Distrib. Comput. (2006)
DOI 10.1007/s00446-005-0133-8

SPECIAL ISSUE PODC 04

Haifeng Yu

Signed quorum systems

Received: 6 August 2004 / Accepted: 2 July 2005 / Published online: 13 January 2006

Abstract With n servers that independently fail with probability of $p < 0.5$, it is well known that the majority quorum system achieves the best availability among all quorum systems. However, even this optimal construction requires $(n+1)/2$ functioning servers out of n. Furthermore, the number of probes needed to acquire a quorum is also lower bounded by $(n+1)/2$.

Motivated by the need for a highly available and low probe complexity quorum system in the Internet, this paper proposes *signed quorum systems* (SQS) that can be available as long as any $O(1)$ servers are available, and simultaneously have $O(1)$ probe complexity. SQS provides probabilistic intersection guarantees and exploits the property of *independent mismatches* in today's Internet. Such key property has been validated previously under multiple Internet measurement traces. This paper then extensively studies the availability, probe complexity, and load of SQS, derives lower bounds for all three metrics, and constructs matching upper bounds. We show that in addition to the qualitatively superior availability and probe complexity, SQS also decouples availability from load and probe complexity, so that optimal availability can be achieved under most probe complexity and load values.

Keywords Quorum systems · Availability · Probe complexity · Load · Tradeoff

1 Introduction

Quorum systems are well known techniques to achieve mutual exclusion, preserve consistency on replicated data, improve availability, and share load. A *quorum system* is a set of *quorums*, each of which itself is a subset of *servers* from a universe of servers. It is guaranteed that any two quorums intersect. To perform an action potentially conflicting with other actions, a *client* coordinates the action with a quorum. System availability is improved because the system is available whenever a single quorum is available, while per-server load is reduced because a client no longer needs to contact all servers.

There have been three major measures of goodness for quorum systems: *availability*, *probe complexity*, and *load*. Intuitively, availability is the probability that the system has at least one live quorum. Probe complexity describes the number of messages (or probes) a client needs to send in order to acquire a quorum. Finally, load is the probability of accessing the busiest server (i.e., what fraction of the overall load is sustained by the busiest server).

For traditional quorum systems, it is known [2] that when servers fail independently (in fail-stop fashion) with probability of $p < 0.5$, the majority quorum system [15] (where a quorum is any majority subset of all servers) achieves the best availability. However, even in this optimal quorum system, the system needs to have $(n+1)/2$ functioning servers to be available. Furthermore, the probe complexity is also lower bounded by $(n+1)/2$. By guaranteeing intersection only with high probability, probabilistic quorum systems (PQS) [9] overcome such fundamental limitation on availability and probe complexity. However, the PQS construction in [9] still needs $\theta(\sqrt{n})$ functioning servers and similar probe complexity. Traditional quorum systems also have fundamental tradeoffs [11] among availability, probe complexity and load:

$$1\text{-Availability} \geq p^{n \times \text{Load}} \tag{1}$$

$$1\text{-Availability} \geq p^{\text{Probe Complexity}} \tag{2}$$

$$\text{Load} \geq 1/\text{Probe Complexity} \tag{3}$$

Such fundamental tradeoffs prevent us from obtaining the best of multiple measures simultaneously.

This paper is motivated by the need for a highly-available and low probe complexity quorum system in the Internet. For distributed systems in the wide-area environment, management is typically decentralized and nodes are less well maintained, resulting in larger p values. Yet it is still important to achieve good availability with a small

H. Yu (✉)
Intel Research Pittsburgh and Carnegie Mellon University
E-mail: yhf@cs.cmu.edu

number of servers. The cost of wide-area communication in the Internet is also much more expensive than local processing, making it crucial to bound probe complexity.

1.1 Our results

To qualitatively improve the availability and probe complexity of previous quorum techniques, this paper proposes *signed quorum systems* (SQS) that can be available as long as any $O(1)$ servers are available, and simultaneously have $O(1)$ probe complexity. Furthermore, SQS breaks the tradeoff between availability and load/probe complexity (Inequality 1 and 2) – optimal availability can now be achieved under most load and probe complexity values.

In signed quorum systems, a quorum may contain negative elements. For example, over a three server universe $\{1, 2, 3\}$, a signed quorum system can be $\{\{-1, 3\}, \{1, -2, -3\}\}$. In the quorum of $\{-1, 3\}$, the element "3" is interpreted same as before. Namely, the client needs to obtain a reply from server 3. The element "-1" means that the client believes that server 1 has failed, according to some inaccurate failure detection mechanism. Since failure detection can be inaccurate, it is possible that another client can still obtain a reply from server 1. Such an event is called a *mismatch*.

Following the arguments of PQS, SQS also provides probabilistic guarantee on intersection. However, SQS does not achieve such guarantee via an explicit access strategy [9]. Rather, SQS is designed for servers randomly distributed across the Internet, and we make a key assumption[1] on the independence of mismatches *on different servers*. If mismatches happen independently for different servers with probability of ε, then the probability of one client acquiring $\{-1, 3\}$ and another client acquiring $\{1, -2, -3\}$ is at most ε^2. This is true because two mismatches are needed on server 1 and 3. We define a *dual pair* to be a pair in the form of $\{i, -i\}$, and the number of dual pairs between two quorums is called their *dual overlap*. The previous two quorums thus have a dual overlap of two (from the dual pairs of $\{-1, 1\}$ and $\{3, -3\}$). In SQS, it is required that any two quorums either intersect over some positive element, or their dual overlap is at least 2α, where α is a positive constant. As a result, if quorums Q and Q' do not intersect on any positive element, then the probability that they can be both acquired is as small as $\varepsilon^{2\alpha}$.

To intuitively understand the validity of our key assumption, notice that when mismatches are strongly correlated for servers randomly distributed in the Internet, they tend to indicate the existence of a "hard" partition. However, recent network measurements and evaluation [1, 14, 18] have shown that "hard" partitions where a significant fraction of Internet nodes are unable to communicate with the rest of the network, are rare in today's Internet. A client with a lost network connection can also observe correlated mismatches,

[1] Section 2.2 will show that in fact, PQS also needs to make similar assumptions in asynchronous systems.

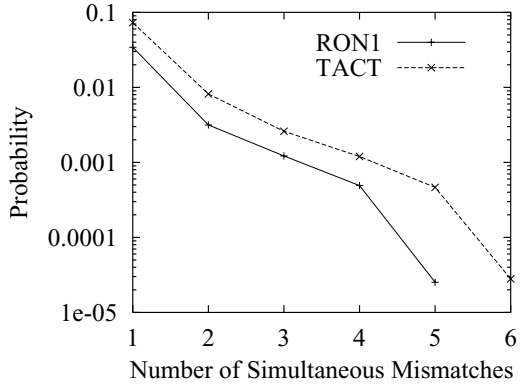

Fig. 1 Sample results from [17] to validate the assumption of independent mismatches. Both curves are near-linear, indicating independence mismatches

but we have previously shown [17] that a simple filtering step can be quite effective to prevent those clients from acquiring any quorum. The filtering step requires a client to reach some node outside of its local domain (i.e., to test whether it simply has lost its network connection) before it is allowed to acquire a quorum.

Validating this key assumption of independent mismatches was one focus of our previous work [17] on the witness model, an implicit (non-optimal) SQS construction. Our results based on the RON traces [1] from MIT and the TACT [18] trace from Duke show that the average correlation among mismatches is below 5%. Figure 1 plots some sample results. Notice that strictly speaking, mismatches will not be *fully* independent. However, from a practical perspective, all we need is that the probability of having 2α simultaneous mismatches is small.

Continuing from our earlier efforts, this paper proposes and formalizes the concept of SQS, and then extensively studies the availability, probe complexity, and load of SQS. Specifically, we make the following contributions:

- We propose and formalize the concept of SQS, which generalizes traditional quorum systems. SQS provides probabilistic intersection guarantees and the probability of non-intersection is exponentially small with 2α.
- We study the availability of SQS and prove that our $\mathcal{OPT}_a$ construction has optimal availability, where the system is available as long as any α servers are available. Such availability is not possible under traditional quorum systems or PQS (with non-zero intersection guarantee).
- For the set of SQS with optimal availability, we derive a lower bound for probe complexity. A matching upper bound is achieved by our $\mathcal{OPT}_d$ construction that has the same availability as $\mathcal{OPT}_a$. Specifically, the expected probe complexity of $\mathcal{OPT}_d$ is smaller than $2\alpha/(1-p) = O(1)$ under arbitrary n values. Such probe complexity is again, not possible under traditional quorum systems or PQS (with non-zero intersection guarantee).

Table 1 Important SQS constructions in this paper. We also indicate whether the achieved behavior is optimal

SQS construction	Availability	Expected probe complexity	Load
$\mathcal{OPT}_a$	available if any α out of n servers are available (optimal)	n	1
$\mathcal{OPT}_d$	same as $\mathcal{OPT}_a$	$< 2\alpha/(1-p)$ (optimal conditioned upon optimal availability)	1
SQS constructed from composition	same as $\mathcal{OPT}_a$	x, where x is tunable and $x = \Omega(\alpha)$ and $x = O(\sqrt{n})$	$O(1/x)$ (optimal condition upon probe complexity being x)

- We design a powerful *composition* technique that allows us to compose certain traditional quorum system $\mathcal{Q}$ over k servers with $\mathcal{OPT}_a$ over n servers ($k \leq n$). We prove that the composition result is an SQS with $\mathcal{OPT}_a$'s availability, and $\mathcal{Q}$'s probe complexity and load. This shows that SQS breaks the fundamental tradeoff between availability and load/probe complexity (Inequality 1 and 2). Now optimal availability can be achieved all the time, as long as probe complexity is $\Omega(\alpha)$.
- We show that a similar load and probe complexity tradeoff (Inequality 3) exists for SQS. For any probe complexity value x, $x = \Omega(\alpha)$ and $x = O(\sqrt{n})$, we show that composition allows us to construct an SQS with load of $O(1/x)$[2] and optimal availability.

Table 1 summarizes the major SQS constructions in this paper. The next section discusses related work, and we introduce the formal definitions of SQS, availability, load, and probe complexity in Sect. 3. Section 4 formalizes the notation of mismatches and rigorously argues why our SQS definition is able to bound the probability of non-intersection. We construct SQS with optimal availability, optimal probe complexity and optimal load in Sects. 5, 6, and 7. Finally, Sect. 8 draws our conclusions.

2 Related work

This work is a continuation from our earlier work [17] on the *witness model*, which is an implicit non-optimal SQS construction. There we focus on validating the assumption on independent mismatches and the application of the witness model to distributed consensus. The general concept of SQS, their availability, probe complexity and load properties are not studied in [17].

2.1 Strict quorum systems

Most traditional quorum systems are *strict* in the sense that they always guarantee quorum intersection. Within such context, there have been extensive research efforts [2,4–7,11–13,15] on quorum systems, and we only focus on those closely related to this paper.

Most previous work [2, 12] on the availability of quorum systems uses the same availability definition as ours, namely, the probability that at least one live quorum exists given that servers fail independently. The probe complexity of quorum systems is first studied in [13], where only worse-case probe complexity for deterministic probe algorithms is considered. Later, Hassin et al. [6] study the average probe complexity for deterministic algorithms and the worse-case probe complexity for randomized algorithms. Another measure for probe complexity is *the cost of failure* [3], defined as the normalized number of extra probes needed due to failures.

Load balancing in quorum systems is first studied in [7], where the metric considered is the load ratio between the busiest and least-loaded server. Naor et al. [11] define and study load as the probability of accessing the busiest server. Naor et al. further prove a tradeoff between availability and load (Inequality 1). The other two tradeoffs (Inequality 2 and 3) also directly follow from their proof.

Quorum systems have also been used to mask Byzantine failures [8]. To mask such failures, it is required that any two quorums intersect on sufficient number of servers so that the client will not be misled by malicious servers. In comparison, SQS only tolerates fail-stop failures and dual overlap is for the number of dual pairs, instead of direct intersection.

2.2 Probabilistic quorum systems

SQS follows PQS's spirit of providing probabilistic guarantees on quorum intersection. In PQS, such guarantee is achieved by enforcing an access strategy on quorums. For example, in one PQS construction [9], quorums are all the sets of size $l\sqrt{n}$ and each quorum is accessed with equal probability. This guarantees an intersection probability of at least $1 - e^{-l^2}$.

