Distrib. Comput. (2001) 14: 41–48

DISTRIBUTED COMPUTING

Access cost for asynchronous Byzantine quorum systems

Rida A. Bazzi

Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287–5406, USA (e-mail: bazzi@asu.edu)

Received: September 1999 / Accepted: September 2000

Summary. Quorum systems have been used to implement many coordination problems in distributed systems. In this paper, we study the cost of accessing quorums in asynchronous systems. We formally define the asynchronous access cost of quorum systems and argue that the asynchronous access cost and not the size of a quorum is the right measure of message complexity of protocols using quorums in asynchronous systems. We show that previous quorum systems proposed in the literature have a very high asynchronous access cost. We propose a reformulation of the definition of Byzantine quorum systems that captures the requirement for non-blocking access to quorums in asynchronous systems. We present new Byzantine quorum systems with low asynchronous access cost whose other performance parameters match those of the best Byzantine quorum systems proposed in the literature. In particular, we present a construction for the disjoint failure pattern that outperforms previously proposed systems for that pattern.

Key words: Quorum – Fault tolerance – Byzantine failures – Distributed systems – Asynchronous – Access cost

1 Introduction

A quorum system is a collection of sets (quorums) that mutually intersect. Quorum systems have been used to implement mutual exclusion [1,9], replicated data systems [8], commit protocols [17], and distributed consensus [13]. Work on quorum systems traditionally considered crash failures [1,2,5–9, 16,15]. Malkhi and Reiter [10] proposed the interesting notion of Byzantine quorums–quorum systems that can tolerate Byzantine failures. They presented protocols to implement a distributed shared register variable using Byzantine quorums. Their implementation requires a client accessing a quorum to wait for responses from every server in a quorum set, but they did not study the problem of finding a quorum set whose elements are available – an available quorum.

In this paper, we study the cost of finding an available quorum in the presence of Byzantine failures. We consider both the direct access model in which processes access a quorum in one round of communication, and the incremental access model in which processes can access a quorum in multiple rounds of communication. We formally define the asynchronous access cost of quorum systems, and we argue that the asynchronous access cost and not the size of a quorum is the right measure of message complexity of protocols using quorums in asynchronous systems. We also show that previous quorum systems proposed in the literature have a very high asynchronous access cost.

We propose a reformulation of the definition of Byzantine quorum systems that captures the requirement for non-blocking access to quorums in asynchronous systems. We introduce *non-blocking Byzantine quorum systems* and show that they can be achieved at a low cost and we present non-blocking Byzantine quorum constructions for two failure models. The constructions we present are the first that do not require blocking and have a low cost of access. Also, the construction we present for the disjoint failure model yields a Byzantine quorum system that has better performance parameters than previously proposed systems. Our constructions rely on a new access model we call *partial access*. With partial access, a processor need not wait for a reply from each process in a quorum set; the quorum system should be designed to ensure that any two partial accesses have a large enough intersection to ensure consistency. It turns out that the set of partial accesses of a non-blocking Byzantine quorum system is a Byzantine quorum system as defined in [10].

The rest of the paper is organized as follows. Section 2 discusses related work and Sect. 3 summarizes our contributions. Section 4 presents the definitions and introduces the notion of asynchronous access cost. Section 5 gives examples of the asynchronous access cost for two Byzantine quorum systems. Section 6 reformulates the definition of Byzantine quorums to capture the asynchronous access cost as a design objective. Section 7 presents non-blocking quorum systems with low asynchronous access cost and whose other performance

This material is based upon work supported by the National Science Foundation under Grant No. CCR-9876052. An earlier version of this paper appears in the *Proceedings of the Thirteenth International Conference Conference on Distributed Computing, Bratislava, Slovakia, 1999*.

parameters match those of the best Byzantine quorum systems proposed in the literature. Section 8 concludes the paper.

2 Related work

The problem of finding an available quorum has been addressed by researchers for the case of detectable crash failures [2,12,15]. In [15], the *probe complexity* of a quorum system is defined. The probe complexity is the minimum number of processors that need to be contacted to establish the existence or non-existence of an available quorum in the system. In the definition of probe complexity, processors can be probed incrementally and the identity of the processor to be probed next can depend on the responses received from previous probes. In [12], Neilsen proposes a dynamic probe strategy that improves on the results of [15]. In [2], the author formally defined the concept of *cost of failures*, which can be thought of as the probe complexity per failure.

Both [2] and [15] assume that failures can be detected. Their incremental access methods are not directly applicable to asynchronous systems in which failures cannot be detected.

The problem of finding an available quorum in the presence of Byzantine failures has not been studied by other researchers. Due to the nature of Byzantine failures and system asynchrony, the definition of Byzantine quorum systems proposed in [10] requires that an available quorum exists in the system. Unfortunately, that requirement does not say anything about the cost of finding an available quorum. The availability requirement of Byzantine quorum systems was relaxed in [3] for systems in which timeouts can be used to detect failures. In such systems, the author shows that any quorum set Q can be accessed without a need to access servers that do not belong to Q.

3 Contributions

To our knowledge, this is the first work that studies the cost of accessing Byzantine quorum systems in asynchronous systems. We consider both the direct access model and the incremental access model. We introduce non-blocking Byzantine quorum systems and provide necessary and sufficient conditions for their existence. Unlike Byzantine quorum systems, non-blocking Byzantine quorum systems capture the asynchronous access cost. In that respect, they are similar to synchronous Byzantine quorums [3].

