ldots jr$. Moreover, denote the intersection of column c and band j as *mini-column* (c, j). Then, the B-grid quorum system consists of b quorums Q_j ($1 \leq j \leq b$), each containing one mini-column from each band (i.e., b mini-columns (c_i, i), where $1 \leq i \leq b$ and $1 \leq c_i \leq c$) and one node from each column in band j (see also Fig. 2). It is not difficult to see that B-Grid is a quorum system in which every quorum contains exactly $br + c - 1$ nodes.

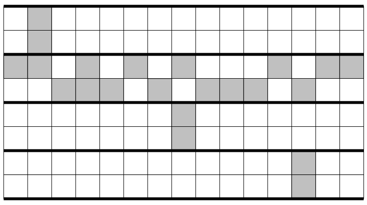

Figure 2: The B-Grid quorum system over a set of $n = brc = 120$ nodes, with $c = 15$ columns, $b = 4$ bands and $r = 2$ rows per band.

3.3 Measures

Two fundamental quorum system measures are *load* and *availability*, studied by Naor and Wool [56] and Peleg and Wool [57], respectively.

3.3.1 Load

In principle, a protocol using a quorum system will need to access some quorum. In the best case, a process accessing a quorum will be able to select a given quorum Q and access all nodes that belong to Q. Such usage of a quorum system will induce *load*, which, in short, measures the minimal access probability of the busiest node in the system. The load measures the quality of a quorum system: low load translates to the busiest node being accessed rarely, which allows it to perform other, unrelated tasks. Intuitively, the lower the load of a quorum system, the better.

With each strategy (see Sec. 2), there is an associated load induced on each node as well as the load on the entire quorum system. Given strategy σ, the load on a given node s_i is the probability that s_i will belong to a quorum selected according to strategy σ. The load of σ on a quorum system is defined as the *maximum* load of each individual node. Finally, the *(system) load of a quorum system* is the *minimum* load across all possible strategies.

Definition 2 (Load)**.** *Let* $\mathbf{QS}$ *be the quorum system over* S*. Then, we define the following:*

1. *Load induced by strategy* σ *on node* $s_i \in S$*:* $l_\sigma(s_i, \mathbf{QS}) = \sum_{s_i \in Q} \sigma_{\mathbf{QS}}(Q)$.
2. *Load induced by strategy* σ *on* $\mathbf{QS}$*:* $\mathcal{L}_\sigma(\mathbf{QS}) = \max_{s_i \in S} l_\sigma(s_i, \mathbf{QS})$.
3. *Load of* $\mathbf{QS}$*:* $\mathcal{L}(\mathbf{QS}) = \min_{\sigma} \mathcal{L}_\sigma(\mathbf{QS})$ *(minimum over all strategies* σ*).*

In [56], Naor and Wool also prove the lower bound on the load of *any* quorum system. Namely, they show that $\mathcal{L}(\mathbf{QS}) \geq \max\left\{\frac{1}{m(\mathbf{QS})}, \frac{m(\mathbf{QS})}{n}\right\}$, which implies $\mathcal{L}(\mathbf{QS}) \geq 1/\sqrt{n}$.

It is very important to highlight that a load of a given quorum system is *independent* of any given protocol that might use this quorum system. On the other hand, a given protocol might come with a given strategy σ and the associated load induced by this strategy.

In a sense, the above definition of load is a best-case one because the connection between strategies and load assumes that a quorum selected by a strategy will always be accessed. This obviously does not account for possible failures and/or asynchrony, which may prevent a selected quorum from being accessed. The reader is referred to [56] for an extended definition of load that accounts for failures. Moreover, Yu [65] proposes a definition of the load measure that extends its scope beyond the best case, by studying the load under asynchrony where *probing* [59] of multiple quorums is typically needed before a quorum can be acquired, which intuitively makes servers busier.

3.3.2 Availability

Resilience. A fundamental availability measure of quorum systems is the *resilience* (sometimes also referred to as *node vulnerability* [13] or *fault-tolerance* [48]). The resilience $\mathcal{R}(\mathbf{QS})$ of a quorum system $\mathbf{QS}$ is defined as the maximal integer t such that, despite a failure of any t nodes in the system, there is a quorum $Q \in \mathbf{QS}$ such that no node belonging to Q fails. Note that the Intersection property implies $\mathcal{R}(\mathbf{QS}) < m(\mathbf{QS})$. Namely, the failure of all nodes in any single quorum implies at least one node failure in every quorum. Therefore, the resilience is bounded by the minimal quorum cardinality.

Of particular importance are *optimally resilient* quorum systems. It is straightforward to show that no quorum system can have a resilience greater than $\left\lfloor \frac{n-1}{2} \right\rfloor$. This is exactly the resilience of a majority coterie. We say that a quorum system is t-resilient if and only if its resilience is at least t.

Finally, the definition of resilience can be extended beyond threshold failures [17]. A quorum system $\mathbf{QS}$ is said to be *resilient* to set system $\mathbf{F}$, if and only if $\forall F \in \mathbf{F}, \exists Q \in \mathbf{QS} : Q \cap F = \emptyset$.

Failure probability. Another availability measure, introduced by Peleg and Wool in [57], is the *global failure probability*, or simply the *failure probability*. In fact, the failure probability (a.k.a. non-availability) of a quorum system $\mathbf{QS}$, denoted by $F_p(\mathbf{QS})$, is in a sense dual to availability. In short, $F_p(\mathbf{QS})$ is the probability that no quorum in $\mathbf{QS}$ will have a non-faulty node. Here, p is the probability of failure of each of the nodes in S, where it is assumed that failures are independent with a uniform probability distribution. Moreover, here a failure is assumed to be a detectable *crash-stop* failure.