Compared to PQS, it may appear that SQS makes an additional strong assumption of independent mismatches. In asynchronous systems, however, we argue that implementing PQS needs to make similar assumptions as well. The fundamental reason is that with an asynchronous scheduler, we do not have full control over the access strategy and the intersection guarantee of PQS can be disrupted by the scheduler.

Following consider a concrete example with two servers (1 and 2), and two clients (x and y). Construct a PQS

[2] Load itself has a lower bound of $\Omega(1/\sqrt{n})$.

$\mathcal{Q} = \{\{1\}, \{2\}, \{1, 2\}\}$ and an access strategy that simply chooses each quorum in $\mathcal{Q}$ with probability of 1/3. A simple calculation can show that intersection happens with probability of 7/9. A straightforward implementation of the previous PQS would be simply to have every client try to acquire each quorum with probability of 1/3. However, suppose that the asynchronous scheduler delays all messages from x to server 2. Since x cannot wait forever, it is forced to use the quorum of {1} all the time. Similarly, the scheduler can force y to (ultimately) always choose and use the quorum of {2}. At this point, the actual access strategy is already different from what we intend, and the intersection probability is actually zero. A closer look reveals that the problem is exactly caused by mismatches on both 1 and 2. Thus to implement PQS, some assumption needs to be made on mismatches to limit the power of the asynchronous scheduler.

In SQS, such assumption on mismatches is explicitly made and validated. Given that the access strategy can be disturbed by the scheduler, it is no longer a convenient way to define SQS. Instead, we directly impose a requirement on dual overlap. Ultimately, there is still a distribution (access strategy) on the quorums that are used in a given SQS. However, such distribution is determined by the scheduler and failures in the system. The dual overlap requirement ensures probabilistic intersection under any of the possible resulting distributions.

It is interesting to see that after clearly stating our assumption of independent mismatches and defining SQS based on dual overlap, we are able to achieve an availability and probe complexity not possible under PQS. In summary, it seems to us that implementing PQS in asynchronous systems needs an implicit assumption on mismatches, but since the assumption is implicit, PQS has not fully exploited its power.

3 Preliminaries and definitions

3.1 Unsigned and signed quorum systems

We consider a *universe* $U = \{1, 2, \ldots, n\}$ of servers. An *unsigned set system* over the universe U is a set of subsets of U. For simplicity, we will drop the phrase "over the universe U" in the following discussion.

Definition 1 An *(unsigned) quorum system* (UQS) $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$ is an unsigned set system where $Q_i \cap Q_j \neq \emptyset$ $(1 \leq i, j \leq m)$. Q_i $(1 \leq i \leq m)$ is called a *quorum*.

For any element $i \in U$, its *dual* is $-i$, and the dual of $-i$ is i. For any set $S \subseteq U$, define its *dual* (denoted as $\text{Dual}(S)$) to be $\{\text{Dual}(i) | i \in S\}$. For any set $S \subseteq (U \cup \text{Dual}(U))$, define its *positive part* (denoted as S^+) to be $S \cap U$, and its *negative part* (denoted as S^-) to be $S \cap \text{Dual}(U)$.

Definition 2 A *signed set system* over the universe U is a set of subsets of $U \cup \text{Dual}(U)$, where for any set S in the signed set system, $S \cap \text{Dual}(S) = \emptyset$.

Definition 3 For a given positive integer α, a *signed quorum system* (SQS) $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$ is a signed set system where for any Q_i and Q_j, $1 \leq i, j \leq m$, at least one of the following two conditions is satisfied:

Intersection: $Q_i^+ \cap Q_j^+ \neq \emptyset$
Dual Overlap: $|Q_i \cap \text{Dual}(Q_j)| \geq 2\alpha$

Since $|Q_i \cap \text{Dual}(Q_j)| = |\text{Dual}(Q_i) \cap Q_j|$, we do not need to have two alternative definitions for dual overlap. By definition, any UQS is also an SQS. When $n < 2\alpha$, it is impossible to satisfy dual overlap. Thus we assume $n \geq 2\alpha$ for the remainder of the paper. Some of our results are obtained under a stronger assumption of $n \geq 3\alpha - 1$ or $n \geq 3\alpha + 1$, and we will state those stronger assumptions in individual theorems. In SQS, any quorum must have at least one positive element, since otherwise the quorum can satisfy neither intersection nor dual overlap with itself.

3.2 Availability

Next, we define three measures of quality for quorum systems: *availability*, *load*, and *probe complexity*. Throughout this paper, we assume that each server fails independently with probability of p, $p < 0.5$. Availability (informally) is the probability that the system has at least one live quorum. Following we formalize such definition using set operations.

Definition 4 A *configuration* of the system is a set C, such that for any $i \in U$, either i or $-i$ (but not both) is in C.

Each configuration captures a unique state of all the servers. The element i belongs to the configuration if server i is available, and $-i$ belongs to the configuration otherwise. Defining a configuration to be a set instead of a binary vector allows the use of set operations for all our reasoning.

Let $\mathcal{C}$ denote the set of all 2^n configurations. For convenience, define $\mathcal{C}_i = \{C | C \in \mathcal{C} \text{ and } |C^+| = i\}, 0 \leq i \leq n$.

Definition 5 For any quorum Q_i in an SQS $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$, its *acceptance set* (denoted as $As(Q_i)$) is $\{C | C \in \mathcal{C} \text{ and } Q_i \subseteq C\}$. The *acceptance set* of $\mathcal{Q}$ (denoted as $As(\mathcal{Q})$) is $\cup_{1 \leq i \leq m} As(Q_i)$. A signed set system is called an *acceptance set* if it is the acceptance set of some SQS.

Clearly, if $Q_i \subseteq C$, it means that the quorum Q_i can be acquired under C. Each configuration C has a certain probability of occurring: $\text{Prob}[C] = p^{|C^-|}(1-p)^{|C^+|}$. Availability is then the probability of all configurations under which some quorum can be acquired.

Definition 6 For any SQS $\mathcal{Q}$, its availability (denoted as $Avail(\mathcal{Q})$) is defined as $\sum_{C \in As(\mathcal{Q})} \text{Prob}[C]$.

3.3 Probe complexity

Informally, probe complexity is the number of probes a client needs to make in order to acquire a quorum, or to

realize that no live quorum exists. Notice that to acquire a quorum in SQS, the client also needs to probe (and fail to obtain a reply) from those servers corresponding to the negative elements in the quorum.

To formalize, we first define the probe strategy used by the client to determine which server to probe next:

Definition 7 Probe Strategy *[13]*: A probe strategy is a binary tree. Each non-leaf node of the tree is labeled with a server, and two outgoing edges are marked by the two possible outcomes of the probe, namely, success or failure. Each leaf of tree denotes an outcome of the probe algorithm, which can either be that a quorum has been acquired or that no live quorums exist.

A probe strategy is *non-adaptive* if for any node in the probe tree, both its children are labeled with the same server. In other words, a non-adaptive probe strategy does not adjust the probe sequence based on the results of earlier probes.

Let $\mathcal{Q}$ be an SQS and let Ψ be the set of all probe strategies for $\mathcal{Q}$. For any configuration $C \in \mathcal{C}$ and any probe strategy $\psi \in \Psi$, define $path(\psi, C)$ to be ψ's branch corresponding to C, and define $depth(\psi, C)$ to be the length of $path(\psi, C)$. The *expected probe complexity for deterministic probe algorithms* (denoted as $PC_e(\mathcal{Q})$) [6] is then:

$$PC_e(\mathcal{Q}, \psi) = \sum_{C \in \mathcal{C}} depth(\psi, C) \cdot \text{Prob}[C]$$
$$PC_e(\mathcal{Q}) = \min_{\psi \in \Psi}\{PC_e(\mathcal{Q}, \psi)\}$$

The *worst-case probe complexity for deterministic probe algorithms* (denoted as $PC_w(\mathcal{Q})$) [13] is defined as:

$$PC_w(\mathcal{Q}, \psi) = \max_{C \in \mathcal{C}}\{depth(\psi, C)\}$$
$$PC_w(\mathcal{Q}) = \min_{\psi \in \Psi}\{PC_w(\mathcal{Q}, \psi)\}$$

Randomized probe strategies are necessary to achieve any non-trivial load. A randomized probe algorithm has a distribution μ over Ψ. Let $PC_e^*(\mathcal{Q})$ denote the *expected probe complexity for randomized probe algorithms*:

$$PC_e^*(\mathcal{Q}, \mu) = \sum_{C \in \mathcal{C}} E_\mu[depth(\psi, C), \psi \in \Psi] \cdot \text{Prob}[C]$$
$$PC_e^*(\mathcal{Q}) = \min_{\mu}\{PC_e^*(\mathcal{Q}, \mu)\}$$

It can be easily shown that $PC_e(\mathcal{Q}) = PC_e^*(\mathcal{Q})$ for any $\mathcal{Q}$, and thus we will focus on $PC_e^*(\mathcal{Q})$ only. The *worse-case probe complexity for randomized probe algorithms* (denoted as $PC_w^*(\mathcal{Q})$) [6] is:

$$PC_w^*(\mathcal{Q}, \mu) = \max_{C \in \mathcal{C}}\{ E_\mu[depth(\psi, C), \psi \in \Psi] \}$$
$$PC_w^*(\mathcal{Q}) = \min_{\mu}\{PC_w^*(\mathcal{Q}, \mu)\}$$

For practical purposes, perhaps PC_e^* is the most important metric, since it describes the expected message complexity for acquiring a quorum.

3.4 Load

In most previous work [9, 11], load is defined based on the *access strategy* of a quorum system. The access strategy is a distribution over all quorums describing the probability that each individual quorum is used. Load is then defined as the probability of accessing the busiest server (as determined by the access strategy).

However, to acquire a quorum Q, we may potentially probe and induce load on more servers than those in Q, which makes the previous definition inaccurate. Thus this paper uses a more practical definition directly based on probe strategy. Such definition is part of our contribution.

For any node a in the probe strategy tree ψ, define its *load* to be:

$$\sum_{C \in \mathcal{C} \text{ and } path(\psi, C) \text{ contains } a} \text{Prob}[C]$$

Intuitively, the load defined above is simply the probability of reaching node a (by multiplying the respect p and $(1-p)$ terms down the tree to a). Server i's *load* is the sum of the load of those tree nodes labeled i. With a deterministic probe algorithm, the root of the probe tree always has a load of 1.0. Thus, we are mainly interested in the load for randomized probe algorithms. Let $\mathcal{Q}$ be an SQS, Ψ be the set of all probe strategies for $\mathcal{Q}$, and μ be a distribution over Ψ, we define $\mathcal{Q}$'s load (denoted as $Load(\mathcal{Q})$) to be:

$$\min_{\mu}\left\{ \max_{1 \le i \le n} \{E_\mu[\text{server } i\text{'s load under } \psi, \psi \in \Psi]\}\right\}$$

With our new definition, the load of a quorum system is at least as high as the load under previous definitions [9, 11]. Following is a proof sketch for such claim. The access strategy τ is uniquely determined by the distribution of μ. For any given μ, if a server has a load of l under the access-strategy-based definition for the corresponding τ, then it is present in acquired quorums with probability of l. With at least such probability, it is probed by the client and will have a load of at least l under our load definition. (The load may be large than l because it may also be probed in other cases.) Finally, since we have a corresponding τ for any μ, the minimum from all μ's will not be smaller than the minimum from all τ's.

All load lower bounds in this paper are applicable to the traditional optimistic definition. (In fact, the only load lower bound in this paper is Theorem 38, which is directly adopted from [11].) On the other hand, all our load upper bounds hold under our new pessimistic definition. This makes our results as strong as possible.

Also notice that according to our definition, $Load(\mathcal{Q})$ and $PC_e^*(\mathcal{Q})$ may or may not be achieved under the same probe algorithm. But for all our probe complexity and load upper bounds, we actually use the same probe algorithm, which again makes our results strong.

4 Properly bounding the probability of non-intersection in SQS

This section proves that our simple SQS definition is sufficient to control the probability of non-intersection. We first formalize our notion of mismatch and the independent mismatch assumption. We say that a client *reaches* a server if and only if it obtains a positive reply from the server. A server probed by two clients can be in one of the following four states: i) $(-,-)$: Neither client reaches the server; ii) $(+,-)$: The first client reaches the server, but not the second client; iii) $(-,+)$: The first client does not reach the server, but the second client does; iv) $(+,+)$: Both clients reach the server. The states of $(-,+)$ and $(+,-)$ are called *mismatch*es.