We propose optimal non-blocking quorum systems and show that they are not equivalent to previously proposed Byzantine quorum systems. For the disjoint failure pattern, we propose a non-blocking quorum system that yields the best known Byzantine quorum system for that failure model.

4 Definitions and system model

4.1 System model

We assume that the system consists of a set $\mathcal{P}$ of n server processors and a number of client processors that are distinct from the servers. All processors can communicate using reliable message passing. We assume that there are no bounds on message delivery time or on processors' speeds and that there are no failure detectors in the system.

4.2 Failure model

Server processors can fail.[1] The assumptions about failures affect the way a quorum can be used. In this paper, we consider systems in which processors are subject to Byzantine failures; i.e., they can deviate arbitrarily from their protocols. In such systems it is usually assumed that there are bounds on the number of failures that can occur in the system. Such bounds have traditionally been expressed with a number t that is an upper bound on the number of failures that can occur in the system. This model was later generalized in [10] to allow more flexibility in describing the failure patterns that the system can exhibit. We adopt the model of [10] in this paper.[2] The set $faulty$ denotes the set of faulty processors in the system. A *failure pattern* $\mathcal{F}$ identifies the possible sets of faulty processors in the system. We write $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$. There exists an element F of $\mathcal{F}$ such that at any given instant, the faulty processors belong to F. The processors do not necessarily know F. A common example of a failure pattern is the *f-threshold* pattern in which $\mathcal{F} = \{F \in \mathcal{P} : \ |F| = f\}$. Another interesting failure pattern is the *disjoint* pattern in which all elements of $\mathcal{F}$ are disjoint [10].

4.3 Quorum and set systems

Definition 1 *A set system $\mathcal{S}$ over $\mathcal{P}$ is a subset of $2^{\mathcal{P}}$.*

In what follows, we assume that all set systems are over $\mathcal{P}$. A quorum system is a particular type of set systems.

Definition 2 *A* quorum system *$\mathcal{Q}$ over $\mathcal{P}$ is a set of subsets (called* quorums*) of $\mathcal{P}$ such that any two quorums have a non-empty intersection.*

The intersection property of quorums is essential for their use in coordination problems.

Processors access a quorum to coordinate their actions. Typically, to access a quorum, a client sends a request to every server, in a quorum set. Upon receiving a request, a correct server updates its state and sends a reply. The client waits until it receives a reply from every server in the quorum. If the client receives replies from all processors in a quorum, then the access is considered successful. If one of the servers failed, then the client attempts to access another quorum that does not have any faulty processor (the question of finding a quorum with no faulty processors has been addressed in [2, 15] for systems with detectable failures). Since processors access a quorum only if all its members are correct, two clients are always guaranteed to receive a response from a common correct server that belongs to a non-empty intersection of two quorums. The correctness of quorum-based protocols rely on this intersection property. A quorum is said to be *available* if all its elements are correct processors [14].

[1] We do not consider client failures in this paper.

[2] In a long-lived system, bounds on failure will be exceeded at some point. We do not address the problem of recovering failed servers in this paper.

4.4 Byzantine quorums in asynchronous systems

If failures are arbitrary, a processor might receive conflicting replies from faulty and correct processors. It follows that a processor must base its coordination decisions on replies that it *knows* to be from correct processors. Motivated by this requirement, Malkhi and Reiter gave the following definition [10]:

Definition 3 *A quorum system tolerates failure pattern $\mathcal{F}$ if*

1. $\forall\, Q_1, Q_2 \in \mathcal{Q}\ \ \forall\, F_1, F_2 \in \mathcal{F} : (Q_1 \cap Q_2) - F_1 \not\subseteq F_2$.
2. $\forall F\ \in \mathcal{F}\ \ \exists\, Q \in \mathcal{Q} : F \cap Q = \emptyset$

The first condition requires that the intersection of two quorums is not contained in the union of two sets in $\mathcal{F}$. This guarantees that the reply of some correct processor can be identified .

The second condition is the availability condition (also called resiliency requirement in [11]). It requires that some quorum consists of correct processors. The availability condition is needed in asynchronous systems because there is no way to differentiate a slow processor from a faulty one. To access a quorum in an asynchronous system, a processor cannot simply send requests to all processors in a quorum set and wait for replies. In the worst case, even in a failure-free execution, a processor might have to send requests to every processor in the system and then wait for replies from a quorum that consists of correct processors (we give an example below). The availability condition is needed to ensure that some quorum is available in the system.

We generalize the definition of [10] and define what it means for a set system to be resilient to a failure pattern $\mathcal{F}$.

Definition 4 *A set system $\mathcal{S}$ is resilient to failure pattern $\mathcal{F}$ if* $\forall\, F \in \mathcal{F}\ \exists\, S \in \mathcal{S}\ :\ S \cap F = \emptyset$.

The work of Malkhi and Reiter [10] and their subsequent work [11] does not address the problem of ensuring that a response is received other than by requiring that the system is resilient to the failure pattern. Addressing this problem is an important contribution of this paper.