More precisely, for each quorum $Q \in \mathbf{QS}$ denote by $F_p(Q)$ the probability that some node in Q fails. Clearly, as we assume p to be uniformly distributed, we have $F_p(Q) = 1 - (1 - p)^{|Q|}$. Then, failure probability is defined simply as

$$F_p(\mathbf{QS}) = \prod_{Q \in \mathbf{QS}} F_p(Q).$$

In general, quorum system **QS** is considered to have good failure probability if $F_p(\mathbf{QS})$ tends to 0 for large values of n assuming $p < 1/2$.

Peleg and Wool also relate two availability measures, namely the resilience and failure probability, by showing that $F_p(\mathbf{QS})$ is at least $e^{-\Omega(\mathcal{R}(\mathbf{QS}))}$ [57]. Moreover, since $\mathcal{R}(\mathbf{QS}) < m(\mathbf{QS})$ and $m(\mathbf{QS}) \leq n\mathcal{L}(\mathbf{QS})$, the tradeoff between resilience and load is expressed as $\mathcal{R}(\mathbf{QS}) < n\mathcal{L}(\mathbf{QS})$. Finally, we can express the tradeoff between the failure probability and the load [56]. Clearly, the failure probability is at least the probability that all the nodes from the quorum containing the smallest number of nodes fail, i.e., $F_p(\mathbf{QS}) \geq p^{m(\mathbf{QS})} \geq p^{n\mathcal{L}(\mathbf{QS})}$.

3.3.3 Comparison

In Table 1 we overview the quality measures of the quorum systems presented in Section 3.2 [46, 56].

QS	$\mathcal{L}(\mathbf{QS})$	$\mathcal{R}(\mathbf{QS})$	$F_p(\mathbf{QS})$
Singleton	1	0	p
Majorities	$\left\lceil \frac{n+1}{2} \right\rceil$	$\left\lfloor \frac{n-1}{2} \right\rfloor$	$e^{-\Omega(n)}$
FPP	$O(\frac{1}{\sqrt{n}})$	$O(\sqrt{n})$	1*
Grid	$O(\frac{1}{\sqrt{n}})$	$O(\sqrt{n})$	1*
B-Grid	$O(\frac{1}{\sqrt{n}})$	$O(\sqrt{n})$	$O(e^{-\frac{n^{1/4}}{2}})$

* For large values of n.

Table 1: Comparison of the load, resilience and failure probability for different quorum systems.

Singleton is interesting when the individual node probability failure is high ($p > 1/2$), when this simple quorum system offers the best failure probability. Otherwise (assuming $p < 1/2$), Majorities has the best availability but poor load. On the other hand, FPP and Grid have optimal load, yet their failure probability is poor for large value of n. This is corrected by B-Grid, which has asymptotically optimal failure probability. The data in Table 1 for B-Grid are given assuming $c = \sqrt{n}$, $r = ln(c)$ and $p \leq \frac{1}{3}$.

Other quorum systems that combine optimal load with optimal failure probability include CWalls proposed by Peleg and Wool [58], Paths by Naor and Wool [56] and Bazzi's Triangle Lattice [15].

4 Byzantine Quorum Systems

So far we have discussed classical quorum systems, applicable in the context of crash failures. However, if node failures can be arbitrary, also called Byzantine [43], simple non-empty quorum intersections are not sufficient to guarantee consistency. Intuitively, if an intersection between two quorums contains, for example, a single node, and this node can be Byzantine, the Byzantine node can simply violate consistency. As an illustration, assuming the Byzantine node is the sole node in the intersection of a write and a read quorum, it can simply "forget" seeing the write and cause inconsistencies in a read.

Although it has been known for some time that tolerating Byzantine failures in, e.g., replication requires a larger fraction of correct nodes than tolerating crash failures only does (see e.g., the seminal work by Lamport et al. [43]), the first formal treatment of the problem of defining quorum systems in Byzantine context was done by Malkhi and Reiter [47]. They define several types of *Byzantine quorum systems* depending on the type of data the targeted application is designed to store, as we explain in the following.

Byzantine quorum systems are specified not only with respect to a given set of nodes S, but also assuming a given set system over S called *adversary* (also called *adversary structure* [36] or *fail-prone set* [47]).

Definition 3 (Adversary). *Given a set S, a set system $\mathbf{B}$ is an adversary for S if and only if $B \in \mathbf{B} \wedge B' \subseteq B \Rightarrow B' \in \mathbf{B}$.*

Intuitively, the adversary is defined to capture all possible combinations of simultaneously Byzantine nodes. In the following, we assume that an adversary for S contains as its elements all possible subsets of nodes whose elements can be simultaneously Byzantine. Armed with the above definition of the adversary, we are ready to define Byzantine quorum systems.

4.1 Dissemination Quorum Systems

One of the families of Byzantine quorum systems proposed in [47] are *dissemination* quorum systems.

Definition 4 (Dissemination quorum systems)**.** *Given a set S and an adversary* $\mathbf{B}$ *for S, a quorum system (over S)* $\mathbf{DQS}$ *is a dissemination quorum system over S if and only if*

(Byzantine intersection) $\forall Q_1, Q_2 \in \mathbf{DQS}, \forall B \in \mathbf{B} : Q_1 \cap Q_2 \not\subseteq B$.

Dissemination quorum systems were proposed in [47] with the aim of storing *self-verifying*, also called *authenticated*, data. In short, authenticated data are data that cannot be forged by a Byzantine node; in practice this feature is typically implemented using digital signatures. We further discuss the use of dissemination quorum systems in Section 4.3.

In [47], Malkhi and Reiter define a dissemination quorum system with an additional availability property requiring resilience to $\mathbf{B}$ (see Sec. 3.3.2). While this is clearly a necessary condition for the availability of a disseminating quorum system and the liveness of an underlying service, it may prohibit some applications which require a service to be safe, but not always live [61]. Therefore and to maintain generality, in this paper, we choose to separate the definitions of quorum systems (i.e., their intersection properties) from the availability considerations.