We assume that mismatch on one server is independent of other servers' states. Further, we assume that Prob[mismatch | state is not $(-,-)$] $\leq \varepsilon$ for any server. This intuitively means that given that one client reaches a server, the probability that the other client cannot reach the server is at most ε. Notice that just assuming Prob[mismatch] $\leq \varepsilon$ is not sufficient. Otherwise we can let Prob$[(-,+)]$ = Prob$[(+,-)] = \varepsilon/2$, Prob$[(+,+)] = 0$, and Prob$[(-,-)] = 1-\varepsilon$, and intersection can never happen. Notice that in such a case, ε = Prob[mismatch] $\leq$ Prob[mismatch | state not$(-,-)] = 1$.

From a practical perspective, independent mismatches in the mathematical sense is not important. In fact, all we care about is that the probability of having multiple simultaneous mismatches is small. We assume pure independence here only to simplify discussion. Also, "hard" network partitions will result in dependent mismatches (as mentioned in Sect. 1), but recent network measurements and evaluation [1, 14, 18] have shown that "hard" partitions where a significant fraction of Internet nodes are unable to communicate with the rest of the network, are rare in today's Internet.

We now use an example to show that the SQS definition, solely by itself, may not yet properly bound non-intersection probability. Dual overlap between two non-intersecting quorums Q_1 and Q_2 does bound the probability that they are both acquired. However, there can be many other quorums that do not intersect with Q_1 either, and the probability of obtaining *one* of those quorums may not be low. Suppose $(n-1) = (m-1) \times 2\alpha$ and we consider the SQS $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$, where:

$$
\begin{aligned}
Q_1 &= \{1, 2, \ldots, (n-1)\} \\
Q_2 &= \{-1, -2, \ldots, -2\alpha, n\} \\
Q_3 &= \{-(2\alpha+1), -(2\alpha+2), \ldots, -4\alpha, n\} \\
&\ldots \\
Q_m &= \{-(n-2\alpha), -(n-2\alpha+1), \ldots, -(n-1), n\}
\end{aligned}
$$

Now assume that the first client acquires Q_1 and the second client reaches the last server. The second client may or may not reach the first $n-1$ servers, and there is a mismatch whenever it cannot reach a server. The independent mismatch assumption controls the probability that mismatches may occur for a *given* set of 2α servers. However, when n grows we have many such sets, and with high probability, the second client can actually find a quorum that does not intersect with Q_1.

Fortunately, we will show that under some easy-to-satisfy conditions on probe strategies and client behavior, the probability of non-intersection in an SQS is truly exponentially small with 2α. This allows us to use the simple definition of SQS and not to be concerned with the details of such probability. All the lower bounds in this paper are derived without these conditions, while all upper bounds satisfy such restrictions.

We first describe the required condition on client behavior. When a client acquires a quorum Q in an SQS using a given probe strategy, define its *probed servers* to be the set S where i belongs to S if the client reaches server i, and $-i$ belongs to S if the client does not reach server i. Clearly, S is a superset of Q. S may contain elements not in Q because the client may probe servers not belonging to the quorum ultimately acquired (i.e., wasted probes). In a traditional quorum system, a client only needs to coordinate with (e.g., read from) the servers in Q^+. In SQS, we instead require that the client coordinates with all servers in S^+. Namely, the client should coordinate with any server that it reaches during the probing process. Since the servers in S^+ are already reached by the client, such a requirement does not result in material difference from a practical perspective. In particular, it does not affect the availability, probe complexity or load of the SQS. Most protocols using quorum systems already implicitly meet this requirement. One (contrived) counterexample is when we use SQS to implement a shared register, the reader can violate this requirement by explicitly discarding the responses from servers not belonging to the acquired quorum.

With the above requirement on client, we define intersection as following:

Definition 8 Consider two clients who each acquire Q_1 and Q_2 in an SQS, respectively. Let S_1 and S_2 be the probed servers of the two clients, respectively. We say that the two clients *intersect* if and only if $S_1^+ \cap S_2^+ \neq \emptyset$.

We now prove that with the above intersection definition and a deterministic, non-adaptive probe strategy, SQS bounds the probability of non-intersection (Prob[non-intersection]) within $\varepsilon^{2\alpha}$. Our guarantee will be for the probability conditioned upon the event that both clients acquire some quorum. Since it is conditioned upon such event, the probability of intersection will actually be smaller than 1-Prob[non-intersection] (in fact, the probability of intersection is roughly (1-Prob[non-intersection]) · availability), since there will be cases where both clients do no acquire some quorum. We only consider the conditioned non-intersection probability because availability is already captured elsewhere, and also, availability is dependent on p rather than ε.

Theorem 9 *Consider two clients using the same deterministic, non-adaptive probe strategy, and let "non-intersection" denote the event that both clients acquire some quorum, but they do not intersect. Then* Prob[*non-intersection*] $\leq \varepsilon^{2\alpha}$.

Proof Let Q_1 be the random variable denoting the quorum acquired by the first client and S_1 be the probed servers. Similarly define Q_2 and S_2. Let D be the random variable denoting the set of servers that are in the $(-,-)$ state. It suffices to prove that Prob$[(S_1^+ \cap S_2^+ = \emptyset)] \leq \varepsilon^{2\alpha}$ under any D.

Since we are considering a deterministic, non-adaptive probe strategy, the clients always probe the servers in the same order. Let such order be T and next delete all servers in D from T. We argue that if $S_1^+ \cap S_2^+ = \emptyset$ (which implies $Q_1^+ \cap Q_2^+ = \emptyset$), then both clients must have probed the first 2α servers in T. The reason is that when $Q_1^+ \cap Q_2^+ = \emptyset$, Q_1 and Q_2 must satisfy dual overlap. It is impossible for D to contain any server in $Q_1 \cap \text{Dual}(Q_2)$, since the servers in D are all in state $(-,-)$. Thus every server in $Q_1 \cap \text{Dual}(Q_2)$ must be in T, and each client probes at least 2α servers in T. Finally, since T is the only probe sequence allowed, both client must have probed at least the *first* 2α servers in T.

Now consider these first 2α servers in T. For S_1^+ and S_2^+ not to intersect, there must be a mismatch on each of these 2α servers. So we have Prob$[(S_1^+ \cap S_2^+ = \emptyset)] \leq$ Prob[mismatches on first 2α servers in T] $\leq \varepsilon^{2\alpha}$. □

To understand the previous theorem, consider the example at the beginning of this section. With a deterministic and non-adaptive probe strategy, the two clients will probe the servers in exactly the same order. Further because of our extended definition of intersection, a client cannot intentionally skip any server causing intersection. This means that even though the second client will still likely acquire some Q_i $(2 \leq i \leq m)$ that does not intersect with Q_1, the probed servers of the second client will intersect with Q_1 with high probability.

The extended definition of intersection seems necessary for Theorem 9 to hold. For probe strategies, Theorem 9 only gives us sufficient conditions (i.e., "deterministic and non-adaptive") to properly bound Prob[non-intersection]. However, for all our upper bound constructions in this paper (except for the SQS in Sect. 7.2), we only need to use deterministic and non-adaptive probe strategies. The constructions in Sec. 7.2 use randomized and adaptive probe strategies to achieve a matching upper bound on load. Since the load under deterministic probe strategies is always 1.0, randomized probe strategies are necessary to achieve a matching upper bound on load. For those constructions in Sect. 7.2, we will provide a customized proof showing that Prob[non-intersection] is properly bounded.

It is yet unclear to us what are the exact sufficient and necessary conditions to properly bound Prob[non-intersection]. Following we show that the "deterministic" requirement in Theorem 9 can be dropped (this improved result, however, is not used anywhere in this paper, and we only use Theorem 9 in its exact form). We will first show that Theorem 9 holds even if the two clients use two *potentially different* deterministic, non-adaptive probe strategies (ψ_1 and ψ_2, respectively).

Lemma 10 *Consider two clients using two* potentially different *deterministic, non-adaptive probe strategies, ψ_1 and ψ_2, respectively. Then there must exist some fixed server i, such that i is* always *probed by both clients as long as they both acquire some quorum.*

Proof Let T_1 be the shortest sequence of servers that the first client probes under any configuration where it acquires some quorum. Similarly define T_2 for the second client. T_1 and T_2 must intersect. Otherwise we can construct a configuration where both clients acquire quorums, but they did not probe any server in common. This in turn, would mean that the quorums they acquire satisfy neither intersection nor dual overlap. Given that T_1 and T_2 intersect, any server in $T_1 \cap T_2$ thus is always probed by both clients as long as they both acquire some quorum. □

Lemma 11 *Consider two clients using two* potentially different *deterministic, non-adaptive probe strategies, and let "non-intersection" denote the event that both clients acquire some quorum, but they do not intersect. Then* Prob[*non-intersection*] $\leq \varepsilon^{2\alpha}$.

Proof We consider α as a variable, and define Prob[non-intersection] more precisely as the Prob[non-intersection] when 2α is used as the threshold for the dual overlap condition in SQS definition. We will show via an induction on 2α that the theorem holds under any $2\alpha \geq 0$. Here to simplify discussion, we temporarily allow 2α to take any non-negative integer values (rather than just positive even integers). This only makes our results stronger than strictly necessary.

The theorem trivially holds for $2\alpha = 0$. Suppose the theorem holds for $2\alpha = w$ (i.e. Prob[non-intersection for $2\alpha = w$] $\leq \varepsilon^w$), and let us consider the case for $2\alpha = w+1$. Lemma 10 tells us that there must exist a fixed server i that is probed by both clients (if they both acquire some quorum). Now consider the four possible states of i, and let Prob$[(+,+)] = x_1$, Prob$[(-,-)] = x_2$, Prob$[(+,-)] = x_3$, and Prob$[(-,+)] = x_4$. We have:

$$\begin{aligned}
&\text{Prob[non-intersection for } 2\alpha = w+1] \\
&\quad = x_1 \cdot 0 + x_2 \cdot \mathit{Prob}[\text{non-intersection for } 2\alpha = w+1] \\
&\quad + x_3 \cdot \mathit{Prob}[\text{non-intersection for } 2\alpha = w] \\
&\quad + x_4 \cdot \mathit{Prob}[\text{non-intersection for } 2\alpha = w]
\end{aligned}$$

Solve the above equation and we obtain:

$$\begin{aligned}
&\text{Prob[non-intersection for } 2\alpha = w+1] \\
&\quad = \frac{x_3 + x_4}{1 - x_2} \cdot \mathit{Prob}[\text{non-intersection for } 2\alpha = w]
\end{aligned}$$

The term of $(x_3+x_4)/(1-x_2)$ is exactly *Prob*[mismatch | state is not $(-,-)$], so we know that $(x_3 + x_4)/(1 - x_2) \leq \varepsilon$. Thus we have *Prob*[non-intersection for $2\alpha = w+1$] $\leq \varepsilon^{w+1}$. □

Theorem 12 *Consider two clients using non-adaptive probe strategies, and let "non-intersection" denote the event that both clients acquire some quorum, but they do not intersect. Then Prob*[*non-intersection*] $\leq \varepsilon^{2\alpha}$.

Proof We only need to consider the most general case where both clients use randomized probe strategies. A randomized non-adaptive probe strategy is modeled (Sect. 3) by each client randomly picking a strategy (according to some distribution) out of a set of deterministic non-adaptive probe strategies. Thus we only need to show that *Prob*[non-intersection] $\leq \varepsilon^{2\alpha}$ regardless of which two probe strategies the two clients pick. This is already proved by Lemma 11. □

The removal of the "deterministic" requirement on probe strategies means that we can use any randomized, non-adaptive probe strategies to achieve a load smaller than 1. However, we should also emphasize that the remaining requirement of "non-adaptive" is still not trivial. It has been proved [10] that for UQS, if the load is $O(1/\sqrt{n})$, then using non-adaptive probe strategies can at best give us a PC_e^* of $O(\sqrt{n}\log n)$. Adaptive probe strategies make it possible to reduce PC_e^* to $O(\sqrt{n})$. In fact, this is exactly the reason why our constructions in Sect. 7.2 have to use randomized *and* adaptive probe strategies, in order to match the lower bounds.