4.5 Cost of access

In this section we introduce the *asynchronous access cost* of a quorum system. We first discuss the effects of asynchrony on accessing a quorum system, then we define our access model and introduce the *cost of access* parameter for evaluating quorum systems.

4.5.1 Asynchrony

In this paper, we consider the problem of accessing quorums in systems that are fully asynchronous and in which there are no bounds on message delivery delays or the speed of processors. In such systems, one way to guarantee replies from some quorum is to send requests to every processor in the system and then wait for replies from some quorum. Obviously, sending requests to every processor is too costly and eliminates the benefits of using quorum systems. Our aim is to improve on the worst-case scenario in which every server needs to be contacted to guarantee a response from some quorum set.

4.5.2 Access models

In this paper, we concentrate on the *direct access model* in which processes access a quorum by sending all requests at once and then wait for replies. The direct access model is not the most general model of accessing quorums. For instance, a process might incrementally access a quorum system by sending requests to some processes, then send further requests based on the replies it receives. We also consider *incremental access strategies* and show that an incremental strategy would require more than one round of message exchange which can be prohibitively high in a fully asynchronous system.[3] The definition of direct access strategy that we present assumes that all clients use the same strategy. The definition can be modified to allow different clients to have different strategies.

In what follows, we will talk about accessing a set system instead of accessing a quorum system. We start by defining access sets.

Definition 5 *Let S be an element of set system $\mathcal{S}$ that is resilient to $\mathcal{F}$. A set A is an* access set *of S with respect to $\mathcal{F}$ if $S \subseteq A$ and*

$$\forall\, F \in \mathcal{F}\ \exists\, S' \in \mathcal{S}\ :\ S' \subseteq A - F$$

Note that a set S might have more than one access set for a given set system $\mathcal{S}$. Also, an access set A might be the access set of more than one set S. In fact, we could have defined an access set independently of the set S and have it depend only on $\mathcal{S}$. We chose to define access sets as a function of S to emphasize the fact that it is the set S that is the target of the access. While a particular request to an access set of S will not ensure a reply from all elements of S, such a reply can be expected under favorable delay conditions (obviously, this is not guaranteed in asynchronous systems). Also, a request to access set of S can ensure replies from some elements of S, which might be desirable, under certain conditions[4].

Definition 6 *A* direct access strategy *for set system $\mathcal{S}$ is a mapping $A_d : \mathcal{S} \mapsto 2^{\mathcal{P}}$ that assigns for each element S of $\mathcal{S}$ an access set of S.*

By sending requests to $A_d(S)$, a client is guaranteed to receive replies from some quorum set. The quorum set from which replies are received is not necessarily S, but it can be S.

We present incremental access strategies informally because our results do not require a formal definition. In an incremental access strategy, a process need not contact all servers of an access set at once. It can contact some servers and then depending on the replies, it decides what servers to contact next. By avoiding a commitment to one access set at the outset, it is conceivable that with an incremental access strategy a smaller number of servers need to be contacted to force a reply from a quorum. This will be at the cost of extra message exchanges.

[3] A more detailed study of the cost of access of the incremental strategy is a subject for future research.

[4] A more detailed discussion of the advantages of of making the definition dependent on S is beyond the scope of this paper.

4.5.3 Cost of access

Definition 7 *Let S be an element of set system $\mathcal{S}$ that is resilient to $\mathcal{F}$. The* asynchronous access cost *of S is the size of the smallest access set of S with respect to $\mathcal{F}$.*

Note that by definition, a Byzantine quorum that tolerates failure pattern $\mathcal{F}$ is also resilient to $\mathcal{F}$. It follows that the asynchronous access cost is well defined for all quorum sets of a Byzantine quorum system.

Definition 8 *The* asynchronous access cost *of a set system $\mathcal{S}$ that is resilient to failure pattern $\mathcal{F}$ is*

$$cost(\mathcal{S}) = \min\{|A| \;:\; \forall\, F \in \mathcal{F} \exists\, S \in \mathcal{S} \;:\; S \subseteq A - F\}$$

In the direct access model, the asynchronous access cost gives the minimum number of servers that need to be contacted to ensure that a response is received from some set in a set system.

As the following theorem shows, in a fully asynchronous system, a client needs to send requests to $cost(\mathcal{S})$ servers each time it needs to successfully access a set system.

Theorem 9 *Let $\mathcal{S}$ be a set system that is resilient to $\mathcal{F}$. In the direct access model, a client needs to send requests to $cost(\mathcal{S})$ servers to guarantee a response from each server in some set in $\mathcal{S}$.*

Proof. In the direct access model, a client c sends all requests at the beginning to some set of servers A. If $|A| < cost(\mathcal{S})$, then, by definition of the asynchronous access cost, there is a faulty set $F \in \mathcal{F}$ such that $A - F$ contains no element of $\mathcal{S}$. If processes in F fail, c will not receive a response from every server in any element of $\mathcal{S}$. □

Corollary 10 *In the incremental access model, if less than $cost(\mathcal{S})$ are contacted in the first round of communication, then at least two rounds of message are needed to guarantee a successful access.*

Proof. By Theorem 9, if less than $cost(\mathcal{S})$ servers are contacted in the first round of an incremental access strategy, then no successful access is guaranteed. It follows that at least two rounds are needed for a successful access. □

4.6 Strategies and load

This section presents the formal definitions of strategy and load as in [14]. It discusses the implications of the asynchronous access cost on the load of a quorum system.