Denote by $\mathbf{B}_t$ a threshold adversary that contains all subsets of S of cardinality at most t ($\mathbf{B}_t = \{Q \subseteq S : |Q| \leq t\}$). A special and particularly important case of a dissemination quorum system, is a t-dissemination quorum system. Given the adversary $\mathbf{B}_t$, a dissemination quorum system is a t-dissemination quorum system if its resilience is at least t.

It is straightforward to show that no t-dissemination quorum system can be constructed if $n \leq 3t$. On the other hand if, e.g., $n = 3t + 1$, an example of a t-dissemination quorum system is a two-thirds majority quorum system: $\mathbf{Maj}_{\frac{2}{3}} = \{Q \subseteq S : |Q| = n - t = 2t + 1\}$. In this context, $\mathbf{Maj}_{\frac{2}{3}}$ is optimally resilient, with the resilience expressed as a function of n: $\mathcal{R}(\mathbf{Maj}_{\frac{2}{3}}) = \left\lfloor \frac{n-1}{3} \right\rfloor$. More generally, it can be shown that, for $n > 3t$, $\mathbf{DQS_t} = \left\{Q \subseteq S : |Q| = \left\lceil \frac{n+t+1}{2} \right\rceil\right\}$ is a t-dissemination quorum system.

It can be also shown, along the lines of [50], that the lower bound on the load of any t-dissemination quorum system $\mathbf{tDQS}$ is given by $\mathcal{L}(\mathbf{tDQS}) \geq \max\left\{\frac{t+1}{m(\mathbf{tDQS})}, \frac{m(\mathbf{tDQS})}{n}\right\}$, which implies $\mathcal{L}(\mathbf{tDQS}) \geq \sqrt{\frac{t+1}{n}}$. More concretely, the load of the t-dissemination quorum system $\mathbf{DQS_t}$ defined above is $\mathcal{L}(\mathbf{DQS_t}) = \frac{1}{n}\left\lceil \frac{n+t+1}{2} \right\rceil$.

4.2 Masking Quorum Systems

In [47], Malkhi and Reiter also proposed *masking* quorum systems with the goal of storing *unauthenticated* data. As we discuss in Section 4.3, relying on masking quorum systems to store unauthenticated data revealed not to be necessary, because unauthenticated data can be stored in a consistent manner using dissemination quorum systems only. However, designing storage protocols for storing unauthenticated data is arguably much simpler when using masking instead of dissemination quorum systems.

Masking quorum systems are defined as follows:

Definition 5 (Masking quorum systems)**.** *Given a set S and an adversary* $\mathbf{B}$ *for S, a quorum system (over S)* $\mathbf{MQS}$ *is a masking quorum system over S if and only if*

(M-Byzantine intersection) $\forall Q_1, Q_2 \in \mathbf{MQS}, \forall B_1, B_2 \in \mathbf{B} : Q_1 \cap Q_2 \nsubseteq B_1 \cup B_2$.

Given the above definition, it is straightforward to see that masking quorums are a further refinement of disseminating quorums: all masking quorum systems are also disseminating quorum systems, but, the opposite does not hold. Analogously to t-dissemination quorum systems, we can define t-masking quorum system [47] as masking quorum systems for the threshold adversary $\mathbf{B}_t$ with the resilience of at least t. It is straightforward to show that no t-masking quorum system **tMQS** can be constructed if $n \leq 4t$. For $n > 4t$, it can be shown [47] that $\mathbf{MQS_t} = \left\{Q \subseteq S : |Q| = \left\lceil \frac{n+2t+1}{2} \right\rceil\right\}$ is a t-masking quorum system. The load of this quorum system is $\mathcal{L}(\mathbf{MQS_t}) = \frac{1}{n}\left\lceil \frac{n+2t+1}{2} \right\rceil$ and is achieved by the strategy that assigns uniform probabilities across all quorums.

More generally, the lower bound on the load of any t-masking quorum system **tMQS** is given by $\mathcal{L}(\mathbf{tMQS}) \geq \max\left\{\frac{2t+1}{m(\mathbf{MQS})}, \frac{m(\mathbf{tMQS})}{n}\right\}$, implying $\mathcal{L}(\mathbf{tMQS}) \geq \sqrt{\frac{2t+1}{n}}$.

Opaque masking quorum systems. In [47] Malkhi and Reiter also propose a specific variant of masking quorum systems designed for the case where the adversary is *opaque*, in a sense that the adversary is fixed, yet not known to correct nodes. Roughly speaking, this quorum system provides an invariant stating that the *count* on the number of correct nodes in an intersection of any two quorums will be higher (or equal) to the number of stale nodes in any given quorum plus the number of Byzantine nodes. One of the main results in [47] related to opaque masking quorums is that no $\mathbf{B}_t$-resilient opaque quorum systems can be constructed with less than $5t$ nodes. An example of an implicit use of an opaque masking quorum system is the safe storage implementation of Jayanti et al. [39]. For more details on opaque masking quorum systems the reader is referred to [47].

4.3 Usage

Byzantine quorum systems have been widely used in asynchronous read/write storage emulations, typically projected to the threshold failure model. In [47], Malkhi and Reiter also proposed a single-writer multi-reader regular storage construction for storing authenticated data using dissemination quorum systems in which both reads and writes access a given quorum only once. This approach was further extended by the same authors in [49] to implement multi-writer multi-reader atomic storage relying again on dissemination quorum systems and data authentication. The improvement in terms of supporting multiple writers and providing stronger semantics came at the price of having to access quorums twice in each read and write.

In contrast to dissemination quorum systems, masking quorum systems were designed to store unauthenticated data. In this setting, Malkhi and Reiter proposed in [47] a single-writer multi-reader safe storage construction in which read and write operations accessed a quorum only once.