Finally, we summarize the set of (sufficient) conditions to properly bound the probability of non-intersection in SQS within $\varepsilon^{2\alpha}$:

- Mismatch on one server is independent of other servers' states.
- For any server, *Prob*[mismatch | state is not $(-,-)$] $\leq \varepsilon$.
- Clients should coordinate with all probed servers rather than only the servers in the acquired quorum.
- The probe strategy is non-adaptive.

5 SQS with optimal availability

Starting from this section, we search for the "best" SQS in terms of availability (Sect. 5), probe complexity (Sect. 6), and load (Sect. 7). We derive lower bounds for all three metrics, and achieve matching upper bounds (within constants). Our results will clearly determine the benefits we can obtain in exchange for a small probability of non-intersection.

5.1 Sufficient and necessary conditions for optimal availability

SQS relaxes from UQS by allowing dual overlap to replace intersection. This improves availability because quorums may now contain only a small number of positive elements and rely on the negative elements to satisfy the dual overlap condition. In order to find the SQS with the highest availability, we first restrict ourselves to acceptance sets. It is easy to show that acceptance sets themselves are SQS also. Furthermore, an SQS's acceptance set has the same availability as the SQS. This means that there must exist an acceptance set that can achieve optimal availability.

size $= n$ | pos $\geq \alpha$

When $n = 3$ and $\alpha = 1$, $\mathcal{OPT}_a =$

$\{1,-2,-3\}$ $\{1,2,-3\}$ $\{1,2,3\}$
$\{-1,2,-3\}$ $\{-1,2,3\}$
$\{-1,-2,3\}$ $\{1,-2,3\}$

Fig. 2 $\mathcal{OPT}_a$. All quorums have size of n and at least α positive elements

Theorem 13 *If $\mathcal{Q}$ is an SQS, then: i) $As(\mathcal{Q})$ is an SQS; ii) $As(As(\mathcal{Q})) = As(\mathcal{Q})$; iii) $Avail(\mathcal{Q}) = Avail(As(\mathcal{Q}))$.*

Proof The first two claims are obvious, and we only prove the last claim. Consider any $S \in As(As(\mathcal{Q}))$. There must exist $S' \in As(\mathcal{Q})$ and $S'' \in \mathcal{Q}$, such that $S'' \subseteq S' \subseteq S$. Since S'' belongs to $\mathcal{Q}$ and S is a configuration, we know that $S \in As(\mathcal{Q})$ and $As(As(\mathcal{Q})) \subseteq As(\mathcal{Q})$. This means $As(As(\mathcal{Q})) = As(\mathcal{Q})$. Finally, directly from the definition of availability, we have $Avail(\mathcal{Q}) = Avail(As(\mathcal{Q}))$. □

An acceptance set has the nice property that its quorums are actually configurations. As a result, the availability of an acceptance set $\mathcal{Q}$ is simply $\sum_{Q\in\mathcal{Q}} Prob[Q]$.

Since the size of any quorum in an acceptance set is n, as long as all quorums have at least α positive elements, dual overlap or intersection must be satisfied. Thus, we construct the following SQS $\mathcal{OPT}_a = \cup_{i=\alpha}^{n}\mathcal{C}_i$. $\mathcal{OPT}_a$ contains all configurations that have at least α available servers (Fig. 2). It is obvious that $\mathcal{OPT}_a$ is an SQS:

Theorem 14 *$\mathcal{OPT}_a$ is an SQS.*

It is easy to see that $\mathcal{OPT}_a$ is at least "locally optimal". Namely, we cannot add another configuration into $\mathcal{OPT}_a$ while still keeping it an SQS. We now intend to prove that $\mathcal{OPT}_a$ is also "globally optimal". The challenge is to show that availability cannot be improved by replacing some existing quorums in $\mathcal{OPT}_a$ with some new ones. We prove this by carefully grouping the new quorums to correspond to the sets of $\mathcal{C}_\alpha, \ldots, \mathcal{C}_{2\alpha-1}$. We will show that the size of each group cannot be larger than the number of deleted old quorums from the corresponding $\mathcal{C}_i$. We do this using a bipartite graph with the new quorums on the left and the configurations in the corresponding $\mathcal{C}_i$ on the right. An edge is added if two quorums cannot coexist. The claim is then proved by counting the number of edges in the graph. Finally, it is easy to show that with $p < 0.5$, any new quorum contributes less to availability than any old quorum in the corresponding $\mathcal{C}_i$. Following we formalize our arguments:

Lemma 15 *For any acceptance set $\mathcal{Q}$, if $\mathcal{C}_i \cap \mathcal{Q} \neq \emptyset$ for some $0 \leq i \leq \alpha - 1$, then $Avail(\mathcal{Q}) < Avail(\mathcal{OPT}_a)$.*

Proof Let $\mathcal{T}_i = \mathcal{C}_i \cap \mathcal{Q}$, for $0 \le i \le n$. Since $\mathcal{T}_i$ and $\mathcal{T}_j$ must be disjoint for different i and j, we know that the availability of $\mathcal{Q}$ is $\Sigma_{i=0}^{n}\Sigma_{C\in\mathcal{T}_i} Prob[C]$. Similarly, the availability of $\mathcal{OPT}_a$ is $\Sigma_{i=\alpha}^{n}\Sigma_{C\in\mathcal{C}_i} Prob[C]$.

Next, we show that $|\mathcal{T}_i| + |\mathcal{T}_{2\alpha-i-1}| \le |\mathcal{C}_{2\alpha-i-1}|$ for $0 \le i \le \alpha-1$. We construct the following bipartite graph, which contains all configurations in $\mathcal{T}_i$ (left side of the graph) and $\mathcal{C}_{2\alpha-i-1}$ (right side of the graph) as nodes. An edge is added between two vertices $C \in \mathcal{T}_i$ and $C' \in \mathcal{C}_{2\alpha-i-1}$ in the bipartite graph if and only if $C^+ \cap C'^+ = \emptyset$. Because $|C^+| = i$ and $|C'^+| = 2\alpha - i - 1$, C and C' can never satisfy the dual overlap condition. Thus adding an edge between C and C' means that they cannot both appear in $\mathcal{Q}$. Each vertex in the left part of the bipartite graph has a degree of exactly $\binom{n-i}{2\alpha-i-1}$. On the other hand, the degree of the vertices in the right part of the graph is at most $\binom{n-2\alpha+i+1}{i}$. We want to show that $\binom{n-i}{2\alpha-i-1} > \binom{n-2\alpha+i+1}{i}$:

$$\begin{aligned}
&\binom{n-i}{x} > \binom{n-x}{i} \quad (\text{where } x = 2\alpha - i - 1 > i)\\
&\Leftrightarrow (n-i)!i! > x!(n-x)!\\
&\Leftrightarrow (n-i)(n-i-1)\ldots(n-x+1)\\
&\quad > x(x-1)\ldots(i+1)
\end{aligned}$$

Both sides in the last inequality have $x - i$ terms. Since $n - i > x$ (given that $n \ge 2\alpha$), the last inequality holds, as well as all inequalities above.

With $|\mathcal{T}_i|$ vertices at the left side of the graph, the graph has altogether $|\mathcal{T}_i| \times \binom{n-i}{2\alpha-i-1}$ edges. Since each vertex at the right side of the graph has at most a degree of $\binom{n-2\alpha+i+1}{i}$, and also because $\binom{n-i}{2\alpha-i-1} > \binom{n-2\alpha+i+1}{i}$, there must be at least $|\mathcal{T}_i|$ (or $|\mathcal{T}_i| + 1$ when $\mathcal{T}_i$ is not empty) vertices with non-zero degree at the right side of the graph. None of these configurations can appear in $\mathcal{Q}$. As a result, we know that $|\mathcal{T}_{2\alpha-i-1}| \le |\mathcal{C}_{2\alpha-i-1}| - |\mathcal{T}_i|$. The "$\le$" sign becomes "$<$" if $\mathcal{T}_i$ is nonempty.

For any configuration $C \in \mathcal{C}_i$ and $C' \in \mathcal{C}_{2\alpha-i-1}$ ($0 \le i \le \alpha-1$), it must be true that $\text{Prob}[C] \le \text{Prob}[C']$ (because $p \le 0.5$ and also $2\alpha - i - 1 > i$). As a result, we have $\Sigma_{C\in\mathcal{T}_i}\text{Prob}[C] + \Sigma_{C\in\mathcal{T}_{2\alpha-i-1}}\text{Prob}[C] \le \Sigma_{C\in\mathcal{C}_{2\alpha-i-1}}\text{Prob}[C]$, where "$\le$" becomes "$<$" if $\mathcal{T}_i$ is nonempty. Now we have:

$$\begin{aligned}
Avail(\mathcal{Q}) &= \sum_{i=0}^{n}\left(\sum_{C\in\mathcal{T}_i}\text{Prob}[C]\right)\\
&= \sum_{i=0}^{\alpha-1}\left(\sum_{C\in\mathcal{T}_i}\text{Prob}[C] + \sum_{C\in\mathcal{T}_{2\alpha-i-1}}\text{Prob}[C]\right)\\
&\quad + \sum_{i=2\alpha}^{n}\sum_{C\in\mathcal{T}_i}\text{Prob}[C]\\
&< \sum_{i=\alpha}^{2\alpha-1}\sum_{C\in\mathcal{C}_i} Prob[C] + \sum_{i=2\alpha}^{n}\sum_{C\in\mathcal{C}_i} Prob[C]\\
&= \sum_{i=\alpha}^{n}\sum_{C\in\mathcal{C}_i} Prob[C] = Avail(\mathcal{OPT}_a)
\end{aligned}$$

□

Theorem 16 *For any SQS $\mathcal{Q}$, $Avail(\mathcal{OPT}_a) \ge Avail(\mathcal{Q})$.*

Proof From Theorem 13, we know that $As(\mathcal{Q})$ has the same availability as $\mathcal{Q}$. If $\mathcal{C}_i \cap As(\mathcal{Q}) \ne \emptyset$ for some i, $0 \le i \le \alpha - 1$, Lemma 15 tells us that $As(\mathcal{Q})$ has lower availability than $\mathcal{OPT}_a$. On the other hand, if $\mathcal{C}_i \cap As(\mathcal{Q}) = \emptyset$ for all i, $0 \le i \le \alpha - 1$, then $As(\mathcal{Q})$ can have availability no higher than $\mathcal{OPT}_a$. □

Corollary 17 *For any SQS $\mathcal{Q}$, if $\mathcal{OPT}_a \subseteq As(\mathcal{Q})$, then $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$.*

From Lemma 15, we can prove a stronger result, which says that we always get $\mathcal{OPT}_a$ if we "expand" any SQS with optimal availability:

Corollary 18 *For any SQS $\mathcal{Q}$, $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$ if and only if $As(\mathcal{Q}) = \mathcal{OPT}_a$.*

5.2 Non-existence of "global minimum"

Even though $\mathcal{OPT}_a$ achieves optimal availability, it has some undesirable properties. For example, all its quorums are of size n, which means n probes are needed to acquire any quorum. Are there any SQS with better properties but still preserving optimal availability? Such definition of "better" is usually defined as *domination* [4]:

Definition 19 Two SQS $\mathcal{Q}$ and $\mathcal{Q}'$, $\mathcal{Q}$ *dominates* $\mathcal{Q}'$ (denoted as $\mathcal{Q} \succ \mathcal{Q}'$) if for $\forall Q' \in \mathcal{Q}'$, $\exists Q \in \mathcal{Q}$, such that $Q \subseteq Q'$.

If $\mathcal{Q}$ dominates $\mathcal{Q}'$, it usually means that $\mathcal{Q}$ has both better probe complexity and load.

For UQS, it can be easily shown that the majority quorum system dominates any UQS that has optimal availability, which means that the majority system is the "globally minimum". If we can find such a "global minimum" for SQS, then that "minimum" will likely give us the best probe complexity and load. Interestingly however, we will show that such a "global minimum" does not exist for SQS.

To prove this result, we first study the common properties of SQS with optimal availability (illustrated in Fig. 3):

Theorem 20 *Suppose $n \ge 3\alpha - 1$. For any SQS $\mathcal{Q}$ where $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$, it must be true that:*

1. $\forall Q \in \mathcal{Q}, |Q^+| \ge \alpha$.
2. $\mathcal{C}_\alpha \subseteq \mathcal{Q}$.
3. $\forall Q \in \mathcal{Q}$, *if* $\alpha \le |Q^+| \le 2\alpha - 1$, *then* $|Q| \ge n + \alpha - |Q^+|$.
4. $\forall Q \in \mathcal{Q}, |Q| \ge 2\alpha$.