A protocol using a quorum system chooses a quorum to access according to some rules. A strategy is a probabilistic rule to choose a quorum. Formally, a strategy is defined as follows.

Definition 11 *Let $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ be a quorum system. A* strategy *$w \in [0,1]^m$ for $\mathcal{Q}$ is a probability distribution over $\mathcal{Q}$.*

For every processor $q \in \mathcal{P}$, a strategy w induces a probability that q is chosen to be accessed. This probability is called the load on q. The *system load* is the load of the *busiest* element induced by the best possible strategy.

Definition 12 *Let w be a strategy for a quorum system $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$. For any $q \in \mathcal{P}$, the* load *induced by w on q is $l_w(q) = \Sigma_{q \in Q_j} w_j$. The* load *induced by w on $\mathcal{Q}$ is*

$$\mathcal{L}_w(\mathcal{Q}) = \max_{q \in \mathcal{P}} l_w(q)$$

The system load *on $\mathcal{Q}$ is*

$$\mathcal{L}(\mathcal{Q}) = \min_{w}\{\mathcal{L}_w(\mathcal{Q})\},$$

where the minimum is taken over all strategies.

The definition of load implicitly assumes that no extra servers need to be contacted when a particular quorum is accessed. This is not the case for Byzantine failures in asynchronous systems because extra servers need to be accessed even in failure-free runs to guarantee a response (assuming clients do not know that the run is failure-free). From the discussion about the cost of access, it follows that the load definition should take the cost of access into consideration. The definition of load can simply be changed by replacing a quorum with the access set of the quorum, while allowing for different access sets for the same quorum at different times. If the only access set of any quorum is the set $\mathcal{P}$ of all servers, it follows that the load is 1, regardless of the quorum size.

5 Asynchronous access cost examples

In this section we give examples of the asynchronous access cost for two Byzantine quorum systems. The first system, the *Paths* system, has optimal quorum size and load (as traditionally defined) combination and high availability in the presence of crash failures. We show that it has a large asynchronous access cost. The second system, the threshold system, has small asynchronous access cost relative to the size of its quorums.

5.1 Paths system

The *Paths* system [11] is defined for the f-threshold failure pattern. It is defined as follows. Let $n = d^2$, be the number of servers arranged in a square grid of the triangular lattice. A quorum consists of $\sqrt{2f+1}$ non-intersecting top-bottom paths and $\sqrt{2f+1}$ non-intersecting left-right paths. In [11], it is shown that any two quorums intersect in $2f+1$ distinct vertices and that the *Paths* system can tolerate no more than $\sqrt{n}$ failures.

The *Paths* systems has small quorum size, small load (not taking access cost into consideration) and high availability in the presence of crash failures. Unfortunately, the asynchronous access cost of the *Paths* system is high as we show below.

Lemma 13 $cost(Path) = \Omega((f + \sqrt{f+1})d)$.

Proof. Let L and R be the sets consisting of the vertices of the left and right edges of the square grid. If a set A is of size $|A| < (f + \sqrt{f+1})d$, then A cannot contain more than $(f + \sqrt{f+1}) - 1$ disjoint paths that connect L and R because each left-right path is of size at least d. By Menger's theorem [4], it follows that there is a set C of vertices of size less than $(f + \sqrt{f+1})$ that separates L and R. Removing f

vertices from C yields a set set A' with cut set of size at most $\sqrt{f+1}-1$. Again, by Menger's theorem, A' cannot have $\sqrt{f+1}$ disjoint paths. □

In particular, the access cost of the *Paths* system is very high if $f = \Omega(d)$.

Corollary 14 *If $f = \Omega(d)$, then $cost(Path) = \Omega(n)$.*

5.2 Threshold system

The threshold system is defined for the f-threshold failure pattern. A quorum of the threshold system consists of any set of size $f + \lceil \frac{n+1}{2} \rceil$. Any two quorums are guaranteed to intersect in at least $2f+1$ elements. For the threshold system, the cost of the system is not much different from the quorum size, but the quorum size is large.

Lemma 15 *The asynchronous access cost of the threshold system is $O(2f + \lceil \frac{n+1}{2} \rceil)$.*

Proof. In fact, a set A of size $2f + \lceil \frac{n+1}{2} \rceil$ vertices is guaranteed to contain a quorum if any f elements are removed from A. Also, a set A of size less than $2f + \lceil \frac{n+1}{2} \rceil$ vertices is not guaranteed to contain a quorum if f elements are removed from A. □

6 Non-blocking quorum systems

The goal of defining *non-blocking quorum systems* is to emphasize the importance of the asynchronous access cost as a design parameter, and to provide a more uniform definition of Byzantine quorum systems. In the examples above, there is no clear relationship between the cost of access and the size of the quorum. Our aim is to reformulate the definition of the quorum system so that the cost of access is found as part of the design of the quorum system and not after the quorum is already designed. So, instead of designing the quorum sets, we directly design the access sets.

We define a non-blocking Byzantine quorum system as follows.