However, Martin and Alvisi showed an atomic storage implementation in [53] that stores unauthenticated data but uses only dissemination quorums. Not surprisingly, the protocol of Martin and Alvisi, called SBQ-L (Small Byzantine Quorums with Listeners), used much more involved techniques than its counterpart of Malkhi and Reiter that stores authenticated data [49]. Following the work by Martin and Alvisi, many storage constructions that use disseminating quorum systems to store unauthenticated data were proposed, typically with the goal of reducing complexity. These include safe and regular storage constructions of Abraham et al. [5] and Guerraoui and Vukolić [32], as well as the atomic storage construction of Aiyer et al. [8].

However, masking quorum systems are arguably simpler to use in unauthenticated storage constructions. This was formally proved by Abraham et al. [5], who showed that, in our terminology, any consistent (i.e., safe) storage that stores unauthenticated data in a disseminating, non-masking, quorum must have write operations access a quorum more than once in the worst case. The same result was also shown for read operations [5, 32]. Using masking quorums helps achieve simpler design (recall here the safe implementation of Malkhi and Reiter [47]), but it comes with the price of lower resilience and higher load, as we already discussed. In this context, masking quorum systems were used in the atomic storage construction of Bazzi and Ding [14] and the regular one of Abraham et al [4]. Bazzi used a variation of masking quorum systems in the synchronous model [16].

Disseminating quorum systems, in particular $\mathbf{Maj}_{\frac{2}{3}}$ (assuming $\mathbf{B}_{\lfloor \frac{n-1}{3} \rfloor}$), underly many other Byzantine fault-tolerant protocols beyond storage, including replication protocols. Examples include the seminal work of Castro and Liskov [19], or the replication protocols of Cowling et al. [23].

Martin et al. proposed in [51] variants of dissemination and masking quorum systems that account for the distinction between read and write quorums. They also proposed the construction of threshold-based disseminating and masking quorum systems with as few as $2t+1$ and $3t+1$ nodes, respectively. To achieve this, these quorum systems sacrifice the resilience of write quorums but maintain read quorums t-resilient. Bazzi [17] also extends the notion of Byzantine quorum systems, defining *non-blocking* quorum systems in the context of studying the asynchronous access cost of quorum systems.

5 Probabilistic Quorum Systems

Brewer's CAP theorem [27] states that, in short, no distributed system can provide consistency, high availability and partition tolerance. As we already discussed, the defining point of quorum systems is consistency. Moreover, quorum systems aim at providing high availability. However, this means that quorum systems cannot imply partition tolerance (this is intuitive from the requirement for the non-empty quorum intersections). This is one of the reasons modern large-scale distributed systems, including cloud computing systems, for which availability and partition tolerance are of paramount importance, relax consistency guarantees only to provide eventual consistency [63]. The natural question that arises is, how do quorum systems fit into this picture? Did they become obsolete?

The answer to the first question is twofold. First, systems that rely on eventual consistency must provide consistency, albeit only eventually, when a partition in the system is repaired. To achieve this, these systems resort to some quorum systems even if only eventually (this also gives a negative answer to the second question above).

Second, quorum systems researchers have been aware of such issues associated with the limited availability of classical, strongly consistent, quorum systems for a long time [64]. Recall here that, for any quorum system **QS**: a) the resilience $\mathcal{R}(\mathbf{QS})$ is at most $\left\lfloor \frac{n-1}{2} \right\rfloor$ (where n is the number of nodes in the system), and b) the failure probability $F_p(\mathbf{QS})$ tends to 1 when the individual failure probability p is greater than $1/2$.

To cope with this, *probabilistic* quorum systems were proposed. In short, these quorum systems aim to improve availability by *relaxing* probabilistically the Intersection property. In the following, we first discuss *ϵ-intersecting* quorum systems of Malkhi et al. [48], and then briefly overview more recent work on highly available quorum systems.

5.1 ϵ-Intersecting Quorum Systems

The pioneer work in the context of probabilistic quorum systems was that of Malkhi et al. [48]. This work introduced *ϵ-intersecting* quorum systems with a subtle refinement of the Intersection property of classical quorum systems that allows non-intersection with a certain probability ϵ. [1]

More precisely, ϵ-intersecting quorum systems can be defined as follows:

Definition 6 (ϵ-intersecting Quorum System)**.** *Given a set $S = \{s_1, s_2 \ldots s_n\}$ ($n \geq 1$), let $\epsilon\mathbf{IQS}$ be a set system and let σ be a strategy for $\epsilon\mathbf{IQS}$ with an associated probability ϵ_σ. Then, a tuple $\langle \epsilon\mathbf{IQS}, \sigma \rangle$ is an ϵ-intersecting quorum system over S if and only if*

(ϵ-intersection) $\forall Q_1, Q_2 \in \epsilon\mathbf{IQS} : P(Q_1 \cap Q_2 \neq \emptyset) \geq 1 - \epsilon_\sigma$.

While ϵ-intersecting quorum systems cannot guarantee consistency, they can, often transparently, substitute classical quorums in existing implementation if strong consistency is not mandatory. Malkhi et al demonstrate this by giving a simple safe single-writer multi-reader storage implementation as an illustration [48] in which both read and write access a given probabilistic quorum only once. As expected, such an implementation violates consistency with probability ϵ_σ [48].

The load of an ϵ-intersecting quorum system $\langle \epsilon\mathbf{IQS}, \sigma \rangle$ is simply the load induced by strategy σ on $\epsilon\mathbf{IQS}$, i.e., $\mathcal{L}(\langle \epsilon\mathbf{IQS}, \sigma \rangle) = \mathcal{L}_\sigma(\epsilon\mathbf{IQS})$. In [48], Malkhi et al. generalize the lower bound on load given by Naor and Wool in [56] (see Sec. 3.3.1) by showing that $\mathcal{L}(\langle \epsilon\mathbf{IQS}, \sigma \rangle) \geq \max\left\{\frac{1-\sqrt{\epsilon_\sigma}}{E_\sigma[|Q|]}, \frac{E_\sigma[|Q|]}{n}\right\}$, where $E_\sigma[|Q|]$ is an expectation of accessed quorum size for quorums $Q \in \epsilon\mathbf{IQS}$ taken over strategy σ.[2] Then, it is simple to show that $\mathcal{L}(\langle \epsilon\mathbf{IQS}, \sigma \rangle) \geq \frac{1-\sqrt{\epsilon_\sigma}}{\sqrt{n}}$.