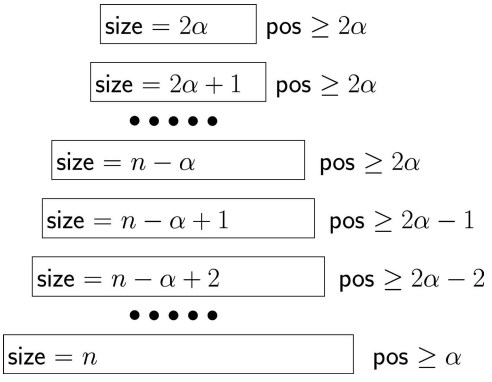

Fig. 3 Any quorum in $\mathcal{Q}$, where $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$, must be one of the forms above

Proof

1. Prove by contradiction and suppose $\exists Q \in \mathcal{Q}$, such that $|Q^+| < \alpha$. Then there must exist $C \in \mathcal{C}_{|Q^+|}$ such that $Q \subseteq C$ and $C \in As(\mathcal{Q})$. However, from Lemma 15, we know that $As(\mathcal{Q})$ (and in turn, $\mathcal{Q}$) cannot have optimal availability. Contradiction.
2. Prove by contradiction and assume $\exists C \in \mathcal{C}_\alpha$, such that $C \notin \mathcal{Q}$. Without loss of generality, suppose $C = \{1, 2, \ldots, \alpha, -(\alpha+1), \ldots, -n\}$. Since $\mathcal{Q}$ has optimal availability, Corollary 18 tells us that $C \in As(\mathcal{Q})$. Thus there exists $Q \in \mathcal{Q}$ such that $Q \subset C$. Since $|Q^+| \geq \alpha$ (from 1), again without loss of generality, assume $Q = \{1, 2, \ldots, \alpha, -(\alpha+1), \ldots, -x\}$, where $x \leq n-1$.
   Next we will find a quorum $Q' \in \mathcal{Q}$ such that Q and Q' cannot satisfy intersection or dual overlap. Because $\mathcal{Q}$ has optimal availability and $As(\mathcal{Q}) = \mathcal{OPT}_a$, we know that $\exists T \in As(\mathcal{Q})$, where $T = \{-1, -2, \ldots, -(n-\alpha), n-\alpha+1, \ldots, n\}$. Since $n \geq 3\alpha - 1$, we have $n - \alpha + 1 > \alpha$ and $T^+ \cap Q^+ = \emptyset$. Furthermore, it is clear that $|T \cap \text{Dual}(Q)| = 2\alpha + x - n \leq 2\alpha - 1$. Since $T \in As(\mathcal{Q})$, there must exist $Q' \in \mathcal{Q}$ such that $Q' \subseteq T$. Based on the property between T and Q, it is easy to see that $Q^+ \cap Q'^+ = \emptyset$ and $|Q \cap \text{Dual}(Q')| \leq 2\alpha - 1$. This means that $\mathcal{Q}$ cannot be an SQS. Contradiction.
3. Prove by contradiction and assume there exists $Q \in \mathcal{Q}$, such that $|Q^+| = x$ and $|Q| = y$, where $\alpha \leq x \leq 2\alpha - 1$ and $x \leq y \leq n + \alpha - x - 1$. Without loss of generality, suppose $Q = \{1, 2, \ldots, x, -(x+1), \ldots, -y\}$. Since $\mathcal{C}_\alpha \subseteq \mathcal{Q}$, we have $Q' \in \mathcal{Q}$, where $Q' = \{-1, -2, \ldots, -(n-\alpha), n-\alpha+1, \ldots, n\}$. Since $n \geq 3\alpha - 1$ and $\alpha \leq x \leq 2\alpha - 1$, it must be true that $n - \alpha + 1 > x$ and $Q^+ \cap Q'^+ = \emptyset$. Furthermore, because $y \leq n + \alpha - x - 1$, we have $|Q \cap \text{Dual}(Q')| = x + (y - (n-\alpha)) \leq 2\alpha - 1$. As a result, Q and Q' satisfy neither intersection nor dual overlap. $\mathcal{Q}$ is not an SQS. Contradiction.
4. From 1, we already know that $\forall Q \in \mathcal{Q}$, $|Q^+| \geq \alpha$. If $|Q^+| \geq 2\alpha$, then trivially $|Q| \geq 2\alpha$. If $\alpha \leq |Q^+| \leq 2\alpha - 1$, then from 3, we know that $|Q| \geq n + \alpha - |Q^+| \geq n - \alpha + 1 \geq 2\alpha$. □

To make our result on "global minimum" strong, we define a weaker form of domination to eliminate the effects of permutation. Doing so is necessary to make the result interesting, since otherwise an SQS may not even dominate itself after a permutation of server indexes.

Definition 21 Consider a permutation $X = (x_1, x_2, \ldots, x_n)$ of $(1, 2, \ldots, n)$. For any set $S \subseteq U \cup \text{Dual}(U)$, S's *permutation* according to X (denoted as $\text{Perm}_X(S)$) is $\{i | x_i \in S\} \cup \{-i | (-x_i) \in S\}$. An SQS $\mathcal{Q}$'s *permutation* according to X (denoted as $\text{Perm}_X(\mathcal{Q})$) is $\{\text{Perm}_X(Q) | Q \in \mathcal{Q}\}$. Two SQS $\mathcal{Q}$ and $\mathcal{Q}'$, $\mathcal{Q}$ dominates $\mathcal{Q}'$ after permutation (denoted as $\mathcal{Q} \succ_\exists \mathcal{Q}'$) if $\exists X$, such that $\mathcal{Q} \succ \text{Perm}_X(\mathcal{Q}')$.

The crux of proving the non-existence of "global minimum" is the next two SQS constructions, where no SQS can dominate both of them. In the following, $\mathcal{HOLE}$ is intuitively the set containing all size-$(n-1)$ quorums with exactly $\alpha + 1$ positive elements. Namely, there is one missing server (hence the name "hole") in any of these quorums. An important property of $\mathcal{HOLE}$ is that it remains the same after any permutation.

$$\mathcal{OPT}_b = \{\{1, 2, \ldots, 2\alpha\}\} \cup \mathcal{OPT}_a$$
$$\mathcal{HOLE} = \{S \mid |S^+| = \alpha + 1 \text{ and } |S| = n - 1 \text{ and } \exists i \text{ such that } \forall j \in U \text{ and } j \neq i, \text{ either } j \text{ or } -j \text{ (but not both) is in } S\}$$
$$\mathcal{OPT}_c = \mathcal{HOLE} \cup \mathcal{OPT}_a$$

We can easily prove that both $\mathcal{OPT}_b$ and $\mathcal{OPT}_c$ are SQS with optimal availability:

Theorem 22 *$\mathcal{OPT}_b$ is an SQS and $Avail(\mathcal{OPT}_b) = Avail(\mathcal{OPT}_a)$.*

Theorem 23 *$\mathcal{OPT}_c$ is an SQS and $Avail(\mathcal{OPT}_c) = Avail(\mathcal{OPT}_a)$.*

Proof Obviously, $\mathcal{OPT}_c$ is a signed set system. Consider any two sets S and T in $\mathcal{OPT}_c$. If both of them belong to $\mathcal{OPT}_a$, then clearly they satisfy either intersection or dual overlap. If none of them belong to $\mathcal{OPT}_a$, and $S^+ \cap T^+ = \emptyset$, then it must be true that $|S^+ \cap Dual(T^-)| \geq \alpha$. Similarly, we have $|T^+ \cap Dual(S^-)| \geq \alpha$. Thus, S and T must satisfy the dual overlap condition. Finally, if $S \in \mathcal{HOLE}$ and $T \in \mathcal{OPT}_a$, suppose $T \in \mathcal{C}_j$ $(j \geq \alpha)$. If $S^+ \cap T^+ = \emptyset$, it must be true that $|S^+ \cap Dual(T^-)| \geq \alpha + 1$ and $|T^+ \cap Dual(S^-)| \geq \alpha - 1$. Then again, S and T satisfy the dual overlap condition. Thus we have shown that $\mathcal{OPT}_c$ is an SQS. Finally, since $\mathcal{OPT}_a \subseteq As(\mathcal{OPT}_c)$, we know from Corollary 17 that $Avail(\mathcal{OPT}_c) = Avail(\mathcal{OPT}_a)$. □

Now notice that $\mathcal{OPT}_b$ contains the quorum of $\{1, 2, \ldots, 2\alpha\}$, while $\mathcal{OPT}_c$ contains the quorum of $\{-2, -3, \ldots, -(n-\alpha-1), (n-\alpha), \ldots, n\}$. When $n \geq 3\alpha + 1$, these two quorums do not satisfy either intersection or dual overlap. From this observation, we can prove that no

SQS can dominate both $\mathcal{OPT}_b$ and $\mathcal{OPT}_c$. This is true even after permutation since $\mathcal{OPT}_c$ remains unchanged after any permutation.

Theorem 24 *Suppose $n \geq 3\alpha + 1$. There does not exist an SQS $\mathcal{Q}$, such that for $\forall \mathcal{Q}'$ where $Avail(\mathcal{Q}') = Avail(\mathcal{OPT}_a)$, $\mathcal{Q} \succ_\exists \mathcal{Q}'$.*

Proof We will show that there does not exist an SQS $\mathcal{Q}$ such that $\mathcal{Q} \succ_\exists \mathcal{OPT}_b$ and $\mathcal{Q} \succ_\exists \mathcal{OPT}_c$. Prove by contradiction and suppose such $\mathcal{Q}$ exists. This means there exist two permutations X and Y such that $\mathcal{Q} \succ \text{Perm}_X(\mathcal{OPT}_b)$ and $\mathcal{Q} \succ \text{Perm}_Y(\mathcal{OPT}_c)$. Let $\overline{X}$ be X's reverse permutation, i.e., $\text{Perm}_{\overline{X}}(\text{Perm}_X(S)) = S$ for $\forall S$. Obviously, such $\overline{X}$ must exist.

Now we have:

$$\begin{aligned}&\mathcal{Q} \succ \text{Perm}_X(\mathcal{OPT}_b)\\ &\Rightarrow \text{Perm}_{\overline{X}}(\mathcal{Q}) \succ \text{Perm}_{\overline{X}}(\text{Perm}_X(\mathcal{OPT}_b))\\ &\Rightarrow \text{Perm}_{\overline{X}}(\mathcal{Q}) \succ \mathcal{OPT}_b\end{aligned}$$

and:

$$\begin{aligned}&\mathcal{Q} \succ \text{Perm}_Y(\mathcal{OPT}_c)\\ &\Rightarrow \text{Perm}_{\overline{X}}(\mathcal{Q}) \succ \text{Perm}_{\overline{X}}(\text{Perm}_Y(\mathcal{OPT}_c))\\ &\Rightarrow \text{Perm}_{\overline{X}}(\mathcal{Q}) \succ \mathcal{OPT}_c\\ &\text{(because } \mathcal{OPT}_c \text{ remains unchanged}\\ &\text{after any permutation)}\end{aligned}$$

Since $\mathcal{Q}$ is an SQS, $\text{Perm}_{\overline{X}}(\mathcal{Q})$ must also be an SQS. Let $Q_1' = \{1, 2, \ldots, 2\alpha\} \in \mathcal{OPT}_b$. There must exist $Q_1 \in \text{Perm}_{\overline{X}}(\mathcal{Q})$, such that $Q_1 \subseteq Q_1'$. Let $Q_2' = \{-2, -3, \ldots, -(n-\alpha-1), (n-\alpha), \ldots, n\} \in \mathcal{OPT}_c$. There must exist $Q_2 \in \text{Perm}_{\overline{X}}(\mathcal{Q})$, such that $Q_2 \subseteq Q_2'$.