Definition 16 *A set system $\mathcal{Q}$ is a* non-blocking masking quorum system *that tolerates failure pattern $\mathcal{F}$ if and only if:*

$$\forall\, Q_1, Q_2 \in \mathcal{Q}\ \ \forall\, F_1, F_2, F_3, F_4 \in \mathcal{F} : \\ ((Q_1 - F_1) \cap (Q_2 - F_2) - F_3) \not\subseteq F_4$$

We define partial access sets of a non-blocking Byzantine quorum system as follows.

Definition 17 *The* partial access sets *of a non-blocking Byzantine quorum system $\mathcal{Q}$ that tolerates failure pattern $\mathcal{F}$ are the sets of the form $Q - F$, where $Q \in \mathcal{Q}$ and $F \in \mathcal{F}$.*

Note that the definition of non-blocking masking quorum systems is similar to the first condition of Definition 3. In fact, given a non-blocking quorum system $\mathcal{Q}$, the set of partial access sets of $\mathcal{Q}$ form a Byzantine Quorum system as defined by Malkhi and Reiter in [10]. Nevertheless, finding a non-blocking quorum system corresponding to a particular Byzantine quorum system is not always straightforward. Furthermore, not every Byzantine quorum system is equal to the set of access sets of a non-blocking Byzantine quorum system.

Our definition non-blocking Byzantine quorum systems requires that the client be able to determine a correct response from any partial access set; to have successful partial accesses, it is enough to guarantee that the quorum system can handle the worst-case failure scenario. To access a quorum Q, a client sends requests to all servers in Q, and then waits for a response from all servers in a partial access set of Q. Such a response is guaranteed by the definition of non-blocking Byzantine quorum systems. Once a response from a partial access set is received, the client can proceed as in [10].

The following theorem gives a sufficient condition for a collection of sets to be a non-blocking Byzantine quorum system.

Theorem 18 *A set $\mathcal{Q}$ is a* non-blocking quorum system *that tolerates failure pattern $\mathcal{F}$ if:*

1. $\forall\, Q_1, Q_2 \in \mathcal{Q}\ \ \forall\, F_1, F_2 \in \mathcal{F}\ :\ (Q_1 \cap Q_2) - F_1 \not\subseteq F_2$, *and*
2. $\forall\, Q_1 \in \mathcal{Q}\ \ \forall\, F_1, F_2 \in \mathcal{F} \exists\, Q_2 \in \mathcal{Q}\ :\ Q_2 \subseteq (Q_1 - F_1) \cup F_2$.

Proof. The proof is by contradiction. Let Q_1 and Q_2 be two quorum sets such that $\exists\, F_1, F_2, F_3, F_4 \in \mathcal{F} : ((Q_1 - F_1) \cap (Q_2 - F_2) - F_3 \subseteq F_4$. It follows that $(Q_1 - F_1) \cap (Q_2 - F_2) \subseteq F_3 \cup F_4$ and $((Q_1 - F_1) \cup F_3) \cap ((Q_2 - F_2) \cup F_4) \subseteq F_3 \cup F_4$. By Condition 2 of the theorem, it follows that there exists two quorum sets Q and Q' such that $Q \cap Q' \subseteq F_3 \cup F_4$. This contradicts Condition 1 of the theorem. □

As we saw, to each non-blocking Byzantine quorum system corresponds a Byzantine quorum system as defined in [10]. The importance of the reformulation lies in the fact that it ties the quorum size to the asynchronous access cost. When designing a non-blocking quorum system, one would have to design a quorum system with small quorums sets (all other parameters being equal), which means that the resulting asynchronous access cost is small. This is due to the fact that the quorums of a non-blocking quorum system are access sets of the underlying Byzantine quorum system. On the other hand, when designing Byzantine quorum systems as defined in the formulation of [10], the asynchronous cost of access is not directly related to the quorum size even if the quorum systems have good performance parameters. One might argue that it is always possible to construct Byzantine quorum systems with low access cost and without the need for a reformulation of the definition. Intuitively, we believe that this is not true unless the access cost is an explicit design parameter, in which case one has to use a definition similar to ours. Also, we believe that our definition provides a natural expression of the access cost as a design parameter. Finally, previous Byzantine quorum constructions in the literature did not take the access cost into account as we saw in Sect. 5.

6.1 Existence of non-blocking quorum systems

Given a failure pattern, we are interested in deciding whether there exists a quorum system that tolerates the failure pattern. The following two propositions give necessary and sufficient conditions.

Proposition 19 *There exists a non-blocking quorum system that tolerates failure pattern $\mathcal{F}$ if and only if $\mathcal{Q} = \{\mathcal{P}\}$ tolerates $\mathcal{F}$.*

Proof. If $\mathcal{Q} = \{\mathcal{P}\}$ tolerates $\mathcal{F}$, then there exists a quorum system that tolerates $\mathcal{F}$. If there exists a quorum system $\mathcal{Q}$ that tolerates $\mathcal{F}$, then there exists a quorum in $\mathcal{Q}$ that cannot be contained in the union of less than five elements of $\mathcal{F}$. It follows that $\mathcal{P}$ is not equal to the union of less than five elements in $\mathcal{F}$ and that $\{\mathcal{P}\}$ tolerates $\mathcal{F}$. □

Proposition 20 *There exists a non-blocking quorum system that tolerates failure pattern $\mathcal{F}$ if and only if*

$$\forall\, A, B, C, D \in \mathcal{F} : \quad \mathcal{P} \neq A \cup B \cup C \cup D.$$

Proof. Direct application of Proposition 19. □

For the case of the f-threshold failure pattern, we get the following corollary.