Clearly, with ϵ_σ small, ϵ-intersecting quorum systems provide marginally better load than classical quorum systems, in general. However, ϵ-intersecting quorum system allow construction of quorum systems that combine both (close to) optimal load and $\Omega(n)$ resilience, which is not possible with classical quorum systems. Namely, Malkhi et al. suggest in [48] an ϵ-intersecting quorum system (denoted by $\epsilon\mathbf{IQS}_{\mathbf{l}\sqrt{n}}$) in which the quorums are all sets of size $l\sqrt{n}$, with a strategy chosen uniformly at random and where constant l is chosen to make ϵ sufficiently small. It can be shown [48] that the probability of two such probabilistic quorums have an empty intersection is:

$$P(Q \cup Q' = \emptyset) = \frac{\binom{n-l\sqrt{n}}{l\sqrt{n}}}{\binom{n}{l\sqrt{n}}} \leq e^{-l^2},$$

[1] Note that, strictly speaking, probabilistic quorum systems are not classical quorum systems (as they may violate the Intersection property), unlike Byzantine quorum systems.

[2] In other words, $E_\sigma[|Q|] = \sum_{Q_j \in \mathbf{QS}} \sigma_j |Q_j|$.

which makes $\epsilon\mathbf{IQS}_{\mathbf{l}\sqrt{n}}$ an e^{-l^2}-intersecting quorum system. Since $\epsilon\mathbf{IQS}_{\mathbf{l}\sqrt{n}}$ can tolerate up to $n - l\sqrt{n}$ crashes its resilience is $\Omega(n)$ (notice here how the probabilistic intersection property relaxes the $m(\mathbf{QS})$ upper bound on resilience). Finally, failure probability of $\epsilon\mathbf{IQS}_{\mathbf{l}\sqrt{n}}$ is less than $e^{-\Omega(n)}$ for $p \leq 1 - \frac{l}{\sqrt{n}}$ [48], which is asymptotically optimal and if $p \geq 1/2$ strictly better than failure probability of any classical quorum system.

5.2 Related Work

Besides the crash variant of ϵ-intersecting quorum systems, Malkhi et al. present in [48] the application of ϵ-intersection property to Byzantine quorum systems, namely to dissemination and masking quorum systems. Merideth and Reiter [55] complement this work by analyzing probabilistic intersections in the context of opaque masking quorum systems.

One issue with ϵ-intersecting quorum systems is that they do not account for the network adversary that controls the system scheduler in, e.g., asynchronous system. Intuitively, such an adversary could always violate the probabilistic intersection guarantees by, e.g., partitioning the writer and the reader and two quorums eQ_1 and eQ_2 by arbitrarily delaying messages sent by the writer to nodes in eQ_2 and by the reader to eQ_1.

Yu [65] explicitly acknowledges this issue, and proposes an alternative definition of probabilistic quorum systems, called *signed* quorum systems. Signed quorum system are not defined around access strategies (for these can be disturbed by the scheduler); in Yu's approach, a strategy is implicit and dictated by the scheduler and failures in the system. In short, signed quorum systems allow both positive and negative node ids in a quorum, where negative ids denote nodes that are suspected to be faulty and cannot be accessed.[3] Signed quorum systems require quorums to intersect, *or* non-intersecting quorums to differ in at least 2α node states (signs) for some integer α. The probability of having an empty intersection between two quorums is then the probability that two clients assess at least 2α nodes in different states which is lower for larger values of α.

Finally, we note that Aiyer et al. [7] argue that Yu's approach remains vulnerable to the adversarial scheduler issue and propose k-quorum protocols to boost availability of classical quorum systems. In short, k-quorum protocols use classical quorum systems yet allow the writer to lazily contact the writer quorum such that all nodes from a quorum are contacted during $k \geq 1$ consecutive writes (vs. $k = 1$ in the classical approach), which allows the reader to return one of the last $k \geq 1$ written values.

[3] In other words, signed quorum systems assume a failure detector [20].

6 Refined Quorum Systems

A lot of attention in distributed computing is focused on optimizing common-case system behavior. The typical research goal in this context is to provide reliable, robust and consistent service under worst-case system conditions, i.e., asynchrony, large number of failures and high contention, and at the same time have such a service perform efficiently in the common-case, characterized by synchrony, few failures, and possibly even low contention. Distributed protocols proposed in this context include (i) replication protocols, both crash-tolerant ones such as Lamport's Fast Paxos [42] and Byzantine fault-tolerant ones such as Q/U [3], Zyzzyva [40] or Aliph [30]; (ii) atomic read/write storage protocols of Goodson et al. [29] and Guerraoui et al. [31]; (iii) consensus protocols of Martin and Alvisi [52] and Zielinski [66], and (iv) the atomic broadcast protocol of Ramasamy and Cachin [60]. A typical goal of such protocols is to minimize the number of quorum accesses in the common case. Ideally, a given quorum should be accessed only once, in particular in the common case.

It turns out that these protocols, in particular those that target optimal resilience, rely in the worst case on classical (e.g., majorities) and dissemination quorum systems (e.g., $\mathbf{Maj}_{\frac{2}{3}}$). However, in the common case, these protocols typically require quorums that are to be accessed only once to have *larger* intersections with other quorums to maintain consistency. In general, the nature of such larger quorum intersections is not captured by classical, dissemination or even masking quorum systems.