Now we will show that with both Q_1 and Q_2, $\text{Perm}_{\overline{X}}(\mathcal{Q})$ cannot be an SQS. Since $n \geq 3\alpha + 1$, we know that $Q_1'^+ \cap Q_2'^+ = \emptyset$, which implies $Q_1^+ \cap Q_2^+ = \emptyset$. Furthermore, $|Q_1 \cap \text{Dual}(Q_2)| \leq |Q_1' \cap Dual(Q_2')| = 2\alpha - 1$. Thus, Q_1 and Q_2 satisfy neither intersection nor dual overlap, and $\text{Perm}_{\overline{X}}(\mathcal{Q})$ is not an SQS. Contradiction. □

6 SQS with optimal probe complexity

When we only optimize for probe complexity, the trivial SQS {{1}} gives the best probe complexity of 1, which is not interesting. From a practical perspective, we usually want an SQS with good availability and good probe complexity. It is reasonable to argue that "good" availability at least means $Avail(\mathcal{Q}) \to 1.0$ when $n \to \infty$.[3] It is easy to show that simply to satisfy such a weak requirement would entail at least 2α probes:

Theorem 25 *For any SQS $\mathcal{Q}$ and any probe strategy ψ for $\mathcal{Q}$, if* $\text{depth}(\psi, C) \leq 2\alpha - 1$ *for any configuration C, then* $\lim_{n\to\infty} Avail(\mathcal{Q}) \not\to 1.0$.

[3] Such property is formally called *Condorcet* in [12].

Proof We consider two cases:

1. The path corresponding to C ultimately acquires a quorum Q. Since $depth(\psi, C) \leq 2\alpha - 1$, it must be true that $|Q| \leq 2\alpha - 1$. Q thus can never satisfy the dual overlap condition with another quorum, which means that Q intersects with every quorum in $\mathcal{Q}$. If all servers in Q fail, then no quorum can be acquired. The probability that all servers in Q fail is $p^{|Q|} \geq p^{2\alpha-1}$. Thus $\lim_{n\to\infty} Avail(\mathcal{Q}) \leq 1 - p^{2\alpha-1} \not\to 1.0$.
2. The path corresponding to C ultimately claims that no quorum can be acquired under C. Suppose the path contains successful probes from the set S_1 of servers, and failed probes from the set S_2 of servers. We know that $|S_1| + |S_2| \leq 2\alpha - 1$. Obviously, with probability of at least $(1-p)^{|S_1|} \cdot p^{|S_2|} \geq (p - p^2)^{2\alpha-1}$, no quorum from Q can be acquired. As a result, $\lim_{n\to\infty} Avail(\mathcal{Q}) \leq 1 - (p - p^2)^{2\alpha-1} \not\to 1.0$. □

On the other hand, interestingly, we will show that even if we insist on optimal availability, there exists an SQS whose probe complexity is smaller than $2\alpha/(1-p)$.

6.1 Probe complexity lower bounds for SQS with optimal availability

The main challenge in deriving the lower bounds comes from the fact that the probe algorithm can be adaptive based on the results of previous probes. We first discuss how to obtain a lower bound for PC_e^*. In the same client-server context for SQS, we define a *ServerProbe* problem. This problem has its own probe complexity (or simply complexity). Later we will show that the expected probe complexity of an SQS with optimal availability is at least as large as the complexity of the ServerProbe problem. On the other hand, in this ServerProbe problem, all servers play identical roles, which makes it easier to compute its complexity.

Definition 26 *ServerProbe Problem*: A client probes the n servers one by one according to a particular strategy, and stops after the ith probe if any of the following conditions are met (let pos be the number of successful probes and neg be the number of failed probes, where $pos + neg = i$):

1. $pos \geq 2\alpha$.
2. $pos \geq n + \alpha - i$.
3. $neg \geq n + 1 - \alpha$.

Under a given configuration, the number of probes before stopping is called the *total probes*. The expected total probes under all configurations is called the *complexity* of the ServerProbe problem.

It is important to see that even though the definition of the problem has the concept of a "strategy", the complexity of ServerProbe is not affected by the strategy. This is true because all servers are identical and none of the termination conditions distinguish between different servers.

Following we compute the complexity of the ServerProbe problem. Let $f(i) = \text{Prob}[\text{total probes} \leq i]$ and

$a(x, y) = \binom{x}{y} p^{x-y} (1-p)^y$. Suppose $n \geq 3\alpha - 1$ and we consider the following cases:

1. $0 \leq i \leq 2\alpha - 1$. None of the three conditions can possibly be met. Thus $f(i) = 0$.
2. $2\alpha \leq i \leq n - \alpha$. It is impossible to meet the last condition of $neg \geq n - \alpha + 1$. The first two conditions can be merged into a single condition of $pos \geq 2\alpha$. Thus we have $f(i) = \Sigma_{j=2\alpha}^{i} a(i, j)$.
3. $n - \alpha + 1 \leq i \leq n$. The three conditions can be merged into the following two: $pos \geq n + \alpha - i$ or $pos \leq i + \alpha - (n + 1)$. Thus we have $f(i) = \Sigma_{j=0}^{i+\alpha-(n+1)} a(i, j) + \Sigma_{j=n+\alpha-i}^{i} a(i, j)$.

The complexity of ServerProbe is then $g(n) = \sum_{i=1}^{n} i \times (f(i) - f(i-1))$. Based on the original definition of the ServerProbe problem (Definition 26), even if we only use the first termination condition of $pos \geq 2\alpha$ and even if $n \to \infty$, the complexity is always upper bounded by $2\alpha/(1-p)$ (i.e., expected number of trials before 2α successes with the success probability being $1 - p$). Thus, we always have $g(n) < 2\alpha/(1-p)$.

Lemma 27 *Suppose $n \geq 3\alpha - 1$. Let $\mathcal{Q}$ be any SQS where $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$. Let ψ be $\mathcal{Q}$'s probe strategy. Consider any configuration C and ψ's corresponding branch of $path(\psi, C)$:*

- *If the leaf of $path(\psi, C)$ claims that no quorum can possibly be acquired under C, then $path(\psi, C)$ contains at least $n + 1 - \alpha$ failed probes.*
- *If the leaf of $path(\psi, C)$ claims that $Q \in \mathcal{Q}$ has been acquired, then $path(\psi, C)$ contains at least 2α or $n + \alpha - depth(\psi, C)$ successful probes.*

Proof

- Proof by contradiction. Since $\mathcal{Q}$ has optimal availability, from Theorem 20 we know that $\mathcal{C}_\alpha \subseteq \mathcal{Q}$. If $path(\psi, C)$ contains $n - \alpha$ or fewer failed probes, then it is still possible that some quorum in $\mathcal{C}_\alpha$ is available. Thus the leaf cannot claim that no quorum can be acquired. Contradiction.
- If the leaf claims that $Q \in \mathcal{Q}$ has been acquired, from Theorem 20, we know that $|Q^+| \geq \alpha$. If $|Q^+| \geq 2\alpha$, then clearly $path(\psi, C)$ has at least 2α successful probes. If $\alpha \leq |Q^+| \leq 2\alpha - 1$, from Theorem 20, we know that $depth(\psi, C) \geq |Q| \geq n + \alpha - |Q^+|$. □

Lemma 28 *Suppose $n \geq 3\alpha - 1$. For any SQS $\mathcal{Q}$ where $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$, we have $PC_e^*(\mathcal{Q}) \geq g(n)$.*

Proof For any probe strategy ψ for $\mathcal{Q}$, we consider a ServerProbe problem using the same strategy. Under any configuration C where $depth(\psi, C) = i$, the total probes of the ServerProbe problem must be equal to or smaller than i (Lemma 27). This means that $PC_e^*(\mathcal{Q})$ is at least the complexity of the ServerProbe problem, which is $g(n)$. □

We now move on to worst-case probe complexity. For PC_w, it is easy to show that the lower bound is n. This is true because from Theorem 20, $\mathcal{C}_\alpha \subseteq \mathcal{Q}$ and in the worst case, n probes are already necessary to determine whether some quorum in $\mathcal{C}_\alpha$ is alive:

Lemma 29 *For any SQS $\mathcal{Q}$ where $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$, $PC_w(\mathcal{Q}) = n$.*

Proof Construct an adversary which crashes $n - \alpha$ servers among the first $n - 1$ servers probed by the client. We show that the client cannot stop within or immediately after these $n - 1$ probes. Proof by contradiction. Suppose the client returns and claims that a quorum has been acquired. Since the client does not know the state of the last server, it must be true that $\mathcal{Q}$ contains a quorum Q accepted by some $C \in \mathcal{C}_{\alpha-1}$. However, this is impossible because otherwise $\mathcal{Q}$ will not have optimal availability (Lemma 15). On the other hand, if the client stops and claims that no quorum can be acquired, then consider the configuration composed of the state of the first $n - 1$ servers as seen by the client, and also an available last server. Clearly, this configuration belongs to $\mathcal{C}_\alpha$, and thus also belongs to $\mathcal{OPT}_a$ and $As(\mathcal{Q})$ (since $\mathcal{Q}$ has optimal availability). As a result, the configuration would accept a quorum in $\mathcal{Q}$. Contradiction. □

To derive a lower bound on PC_w^*, as in [6], we use Yao's theorem [16]. The theorem says that the expected time of a randomized algorithm A_1 with any input distribution D_1 is always bounded from below by the expected time of the best deterministic algorithm A_2 on inputs coming from any distribution D_2. Thus all we need to do is to construct a difficult distribution D_2 such that any A_2 will perform poorly on average. Following we first cite a lemma from [6] and then prove our lower bound.

Lemma 30 [6] *Consider an urn containing n elements of which w are white and b are black, and suppose elements are taken out one by one without replacement. Then the expected number of trials until obtaining the ith white element is $\frac{i(n+1)}{w+1}$.*

Lemma 31 *Suppose $n \geq 3\alpha - 1$. For any SQS $\mathcal{Q}$ where $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$, $PC_w^*(\mathcal{Q}) \geq \frac{(n-\alpha+1)(n+1)}{n-\alpha+2} = \Omega(n)$.*

Proof Consider the distribution of all configurations in $\mathcal{C}_{\alpha-1}$, where each element is chosen with a probability of $1/|\mathcal{C}_{\alpha-1}|$. Lemma 15 tells us that no quorum in $\mathcal{Q}$ can be accepted under any configuration in $\mathcal{C}_{\alpha-1}$. By Lemma 27, we know that the probe algorithm needs to observe at least $n + 1 - \alpha$ failed probes. After each probe, the remaining servers are totally symmetric in the sense that their probability of being available is equal. Thus it does not matter which server is probed first. The problem now is exactly as in Lemma 30 and the expected number of probes needed is $\frac{(n-\alpha+1)(n+1)}{n-\alpha+2}$. □

Theorem 32 *Suppose $n \geq 3\alpha - 1$. For any SQS $\mathcal{Q}$, where $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$, we have:*

$$PC_w(\mathcal{Q}) = n$$
$$PC_e^*(\mathcal{Q}) = PC_e(\mathcal{Q}) \geq g(n)$$
$$PC_w^*(\mathcal{Q}) = \Omega(n)$$

Proof From Lemma 28, Lemma 29 and Lemma 31. □

6.2 Probe complexity upper bounds for SQS with optimal availability

We now try to construct an optimal-availability SQS that can reach the lower bound on PC_e^*. (The lower bounds on PC_w and PC_w^* are trivially met.) The only way we can match this lower bound is to stop immediately after the conditions in Lemma 27 are satisfied. The only way we can match this lower bound is to stop immediately after satisfying any of the earlier three conditions: (i) 2α successful probes, (ii) $n + \alpha - i$ successful probes, and iii) $n + 1 - \alpha$ failed probes. Just to meet these requirements, some of the quorums are already determined. It is then important to ensure that these quorums can actually constitute an SQS, and further, an SQS with optimal availability.

Our SQS here can be best explained intuitively if we describe it together with a probe strategy. The basic idea is simply to arrange the servers in a fixed order. Each client probes the servers one by one according to that order. The client stops as long as it collects 2α positive replies (quorum acquired). If the client has probed all servers, then the client consider a quorum being acquired as long as it has at least α positive replies. Finally, if the client collects fewer than α positive replies after probing all servers, then it fails to acquire a quorum. Since ultimately, a quorum is always acquired as long as any α servers are available, the SQS has optimal availability. To reach the exact lower bound on PC_e^*, we only need one small optimization. When the client has probed *almost* all n servers, it may stop with somewhat fewer than 2α positive replies (but still more than α). Doing so will ensure that the probing process stops under the exactly same conditions as the ServerProbe problem (Definition 26), whose complexity is a lower bound for PC_e^*.