Corollary 21 *There exists a quorum system that tolerates the f-threshold failure pattern if and only if $n \geq 4f + 1$.*

It is interesting to note that the necessary and sufficient condition for the existence of a non-blocking Byzantine quorum system is also a necessary and sufficient condition for the existence of a quorum system [10]. This is expected, given the extremal nature of the system used in the proofs. In fact, the access sets of $\mathcal{Q} = \{\mathcal{P}\}$ are the Byzantine quorum system used in [10] to prove the necessary condition for Byzantine quorum systems. While the necessary conditions are identical to those for Byzantine quorum systems, this does not necessarily imply a correspondence between Byzantine quorum system and non-blocking Byzantine quorum systems. In fact, we will show at the end of Sect. 7.1 that some Byzantine quorum systems are not equal to the set of partial access sets of any non-blocking Byantine quorum system. So, it was conceivable that a Byzantine quorum system can be constructed in a system in which no non-blocking system could be constructed.

7 Non-blocking quorum systems constructions

Depending on the failure patterns, constructing a non-trivial non-blocking Byzantine quorum systems can be harder than constructing a Byzantine quorum system (the set $\mathcal{Q} = \{\mathcal{P}\}$ is a trivial system if a non-blocking quorum system exist). In this section we present two constructions of non-blocking Byzantine quorum systems, one for the threshold failure pattern and one for the disjoint failure pattern.

7.1 Threshold failure pattern

Consider a system of $n = d^2$ processors arranged in a $d \times d$ square grid, $d > 4$ in the presence of the f-threshold failure pattern, $f < (n-1)/4$. Two vertices (x_1, y_1) and (x_2, y_2) of the grid are connected if: $x_1 = x_2 \;\wedge\; y_1 = y_2 + 1$, $x_1 = x_2 \;\wedge\; y_2 = y_1 + 1$, $x_1 = x_2 + 1 \;\wedge\; y_1 = y_2$, $x_2 = x_1 + 1 \;\wedge\; y_1 = y_2$, $x_1 = x_2 + 1 \;\wedge\; y_1 = y_2 + 1$, or $x_2 = x_1 + 1 \;\wedge\; y_2 = y_1 + 1$.

Similar to the *Paths* system [11], we define a non-blocking quorum system $\mathcal{Q}_{fn}$ that consists of $2\lceil\sqrt{f+1}\rceil$ disjoint left-right paths and $2\lceil\sqrt{f+1}\rceil$ disjoint top-bottom paths. Using the same arguments as in [11], it is easy to show that any two quorums are guaranteed to intersect in $4f + 1$ elements. It follows that the quorum system is a non-blocking quorum system.

The quorum system $\mathcal{Q}_{fn}$ has better fault tolerance than the *Paths* system. In fact, $\mathcal{Q}_{fn}$ can tolerate $\lfloor(n-1)/4\rfloor$ failures (Corollary 21), whereas the *Paths* system can tolerate no more than $\sqrt{n}$ failures in the worst case. The reason is that the *Paths* system requires each row to have a number of available vertices for the system to be available. In contrast, a partial access of $\mathcal{Q}_{fn}$ is successful if all but f members of a quorum respond. This can be achieved even if less than $2\sqrt{f+1}$ nodes are available in a given row, whereas the *Paths* system will be unavailable if $\sqrt{2f+1}$ or less nodes are available in a given row. Furthermore, if $2\sqrt{f+1} \leq f$ (i.e. $f \geq 6$), then the non-blocking quorum system will tolerate the failure of a whole row which is not possible for the *Paths* system. If $f \geq 7$, then $\mathcal{Q}_{fn}$ can tolerate the failure of a whole row and a whole column.

More importantly, the cost of accessing $\mathcal{Q}_{fn}$ is $4\lceil\sqrt{f+1}\rceil d$, compared to $\Omega((f + \sqrt{f+1})d)$ for the *Paths* system.

The load of $\mathcal{Q}_{fn}$ is $\approx 4\sqrt{\frac{f+1}{n}}$. The proof is identical to that given for the load of the *Paths* systems given in [11]. The load of $\mathcal{Q}_{fn}$ is larger than the load of the *Paths* which is equal to $\approx 2\sqrt{\frac{2f+1}{n}}$. The load given for $\mathcal{Q}_{fn}$ holds even when taking the cost of access into account; the load for the *Paths* is much higher than the load of $\mathcal{Q}_{fn}$ when taking the cost of access into account.

In this paper we assume that the system is subject to only Byzantine failures. For such systems, failures are constrained by a failure pattern and we do not calculate the failure probability in this model.

The Byzantine quorum system $\mathcal{Q}_f = \{Q - F \;:\; Q \in \mathcal{Q}_{fn} \text{ and } F \in \mathcal{F}\}$ induced by $\mathcal{Q}_{fn}$ is not directly related to the *Paths* system. In fact, many quorums of $\mathcal{Q}_f$ do not contain a quorum of the *Paths* system and vice versa. Also, $\mathcal{Q}_f$ has better fault tolerance than the *Paths* system and the quorums of the *Paths* system are smaller than those of $\mathcal{Q}_{fn}$. This example shows the advantage of using the definition of non-blocking quorum systems.