A general characterization of such quorum systems, called *refined quorum systems*, was proposed by Guerraoui and Vukolić [33]. Refined quorum systems refine classical and Byzantine quorum systems further and distinguish *three different quorum classes*. Intuitively, in the common case, a distributed object implementation can expedite an operation accessing a first-class quorum by allowing it to access such a quorum only once, whereas quorums of the second and the third class must be accessed at least twice and three times, respectively. Refined quorum systems are designed for use both in the Byzantine failure model (assuming unauthenticated data) and in the simple crash failure model (assuming adversary $\mathbf{B} = \{\emptyset\}$).

Refined quorum systems are defined as follows [33]:

Definition 7 (Refined quorum systems). *Given a set S and an adversary $\mathbf{B}$ for S, a set system (over S) $\mathbf{RQS}$ is a refined quorum system over S, if and only if there are two set systems $\mathbf{QC_1}$ and $\mathbf{QC_2}$, such that $\mathbf{QC_1} \subseteq \mathbf{QC_2} \subseteq \mathbf{RQS}$ and*

(Class-1 inters.) $\forall Q_1, Q_1' \in \mathbf{QC_1}, \forall Q \in \mathbf{RQS}, \forall B_1, B_2 \in \mathbf{B} : Q_1 \cap Q_1' \cap Q \not\subseteq B_1 \cup B_2,$

(Class-2 inters.) $\forall Q_1 \in \mathbf{QC_1}, \forall Q_2 \in \mathbf{QC_2}, \forall Q \in \mathbf{RQS}, \forall B_1, B_2 \in \mathbf{B}$:
$$(Q_2 \cap Q \not\subseteq B_1 \cup B_2) \bigvee (Q_1 \cap Q_2 \cap Q \not\subseteq B_1)\text{, and}$$

(Class-3 (Byzantine) inters.) $\forall Q, Q' \in \mathbf{RQS}, \forall B \in \mathbf{B} : Q \cap Q' \not\subseteq B$.

Note that the Class-3 intersection property is the Byzantine intersection property of dissemination quorum systems (see Def. 4) and it is required to hold for all refined quorums. There are also two special classes of refined quorums: class 1 quorums, that belong to $\mathbf{QC_1}$ and class 2 quorums that belong to $\mathbf{QC_2} \setminus \mathbf{QC_1}$. Moreover, the remaining quorums from $\mathbf{RQS} \setminus \mathbf{QC_2}$ are called class 3 quorums. The above definition is given for the special (yet the most interesting) case where $\mathbf{QC_1} \neq \emptyset$.

Refined quorum systems were used in [33] to implement common-case latency optimal SWMR atomic storage and consensus protocols. As we already intuited, these protocols have the property that when a class 1 quorum is available they require such a quorum to be accessed only once in the common case (in the case of storage this subsumes synchrony and no contention). Otherwise, the protocols gracefully degrade to require 2 (resp., 3) accesses in case a class 2 (resp., 3) quorum is available. An important aspect of the intersection properties of refined quorum systems is that they are also necessary which makes the implementations of [33] optimal.

The intuition behind the Class-1 intersection property can roughly be summarized in the requirement that the intersection X between two class 1 quorums Q_1 and Q_1' accessed only once by e.g., the writer and the reader, must have enough information for a subsequent reader accessing quorum Q, so that the latter does not return the stale value. Intuitively, X and Q should intersect here just like masking quorums do since data is not authenticated.

The intuition behind the Class-2 intersection property is more involved: we explain it here assuming a write accessing a class 1 quorum Q_1 and a read *rd* that accesses a class 2 quorum Q_2, followed by another read *rd′* that accesses a class 3 quorum Q. The key idea here is that *rd* is allowed to access class 2 quorum Q_2 twice. Moreover, in the common case, the reader will know which value it should return already after the first access of Q_2. Then, in the second access, the reader can 'confirm' the value by writing it back to quorum Q_2. In case Q_2 has a masking-like intersection with class 3 quorums, including Q (see the first condition in the disjunction), this is sufficient for *rd′* not to miss the value read by *rd*. On the other hand, such a masking-like quorum intersection of class 2 quorums is not always necessary. Namely (see the second condition in the disjunction), if intersections between class 1 quorums and Q_2 act like dissemination quorums with respect to class 3 quorums, it is sufficient that the reader "authenticates" the data by writing the value for the second time in the intersection $X = Q_1 \cap Q_2$ when it accesses

Q_2 for the second time. Here, roughly speaking, writing the value in nodes in X (at least) twice (once by the writer and once by the reader) has the effect of strengthening and confirming unauthenticated data so the masking intersection is no longer required.

For the full details behind refined quorum systems, the reader is referred to [33]. Here, we give two important instantiations of refined quorums that assume the threshold adversary $\mathbf{B_t}$.

First, assume that all quorums are class 1 quorums. Then the Class 1 intersection property implies the other two. Moreover, it is not difficult to see that no t-resilient refined quorum system can be constructed unless $n > 5t$. Assuming $n = 5t + 1$, we can construct a refined quorum system in which all subsets of size $n - t = 4t + 1$ are quorums. This is exactly the quorum system used in latency efficient replication protocols that require $5t + 1$ servers [3, 52]. These protocols were one of the very first to provide a latency-optimized service requiring a quorum to be accessed only once.

However, it turns out that paying the price of $5t+1$ servers is not necessary. Fix t, assume $n = 3t + 1$ and consider a refined quorum system in which $\mathbf{QC_1} = \{S\}$ and $\mathbf{QC_2} = \{Q \subseteq S : |Q| \geq n - t = 2t + 1\}$. It is not difficult to show that such quorum system indeed satisfies the properties of Definition 7: the set of all nodes is a class 1 quorum whereas any two-third majority is a class 2 quorum. This quorum system was used in, e.g., Zyzzyva [40] to allow an optimally resilient replication protocol expedite a common-case operation accessing all servers. In refined quorum system terminology, the set of all servers is in this case a class 1 quorum; therefore, it can safely be accessed only once.