Following is a standard, combinatorial definition of our SQS (Fig. 4):

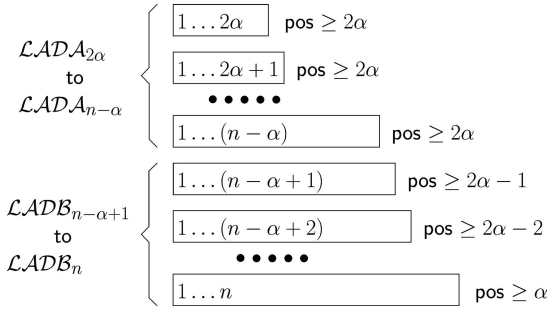

Fig. 4 Construction of $\mathcal{OPT}_a$

$$\mathcal{LAD}_i = \{S \mid |S| = i \text{ and } \forall j \text{ such that } 1 \leq j \leq i, \text{ either } j \text{ or } -j \text{ (but not both) is in } S\}$$
$$\mathcal{LADA}_i = \{S \mid S \in \mathcal{LAD}_i \text{ and } |S^+| \geq 2\alpha\} \text{ for } 2\alpha \leq i \leq n - \alpha$$
$$\mathcal{LADB}_i = \{S \mid S \in \mathcal{LAD}_i \text{ and } |S^+| \geq n + \alpha - i\} \text{ for } n - \alpha + 1 \leq i \leq n$$
$$\mathcal{OPT}_d = \left(\bigcup_{i=2\alpha}^{n-\alpha} \mathcal{LADA}_i\right) \bigcup \left(\bigcup_{i=n-\alpha+1}^{n} \mathcal{LADB}_i\right)$$

It is easy to show that $\mathcal{OPT}_d$ is an SQS with optimal availability:

Lemma 33 *$\forall S \in \mathcal{LAD}_i$ and $\forall T \in \mathcal{LAD}_j$, $i \leq j$:*

1. *If $|S^+| + |T^+| - (j - i) \geq 2\alpha$, then either $S^+ \cap T^+ \neq \emptyset$ or $|S \cap Dual(T)| \geq 2\alpha$.*
2. *If $|S^+| \geq 2\alpha$, then either $S^+ \cap T^+ \neq \emptyset$ or $|S \cap Dual(T)| \geq 2\alpha$.*

Proof 1. If $S^+ \cap T^+ \neq \emptyset$, then we are done. Otherwise let $T' = \{x | -i \leq x \leq i \text{ and } x \in T\}$. Since $|T| - |T'| = j - i$, we know that $|T'^+| \geq |T^+| - (j - i)$. Given that $S^+ \cap T'^+ = \emptyset$, it is obvious that $|S \cap \text{Dual}(T)| = |S \cap Dual(T')| = |S^+| + |T'^+| \geq |S^+| + |T^+| - (j - i) \geq 2\alpha$.

2. If $S^+ \cap T^+ \neq \emptyset$, then we are done. Otherwise we have $|S \cap Dual(T)| \geq |S^+| \geq 2\alpha$. □

Theorem 34 *$\mathcal{OPT}_d$ is an SQS and $Avail(\mathcal{OPT}_d) = Avail(\mathcal{OPT}_a)$.*

Proof First, it is clear that $\mathcal{OPT}_d$ is a signed set system. For any two sets S and T in $\mathcal{OPT}_d$, consider the following three cases:

1. $S \in \mathcal{LADA}_i$ and $T \in \mathcal{LADA}_j$ for some i and j, $2\alpha \leq i \leq j \leq n - \alpha$. Because $|S^+| \geq 2\alpha$, we know from Lemma 33 that either $S^+ \cap T^+ \neq \emptyset$ or $|S \cap Dual(T)| \geq 2\alpha$.
2. $S \in \mathcal{LADA}_i$ and $T \in \mathcal{LADB}_j$ for some i and j, $2\alpha \leq i \leq n - \alpha$ and $n - \alpha + 1 \leq j \leq n$. Because $|S^+| \geq 2\alpha$, we know from Lemma 33 that either $S^+ \cap T^+ \neq \emptyset$ or $|S \cap Dual(T)| \geq 2\alpha$.
3. $S \in \mathcal{LADB}_i$ and $T \in \mathcal{LADB}_j$ for some i and j, $n - \alpha + 1 \leq i \leq j \leq n$. We have $|S^+| + |T^+| - (j - i) \geq (n + \alpha - i) + (n + \alpha - j) - (j - i) = 2n + 2\alpha - 2j \geq 2n + 2\alpha - 2n = 2\alpha$. We now know from Lemma 33 that either $S^+ \cap T^+ \neq \emptyset$ or $|S \cap Dual(T)| \geq 2\alpha$.

Thus, $\mathcal{OPT}_d$ is an SQS. Finally, it is obvious that $\mathcal{OPT}_a = \mathcal{LADB}_n \subseteq As(\mathcal{OPT}_d)$, and from Corollary 17, we have $Avail(\mathcal{OPT}_d) = Avail(\mathcal{OPT}_a)$. □

Our probe strategy for $\mathcal{OPT}_d$ simply probes the servers one by one according to increasing indexes. It is easy to see that the achieved expected probe complexity is exactly $g(n)$:

Theorem 35 *If $n \geq 3\alpha - 1$, then:*

$$Avail(\mathcal{OPT}_d) = Avail(\mathcal{OPT}_a)$$
$$PC_e^*(\mathcal{OPT}_d) \leq g(n) \; < \; 2\alpha/(1 - p)$$

6.3 Notes on novelty and practicality of $\mathcal{OPT}_d$

Our $\mathcal{OPT}_d$ construction is extremely simple with its sequential probe strategy. Sequential probing, by itself, is not a novel idea at all. However, $\mathcal{OPT}_d$ shows that sequential probing, coupled with some small modifications and some extra requirements, can provide very different guarantees (in terms of availability and probe complexity) from those we achieve in traditional quorum systems. To emphasize, these simple modifications are:

- In traditional cases, it may not be required that *all* clients probe the servers in the same order. In $\mathcal{OPT}_d$, it is necessary for all clients to use the same order. Even though in traditional systems, people may still implement the protocol by having all clients use the same order, it is not an explicit requirement for correctness. In fact, for load balancing purposes, there may even be an explicit reason for not doing so.
- If we use sequential probing in traditional quorum systems and $n \geq 4\alpha$, then not all possible sets of 2α positive replies can be quorums (otherwise quorum intersection cannot be guaranteed). In comparison, $\mathcal{OPT}_d$ only needs any 2α positive replies.

Because of it simplicity, $\mathcal{OPT}_d$ can be easily implemented in practical systems. $\mathcal{OPT}_d$ has a large load of 1.0, making it more suitable for cases where load is less important. In some cases, load balancing can be achieved *across* multiple quorum systems rather than *within* a single quorum system. Consider an example where we intend to replicate a set of total o objects on n servers. For different objects, we can use different orders for the n servers. For example, for the ith object (assuming $i \leq n$), the order can be $i, i+1, \ldots, n, 1, 2, \ldots i-1$. Here as long as o is a multiple of n, the load will be perfectly balanced. In practical systems it is likely that $o \gg n$, in which case the load is close to perfectly balanced. $\mathcal{OPT}_d$ is thus applicable to these scenarios as well.

7 SQS with optimal load

7.1 Lower bounds

Different from previous load definitions, our more practical definition captures the load induced by wasted probes. Such a definition introduces an extra challenge because we can no longer compute load based on distributions on the quorums. To address such a challenge, since one major goal/benefit of SQS is high availability, we focus on SQS whose availability is greater than 0.5. We first obtain lower bound given that some quorum is acquired. The overall load is then lower bounded within a 1/2 factor. We use the following notations:

$LoadA(\mathcal{Q}) = Load(\mathcal{Q})$ given that a quorum is acquired
$LoadF(\mathcal{Q}) = Load(\mathcal{Q})$ given that no quorum can be acquired

By definition, we have the following relationship:

$$LoadA(\mathcal{Q}) \cdot Avail(\mathcal{Q}) + LoadF(\mathcal{Q}) \cdot (1 - Avail(\mathcal{Q})) = Load(\mathcal{Q})$$

Similarly, we define $PCA_e^*(\mathcal{Q})$ and $PCF_e^*(\mathcal{Q})$ for probe complexity. From the above relationship, we trivially have:

Lemma 36 *When $Avail(\mathcal{Q}) \geq 0.5$, we have $Load(\mathcal{Q}) \geq LoadA(\mathcal{Q})/2$ and $PC_e^*(\mathcal{Q}) \geq PCA_e^*(\mathcal{Q})/2$.*

Lemma 37 *For any SQS $\mathcal{Q}$, suppose the smallest quorum size in $\mathcal{Q}$ is x, then $PCA_e^*(\mathcal{Q}) \geq x$.*

Our lower bound on load is exactly the same as for UQS [11]. The reason is that if we make all negative elements positive, then an SQS becomes a UQS. The load of the SQS cannot be better than that of the corresponding UQS. Thus the result from [11] applies. This also means that if we are only concerned with load, then SQS does not provide any benefits:

Theorem 38 *For any SQS $\mathcal{Q}$, suppose the smallest quorum size in $\mathcal{Q}$ is x, then $LoadA(\mathcal{Q}) \geq max(x/n, 1/x)$.*

Corollary 39 *For any SQS $\mathcal{Q}$ where $Avail(\mathcal{Q}) \geq 0.5$, we have $Load(\mathcal{Q}) \geq 1/(2\sqrt{n})$ and $Load(\mathcal{Q}) \geq 1/(4PC_e^*(\mathcal{Q}))$.*

This shows that SQS has similar tradeoff between load and probe complexity as UQS.

7.2 Composition of UQS and $\mathcal{OPT}_a$

To approach the lower bound on load, we *compose* certain UQS with $\mathcal{OPT}_a$ to obtain new SQS. The nice property of composition is that the resulting SQS has the availability of $\mathcal{OPT}_a$, and the load and probe complexity of the UQS. The client will first try to use the UQS, and if it fails to acquire a quorum, it will try to acquire a quorum in $\mathcal{OPT}_a$. Because the probe complexity of $\mathcal{OPT}_a$ is quite bad, we need to add some cushion between the UQS and $\mathcal{OPT}_a$ to control the probe complexity. The cushion is similar as the $\mathcal{OPT}_d$ construction. Finally, we need to carefully ensure that the composition result is still an SQS.

Definition 40 Consider any UQS $\mathcal{UQ}$ over the universe of $\{1, 2, \ldots, k\}$ ($k \leq n$), where the size of any quorum in $\mathcal{UQ}$ is at least 2α. Define the *composition* of $\mathcal{UQ}$ and $\mathcal{OPT}_a$ (denoted as $\mathcal{UQ} + \mathcal{OPT}_a$) to be the signed set system $\mathcal{Q}$ where:

$$\mathcal{LADC}_i = \{S | S \in \mathcal{LAD}_i \text{ and } |S^+| = k\} \text{ for } k \leq i \leq n$$
$$\mathcal{Q} = \mathcal{UQ} \cup (\cup_{i=k}^{n} \mathcal{LADC}_i) \cup \mathcal{OPT}_a$$

Figure 5 illustrates the composition technique. From now on, when we use the notation $\mathcal{UQ} + \mathcal{OPT}_a$, we imply that $\mathcal{UQ}$ satisfies the conditions in the above definition.

Theorem 41 *For any $\mathcal{UQ}$, $\mathcal{UQ} + \mathcal{OPT}_a$ is an SQS.*

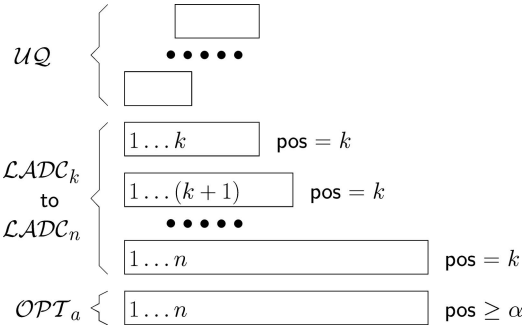

Fig. 5 Composition of $\mathcal{UQ}$ and $\mathcal{OPT}_a$

In order to preserve the load and probe complexity of $\mathcal{UQ}$ in $\mathcal{UQ} + \mathcal{OPT}_a$, our probe algorithm first probes the quorums in $\mathcal{UQ}$, and then moves on to other quorums in $\mathcal{UQ} + \mathcal{OPT}_a$. If the availability of $\mathcal{UQ}$ is reasonably high (with respect to the value of k), the possibility of probing other quorums is low. Thus the resulting probe complexity and load will be dominated by the probe complexity and load of $\mathcal{UQ}$. On the other hand, as long as we try all quorums in $\mathcal{UQ} + \mathcal{OPT}_a$, the availability of $\mathcal{OPT}_a$ is preserved.