The difference between the two quorum systems is illustrated in Fig. 1a. In the figure, accessing the non-blocking quorum will be successful even if all but one node in a given row fail. In contrast, the *Paths* system will be unavailable if less than three nodes are available in a given row.

Finally, we note that the *Paths* system is not equal to the set of partial access sets of any non-blocking Byzantine quorum system. In fact, let $\mathcal{Q}_{nb}$ be a non-blocking quorum system and let Q be a quorum of the *Paths* system which is a partial access set for a quorum in $\mathcal{Q}_{nb}$. It follows that for some $Q' \in \mathcal{Q}_{nb}$ $Q' - F = Q$, where $|F| = t$ and F is disjoint from Q. The set Q', is of size at most $|Q| + t$. By argument similar to that in the proof of Lemma 13, it is easy to show that for appropriate n and t we can find a set F', $|F'| = t$, such that $Q' - F'$ contains no quorum of the *Paths* system.

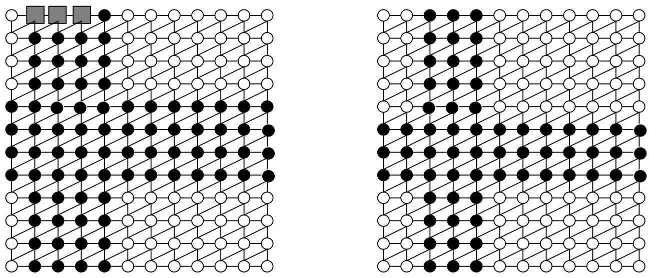

Fig. 1a. Access set of a non-blocking Byzantine quorum for $n = 144$ and $f = 3$ with one quorum in black (the gray squares belong to access set but not to the quorum). Note that the maximum number of independent vertical paths in the quorum set is one; **b** A quorum of the *Paths* system. Note that it contains no quorum set of the non-blocking quorum system

7.2 Disjoint failure pattern

In this section we provide an efficient construction of a non-blocking quorum system $\mathcal{Q}_{dn}$ for a failure pattern $\mathcal{F}$ whose elements are disjoint. The construction is similar to the construction given in [3]. We present the construction in some detail to show that our definition of non-blocking Byzantine quorum systems yields quorum systems that are not easily designed for the original definition. In fact, the Byzantine quorum system $\mathcal{Q}_d$ defined by the partial access sets of $\mathcal{Q}_{dn}$ outperforms the ones proposed in [10] for the disjoint failure pattern. Also, we do not know how to express $\mathcal{Q}_d$ other than as the set of partial access sets of $\mathcal{Q}_{dn}$. We do not provide proofs for most of our claims because they are almost identical to those of [3].

Let $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ be the set of failure sets ordered in decreasing size. We assume without loss of generality that $m > 4$. Let $\alpha = n - (\Sigma_{i=1}^{4}|F_i|)$.

Our construction will proceed as follows. First we show that there are five disjoint sets of size greater than $\frac{\alpha}{10}$ such that no two of them will have an non-empty intersection with the same faulty set. Then, on each of the five sets S_i, $i = 0, \ldots, 4$, we construct a traditional quorum system whose load is $O(\frac{1}{\sqrt{|S_i|}})$. On $\mathcal{P}$, we construct a quorum system whose elements consist of the union of five quorums, one from each of the five sets.

Let $m_0 = 4$ and define m_i, $i = 1, \ldots, 4$ as follows:

$$m_i = \min\left\{ j \,:\, |F_i \;\cup \bigcup_{m_{i-1}<k\le j+1} F_k| \ge \frac{\alpha}{10} \right\}$$

Note that j and k are bound variables in the definition of m_i. Also, m_i, $1 \le i \le 4$, are always guaranteed to exist.

Now, define the five sets S_i, $0 \le i \le 4$, as follows:

- $S_i = F_i \;\cup\; \bigcup_{m_i<k\le m_{i+1}} F_k$, if $i < 4$, and
- $S_4 = \bigcup_{m_4<k} F_k$

Proposition 22 $S_i \cap S_j = \emptyset$, $0 \le i \ne j \le 4$.

Proposition 23 $S_i \ge \frac{\alpha}{10}$ *for* $0 \le i \le 4$.

Now, we describe the non-blocking quorum system $\mathcal{Q}_{dn}$. On $S_i, 0 \le i \le 4$, define a quorum system with load $O(\frac{1}{\sqrt{|S_i|}})$. Many such systems exist. One such system is the triangle lattice system [2]. In the triangle lattice system over S_i, we can choose $\sqrt{2|S_i|}$ quorums such that each processor belongs to exactly two quorums. If we choose each quorum with a probability $\frac{1}{\sqrt{2|S_i|}}$, it follows that the load of the triangle lattice system on S_i is at most $\frac{2}{\sqrt{2|S_i|}} = \sqrt{\frac{2}{|S_i|}}$. Define a quorum on $\mathcal{P}$ to be the union of five quorum sets one from each of the triangle lattices defined on S_i, $0 \le i \le 4$.