7 Concluding Remarks

In this paper, we gave a brief overview of the evolution of quorum systems, reflected through the refinements of the original non-empty intersection property of quorums. We also emphasized the impact these refinements have on two fundamental applications: distributed storage and replication.

Bearing in mind that our goal was not to provide a comprehensive overview of all aspects of quorum systems, let alone all protocols that make use of the quorum notion, we highlight some of these additional aspects not addressed in this paper.

One practical problem that arises when quorums are cast from into a real system is the problem of quorum deployment introduced by Gilbert and Malewitz [28]. This problem raises the question of using quorums optimally, with the goal of determining the mapping from the real nodes in the network to the abstract nodes in the quorum specification. This is tightly related to the problem of quorum placement in networks as discussed by Gupta et al. [34]. For more information on these aspects, the reader is referred to a recent related survey of Merideth and Reiter [54].

Acknowledgments

I thank Christian Cachin for valuable comments and Charlotte Bolliger for improvements on the readability of this manuscript.

References

[1] coterie. In Merriam-Webster Online Dictionary. Retrieved April 27, 2010 from http://www.merriam-webster.com/dictionary/coterie.

[2] quorum. In Merriam-Webster Online Dictionary. Retrieved April 27, 2010 from http://www.merriam-webster.com/dictionary/quorum.

[3] Michael Abd-El-Malek, Gregory R. Ganger, Garth R. Goodson, Michael K. Reiter, and Jay J. Wylie. Fault-scalable Byzantine fault-tolerant services. In *Proceedings of the 20th ACM symposium on Operating systems principles*, pages 59–74, October 2005.

[4] Ittai Abraham, Gregory Chockler, Idit Keidar, and Dahlia Malkhi. Wait-free regular storage from Byzantine components. *Inf. Process. Lett.*, 101(2):60–65, 2007.

[5] Ittai Abraham, Gregory V. Chockler, Idit Keidar, and Dahlia Malkhi. Byzantine disk paxos: optimal resilience with Byzantine shared memory. *Distributed Computing*, 18(5):387–408, 2006.

[6] D. Agrawal and A. El Abbadi. Efficient solution to the distributed mutual exclusion problem. In *Proceedings of the 8th annual ACM Symposium on Principles of distributed computing*, pages 193–200, New York, NY, USA, 1989. ACM.

[7] Amitanand S. Aiyer, Lorenzo Alvisi, and Rida A. Bazzi. On the availability of non-strict quorum systems. In *Proceedings of the 19th International Conference on Distributed Computing*, pages 48–62, 2005.

[8] Amitanand S. Aiyer, Lorenzo Alvisi, and Rida A. Bazzi. Bounded wait-free implementation of optimally resilient Byzantine storage without (unproven) cryptographic assumptions. In *Proceedings of the 21st International Symposium on Distributed Computing*, pages 7–19, September 2007.

[9] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella. The totem single-ring ordering and membership protocol. *ACM Trans. Comput. Syst.*, 13(4):311–342, 1995.

[10] Hagit Attiya. Robust simulation of shared memory: 20 years after. *Bulletin of the European Association for Theoretical Computer Science EATCS*, (100):99–113, Feb 2010.

[11] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. Sharing memory robustly in message-passing systems. *J. ACM*, 42(1):124–142, 1995.

[12] Hagit Attiya and Jennifer Welch. *Distributed Computing. Fundamentals, Simulations, and Advanced Topics*. McGraw-Hill, 1998.

[13] Daniel Barbara and Hector Garcia-Molina. The vulnerability of vote assignments. *ACM Trans. Comput. Syst.*, 4(3):187–213, 1986.

[14] Rida Bazzi and Yin Ding. Non-skipping timestamps for Byzantine data storage systems. In *Proceedings of the 18th International Symposium on Distributed Computing*, pages 405–419, Oct 2004.

[15] Rida A. Bazzi. Planar quorums. *Theor. Comput. Sci.*, 243(1-2):243–268, 2000.

[16] Rida A. Bazzi. Synchronous Byzantine quorum systems. *Distrib. Comput.*, 13(1):45–52, 2000.

[17] Rida A. Bazzi. Access cost for asynchronous Byzantine quorum systems. *Distributed Computing*, 14(1):41–48, 2001.

[18] Kenneth P. Birman and Robert V. Renesse. *Reliable Distributed Computing with the ISIS Toolkit*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[19] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[20] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. 43(2):225–267, March 1996.

[21] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. *IEEE Trans. on Knowl. and Data Eng.*, 4(6):582–592, 1992.

[22] Gregory Chockler, Rachid Guerraoui, Idit Keidar, and Marko Vukolić. Reliable distributed storage. *IEEE Computer*, 42(4):60–67, 2009.

[23] James Cowling, Daniel Myers, Barbara Liskov, Rodrigo Rodrigues, and Liuba Shrira. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementations*, pages 177–190, November 2006.

[24] Susan B. Davidson, Hector Garcia-Molina, and Dale Skeen. Consistency in a partitioned network: a survey. *ACM Comput. Surv.*, 17(3):341–370, 1985.

[25] Hector Garcia-Molina and Daniel Barbara. How to assign votes in a distributed system. *J. ACM*, 32(4):841–860, 1985.

[26] David K. Gifford. Weighted voting for replicated data. In *Proceedings of the 7th ACM symposium on Operating systems principles*, pages 150–162, December 1979.

[27] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[28] Seth Gilbert and Grzegorz Malewitz. The quorum deployment problem. In *8th International Conference on Principles of Distributed Systems*, pages 316–330, 2004.

[29] Garth Goodson, Jay Wylie, Gregory Ganger, and Michael Reiter. Efficient Byzantine-tolerant erasure-coded storage. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 135–144, 2004.