Proposition 24 *The resulting system* $\mathcal{Q}_{dn}$ *is a non-blocking quorum system that tolerates* $\mathcal{F}$.

Proof. In fact, any two quorums intersect in five servers, no two of which belong to the same faulty set. Therefore the intersection of two quorums does not belong to the union of four faulty sets. □

Let $\mathcal{Q}_d$ be the Byzantine quorum system defined by the access sets of $\mathcal{Q}_{dn}$. $\mathcal{Q}_d = \{Q - F \,:\, Q \in \mathcal{Q}_{dn}$ and $F \in \mathcal{F}\}$. It follows that $\mathcal{Q}_d$ tolerates the disjoint failure pattern $\mathcal{F}$.

Proposition 25 *The load of the quorum system* $\mathcal{Q}_d$ *is* $O(\frac{1}{\sqrt{\alpha}})$.

Lemma 26 *The load of* $\mathcal{Q}_d$ *is optimal.*

Proof. The proof uses the same techniques as those used in [3] and is omitted. □

The load of $\mathcal{Q}_d$ stated above is the traditional load. Nevertheless, the load of $\mathcal{Q}_{dn}$ is of the same order as the load of $\mathcal{Q}_d$ and is therefore optimal. Finally, we know of no simpler way to express $\mathcal{Q}_d$ or to obtain a Byzantine quorum system that tolerate $\mathcal{F}$ and has a better load.

8 Conclusion

An important contribution of this paper is the recognition that in asynchronous systems, it is not enough to design a quorum systems with small quorum sets, but it is more important to design a quorum set that is amenable to efficient access. We proposed a new definition of Byzantine quorum systems that lend themselves to non-blocking access. We have shown that designing non-blocking Byzantine quorums with small access cost is possible in asynchronous systems.

One might think that the only difference between Byzantine quorum systems and non-blocking Byzantine quorum systems is that of how access is achieved, and that it is possible to come up with a new definition of access for Byzantine quorum systems to achieve non-blocking access. While this is possible, the resulting non-blocking quorum system is not guaranteed to have a small access cost. To design Byzantine quorum system with a small access cost, we believe that a formulation similar to ours is needed.

Acknowledgements. I thank the reviewers for many useful and detailed suggestions.

References

1. D. Agrawal, A. El–Abbadi: An efficient and fault-tolerant solution for distributed mutual exclusion. ACM Trans Comput Syst 9(1): 1–20 (1991)

2. R.A. Bazzi: Planar Quorums. Theor Comput Sci 243: 243–268 (2000)
3. R.A. Bazzi: Synchronous Byzantine quorum systems. Distrib Comput 13(1): 45–52 (2000)
4. B. Bollobás: Graph Theory, An Introductory Course. Graduate Texts in Mathematics, Berlin Heidelberg New York: Springer 1979
5. H. Garcia-Molina, D. Barbara: How to assign votes in a distributed system. Journal of the ACM 32(4): 481–860 (1985)
6. D.K. Gifford: Weighted Voting for Replicated Data. Proceeding of 7th ACM Symposium on Operating Systems Principles, pp 150–162, December 1979
7. A. Kumar: Hierarchical quorum consensus: A new algorithm for managing replicated data. IEEE Trans Comput 40(9): 996–1004 (1991)
8. A. Kumar, M. Rabinovich, R. Sinha: A performance study of general grid structures for replicated data. In: Proceedings of International Conference on Distributed Computing Systems, pp 178–185, May, 1993
9. M. Maekawa: A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. ACM Trans Comput Syst 3(2): 145–159 (1985)
10. D. Malkhi, M. Reiter: Byzantine Quorum Systems. Distrib Comput 11(4): 203-213 (1998)
11. D. Malkhi, M. Reiter, A. Wool: The load and availability of Byzantine quorum systems. In: Proceedings of the Sixteenth ACM Symposium on Principles of Distrib Comput, August, 1997
12. M.L. Neilsen: A Dynamic probe strategy for quorum systems. In: Proceedings of the IEEE 17th International Conference on Distrib Comput Systems, pp 95–99, 1997
13. M.L. Neilsen: Quorum Structures in Distributed Systems. Ph.D. Thesis, Department of Computer and Information Sciences, Kansas State University, 1992
14. M. Naor, A. Wool: The Load, capacity and availability of quorum systems. SIAM J Comput 27(2): 423–447 (1998)
15. D. Peleg, A. Wool: How to be an Efficient Snoop, or the Probe Complexity of Quorum Systems. In: Proceedings of the 15th ACM Symposium on Principles of Distrib Comput, pp 290–299, 1996
16. D. Peleg, A. Wool: The availability of quorum systems. Inform Comput 123(2): 210–223 (1995)
17. D. Skeen: A quorum–based commit protocol. In: Proceedings of 6th Berkeley Workshop on Distributed Data Management and Computer Networks, pp 69–80, 1982

Rida A. Bazzi received a B.E. in Computer and Communications Engineering from the American University of Beirut in 1989, and an M.Sc. and a Ph.D. in Computer Science from Georgia Institute of Technology in 1994. He is currently an assistant professor in the Computer Science and Engineering Department at Arizona State University. His major research interests are distributed computing, fault tolerance, and security.