[30] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 BFT protocols. In *Proceedings of the 5th ACM SIGOPS/EuroSys European Conference on Computer Systems*, pages 363–376, 2010.

[31] Rachid Guerraoui, Ron R. Levy, and Marko Vukolić. Lucky read/write access to robust atomic storage. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 125–136, June 2006.

[32] Rachid Guerraoui and Marko Vukolić. How Fast Can a Very Robust Read Be? In *Proceedings of the 25th ACM Symposium on Principles of Distributed Computing*, pages 248–257, July 2006.

[33] Rachid Guerraoui and Marko Vukolić. Refined quorum systems. *Distributed Computing*, 2010. http://dx.doi.org/10.1007/s00446-010-0103-7.

[34] Anupam Gupta, Bruce M. Maggs, Florian Oprea, and Michael K. Reiter. Quorum placement in networks to minimize access delays. In *Proceedings of the 24th annual ACM symposium on Principles of distributed computing*, pages 87–96, New York, NY, USA, 2005. ACM.

[35] Maurice Herlihy. A quorum-consensus replication method for abstract data types. *ACM Trans. Comput. Syst.*, 4(1):32–53, 1986.

[36] Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *Proceedings of the 16th annual ACM symposium on Principles of distributed computing*, pages 25–34, 1997.

[37] Ron Holzman, Yosi Marcus, and David Peleg. Load balancing in quorum systems. *SIAM J. Discret. Math.*, 10(2):223–245, 1997.

[38] T. Ibaraki and T. Kameda. A theory of coteries: Mutual exclusion in distributed systems. *IEEE Trans. Parallel Distrib. Syst.*, 4(7):779–794, 1993.

[39] Prasad Jayanti, Tushar Deepak Chandra, and Sam Toueg. Fault-tolerant wait-free shared objects. *Journal of the ACM*, 45(3):451–500, 1998.

[40] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative Byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4):1–39, 2009.

[41] Leslie Lamport. On interprocess communication. *Distributed computing*, 1(1):77–101, May 1986.

[42] Leslie Lamport. Fast Paxos. *Distributed Computing*, 19(2):79–103, 2006.

[43] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[44] Nancy A. Lynch. *Distributed Algorithms*. Morgan-Kaufmann, 1996.

[45] M. Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. *ACM Trans. Comput. Syst.*, 3(2):145–159, 1985.

[46] Dahlia Malkhi. *Quorum Systems. The Encyclopedia of Distributed Computing,* J. Urban and P. Dasgupta, eds., Kluwer Academic, 2000.

[47] Dahlia Malkhi and Michael Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[48] Dahlia Malkhi, Michael Reiter, Avishai Wool, and Rebecca Wright. Probabilistic quorum systems. *Inf. Comput.*, 170(2):184–206, 2001.

[49] Dahlia Malkhi and Michael K. Reiter. Secure and scalable replication in phalanx. In *Proceedings of the 17th Symposium on Reliable Distributed Systems*, pages 51–58, 1998.

[50] Dahlia Malkhi, Michael K. Reiter, and Avishai Wool. The load and availability of Byzantine quorum systems. *SIAM J. Comput.*, 29(6):1889–1906, 2000.

[51] J-P. Martin, L. Alvisi, and M. Dahlin. Small Byzantine quorum systems. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 374–383, June 2002.

[52] Jean-Philippe Martin and Lorenzo Alvisi. Fast Byzantine consensus. *IEEE Trans. Dependable Secur. Comput.*, 3(3):202–215, 2006.

[53] Jean-Philippe Martin, Lorenzo Alvisi, and Michael Dahlin. Minimal Byzantine storage. In *Proceedings of the 16th International Conference on Distributed Computing*, pages 311–325, October 2002.

[54] Michael G. Merideth and Michael K. Reiter. *Selected Results from the Latest Decade of Quorum Systems Research.* In Bernadette Charron-Bost, Fernando Pedone, and André Schiper, editors, Replication: Theory and Practice, LNCS, vol. 5959, pages 185-206. Springer, 2010.

[55] Michael G. Merideth and Michael K. Reiter. Probabilistic opaque quorum systems. In *Proceedings of the 21st International Conference on Distributed Computing*, pages 403–419, 2007.

[56] Moni Naor and Avishai Wool. The load, capacity, and availability of quorum systems. *SIAM J. Comput.*, 27(2):423–447, 1998.

[57] David Peleg and Avishai Wool. The availability of quorum systems. *Inf. Comput.*, 123(2):210–223, 1995.

[58] David Peleg and Avishai Wool. Crumbling walls: A class of practical and efficient quorum systems. *Distributed Computing*, 10(2):87–97, 1997.

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

[60] HariGovind V. Ramasamy and Christian Cachin. Parsimonious asynchronous Byzantine-fault-tolerant atomic broadcast. In *Proceedings of the 9th International Conference on Principles of Distributed Systems*, pages 88–102, December 2005.

[61] Rodrigo Rodrigues, Petr Kouznetsov, and Bobby Bhattacharjee. Large-scale Byzantine fault tolerance: safe but not always live. In *Proceedings of the 3rd workshop on on Hot Topics in System Dependability*, page 17, Berkeley, CA, USA, 2007. USENIX Association.

[62] Robert H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4(2):180–209, 1979.

[63] Werner Vogels. Eventually consistent. *Commun. ACM*, 52(1):40–44, 2009.

[64] Avishai Wool. Quorum systems in replicated databases: Science or fiction? *Bulletin of the IEEE Technical Committee on Data Engineering*, 21:3–11, 1998.

[65] Haifeng Yu. Signed quorum systems. *Distributed Computing*, 18(4):307–323, 2006.

[66] Piotr Zielinski. Optimistically terminating consensus: All asynchronous consensus protocols in one framework. In *Proceedings of The 5th International Symposium on Parallel and Distributed Computing*, pages 24–33, 2006.