Theorem 42 *For any $\mathcal{UQ}$ and $\mathcal{Q} = \mathcal{UQ} + \mathcal{OPT}_a$:*

$$Load(\mathcal{Q}) \leq Load(\mathcal{UQ}) + (1 - Avail(\mathcal{UQ}))$$
$$PC_e^*(\mathcal{Q}) \leq PC_e^*(\mathcal{UQ}) + (1 - Avail(\mathcal{UQ})) \cdot k/(1-p)$$
$$Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$$

Proof By definition, there exists a probe algorithm A for $\mathcal{UQ}$ that can achieve load of $Load(\mathcal{UQ})$ and probe complexity of $PC_e^*(\mathcal{UQ})$. We construct a probe algorithm for $\mathcal{Q}$ as following:

1. Use A on the servers $\{1, 2, \ldots, k\}$. If a quorum in $\mathcal{UQ}$ is acquired, return.
2. Probe the servers one by one from 1 to n. If a quorum in $(\cup_{i=k}^{n} \mathcal{LADC}_k)$ is acquired, return.
3. At this point, all servers have been probed. If some quorum in $\mathcal{OPT}_a$ has been acquired, return. Otherwise claim that no quorum can be acquired.

If the client returns in the first step, then the load on any server must be no larger than $Load(\mathcal{UQ})$. If the client continues to the second step (which happens with probability of $1 - Avail(\mathcal{UQ})$), then the load on any node can be at most 1. Thus we have $Load(\mathcal{Q}) \leq Load(\mathcal{UQ}) + (1 - Avail(\mathcal{UQ}))$. For probe complexity, if the client returns in the first step, the expected probe complexity is $PC_e^*(\mathcal{UQ})$. If the client continues to the second step, then it will stop as soon as having k successful probes (or having probed all servers). A simple calculation can show that the expected number of probes is upper bounded by $k/(1-p)$ for arbitrary n. Finally, it is obvious that $\mathcal{OPT}_a \subseteq As(\mathcal{Q})$ and Corollary 17 tells us that $Avail(\mathcal{Q}) = Avail(\mathcal{OPT}_a)$. □

Corollary 43 *For any $\mathcal{UQ}$ where $1 - Avail(\mathcal{UQ}) \leq 1/k \cdot c$ for some constant c, then:*

$$Load(\mathcal{UQ} + \mathcal{OPT}_a) = O(Load(\mathcal{UQ}))$$
$$PC_e^*(\mathcal{UQ} + \mathcal{OPT}_a) = O(PC_e^*(\mathcal{UQ}))$$
$$Avail(\mathcal{UQ} + \mathcal{OPT}_a) = Avail(\mathcal{OPT}_a)$$

Proof Probe complexity and availability are trivial from Theorem 42. For load, notice that the lower bound on $Load(\mathcal{UQ})$ is $\Omega(1/\sqrt{k})$, which already dominates $1/k \cdot c$. □

Different from our previous SQS constructions, $\mathcal{UQ} + \mathcal{OPT}_a$ uses a potentially randomized and adaptive probe strategy (because of the probe strategy on $\mathcal{UQ}$), and Theorem 9 does not apply. Following we prove that the probability of non-intersection is still properly bounded under such probe strategy:

Theorem 44 *For any $\mathcal{UQ}$ and $\mathcal{Q} = \mathcal{UQ} + \mathcal{OPT}_a$, consider two clients using our previous probe strategy on $\mathcal{Q}$, and let "non-intersection" denote the event that both clients acquire some quorum, but they do not intersect. Then* Prob[*non-intersection*] $\leq 2\varepsilon^{2\alpha}$.

Proof Let Q_1 be the random variable denoting the quorum acquired by the first client and S_1 be the probed servers. Similarly define Q_2 and S_2. Let $\mathcal{SQ} = \mathcal{Q} - \mathcal{UQ}$. We trivially have Prob[non-intersection and $Q_1 \in \mathcal{UQ}$and $Q_2 \in \mathcal{UQ}$] = 0. Now consider the case where $Q_1 \in \mathcal{UQ}$ and $Q_2 \in \mathcal{SQ}$. Because the second client must have probed all of the first k servers, it must have probed all servers in Q_1. In order for the two clients not to intersect, there must be mismatches on all servers in Q_1. Since $|Q_1| \geq 2\alpha$, we have Prob[non-intersection $|(Q_1 \in \mathcal{UQ}$ and $Q_2 \in \mathcal{SQ})] \leq \varepsilon^{2\alpha}$. Similarly, we can prove that Prob[non-intersection $|$ $(Q_1 \in \mathcal{SQ}$ and $Q_2 \in \mathcal{UQ})] \leq \varepsilon^{2\alpha}$.

For the case where both Q_1 and Q_2 belong to $\mathcal{SQ}$, notice that $\mathcal{SQ}$ is an SQS itself. We construct a probe strategy for $\mathcal{SQ}$ using the second and the third step of the probe strategy for $\mathcal{Q}$. Clearly, this new probe strategy is deterministic and non-adaptive. From Theorem 9, we know that the probability (denoted as δ) that two clients of $\mathcal{SQ}$ acquire quorums in $\mathcal{SQ}$ but do not intersect is upper bounded by $\varepsilon^{2\alpha}$. Define the *global state* to be the vector describing the exact state (i.e., $(+,+)$, $(+,-)$, $(-,+)$ or $(-,-)$) of all servers when probed by two clients.[4] Consider any global state where i) two clients of $\mathcal{Q}$ acquire quorums Q_1 and Q_2, and ii) $Q_1 \in \mathcal{SQ}$ and $Q_2 \in \mathcal{SQ}$, and iii) the two clients do not intersect. Then under the same global state, the two clients of $\mathcal{SQ}$ will also acquire Q_1 and Q_2, and they will not intersect either. So we have Prob[non-intersection and $Q_1 \in \mathcal{SQ}$ and $Q_2 \in \mathcal{SQ}$)] $\leq \delta$. Finally, Fig. 6 puts the above four cases together and proves that Prob[non-intersection] $\leq 2\varepsilon^{2\alpha}$. □

[4] We cannot use the notion of configuration here because configuration only describes the server states as observed by a single client.

$$
\begin{aligned}
& Prob[\text{non-intersection}] \\
= \; & Prob[\text{non-intersection and } Q_1 \in \mathcal{UQ} \text{ and } Q_2 \in \mathcal{UQ}] + Prob[\text{non-intersection and } Q_1 \in \mathcal{SQ} \text{ and } Q_2 \in \mathcal{UQ}] + \\
& Prob[\text{non-intersection and } Q_1 \in \mathcal{UQ} \text{ and } Q_2 \in \mathcal{SQ}] + Prob[\text{non-intersection and } Q_1 \in \mathcal{SQ} \text{ and } Q_2 \in \mathcal{SQ}] \\
\leq \; & Prob[\text{non-intersection} \mid (Q_1 \in \mathcal{UQ} \text{ and } Q_2 \in \mathcal{SQ})] \cdot Prob[Q_1 \in \mathcal{UQ} \text{ and } Q_2 \in \mathcal{SQ}] + \\
& Prob[\text{non-intersection} \mid (Q_1 \in \mathcal{SQ} \text{ and } Q_2 \in \mathcal{UQ})] \cdot Prob[Q_1 \in \mathcal{SQ} \text{ and } Q_2 \in \mathcal{UQ}] + \delta \\
= \; & \varepsilon^{2\alpha} \cdot (Prob[Q_1 \in \mathcal{UQ} \text{ and } Q_2 \in \mathcal{SQ}] + Prob[Q_1 \in \mathcal{SQ} \text{ and } Q_2 \in \mathcal{UQ}]) + \delta \\
\leq \; & 2\varepsilon^{2\alpha}
\end{aligned}
$$

Fig. 6 Putting the four cases of non-intersection together

7.3 Composition with the paths UQS

Next we compose the Paths [10, 11] UQS with $\mathcal{OPT}_a$. We first provide a brief, informal review of the Paths UQS. Consider an $l \times (l+1)$ (i.e., l rows and $l+1$ columns) grid G. Define G^* to be the grid obtained by rotating G clockwise by 90 degrees. In order words, G^* is an $(l+1) \times l$ grid. Lay G^* on top of G so that each edge in G^* (except for the edges at the boundary) crosses over an edge in G. An edge in G^*, together with the edge it crosses in G, is called an *edge pair*. Each edge pair corresponds to a server in the universe. A quorum is the union of (servers identified with) the edges of a left-right path in G and the edges of a top-bottom path in G^*. The intersection property is guaranteed since every left-right path in G crosses every top-bottom path in G^*.

Following are the results from [10, 11] regarding the Paths UQS:

Theorem 45 [10, 11] *Let $\mathcal{PH}(l)$ denote the Paths quorum system with $k = 2l^2 + 2l + 1$ servers. Then:*

$$
\begin{aligned}
Load(\mathcal{PH}(l)) &= O(1/l) \\
PC_e^*(\mathcal{PH}(l)) &= O(l) \\
1\text{-}Avail(\mathcal{PH}(l)) &= O(e^{-l})
\end{aligned}
$$

It is simple to show that the smallest quorum size in $\mathcal{PH}(l)$ is l.

Corollary 46 *Let $2\alpha \leq l \leq (\sqrt{2n-1} - 1)/2$, then:*

$$
\begin{aligned}
Load(\mathcal{PH}(l) + \mathcal{OPT}_a) &= O(1/l) \\
PC_e^*(\mathcal{PH}(l) + \mathcal{OPT}_a) &= O(l) \\
Avail(\mathcal{PH}(l) + \mathcal{OPT}_a) &= Avail(\mathcal{OPT}_a)
\end{aligned}
$$

Setting different values for l yields SQS constructions that reach the optimal tradeoff between load and probe complexity while preserving optimal availability.

8 Conclusions

Motivated by the need for highly available and low probe complexity quorum systems in the Internet, this paper proposes *signed quorum systems* (SQS). SQS provides probabilistic intersection guarantee and utilizes the *independent mismatch* property in today's Internet. We show that our optimal SQS construction $\mathcal{OPT}_d$ is available as long as any α servers are available, and simultaneously has a probe complexity of at most $2\alpha/(1-p)$. These properties qualitatively improve upon traditional quorum systems, where even the optimal construction requires $(n+1)/2$ available servers. Our composition technique further shows that SQS can decouple availability from load and probe complexity, and optimal availability can now be achieved under most load and probe complexity values.

Acknowledgements The dropping of the "deterministic" requirement in Theorem 9 resulted from discussions with Praveen Yalagandula during his 2004 summer internship with me at Intel Research Pittsburgh. I would like to thank Phillip B. Gibbons, Dahlia Malkhi, David Peleg, Adrian Perrig, Dazhi Wang, Avishai Wool, and Peng Yin for helpful discussion on various issues related to this paper. I also thank the anonymous reviewers for their detailed feedbacks, which significantly improved this paper.

References

1. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay networks. In: Proceedings of the 18th Symposium on Operating Systems Principles, (October 2001)
2. Barbara, D., Garcia-Molina, H.: The reliability of voting mechanisms. IEEE Transactions on Computers **36**(10), 1197–1208 (1987)
3. Bazzi, R.: Planar quorums. Theoretical Computer Science **243**(1–2), 243–268 (2000)
4. Garcia-Molina, H., Barbara, D.: How to assign votes in a distributed system. Journal of the ACM **32**(4), 841–860 (1985)
5. Gifford, D.K.: Weighted voting for replicated data. In: Proceedings of the 7th Symposium on Operating Systems Principles, (December 1979)
6. Hassin, Y., Peleg, D.: Average probe complexity in quorum systems. In: Proceedings of the ACM Symposium of Principles of Distributed Computing, (August 2001)
7. Holzman, R., Marcus, Y., Peleg, D.: Load balancing in quorum systems. SIAM Journal on Discrete Mathematics **10**(2), 223–245 (1997)
8. Malkhi, D., Reiter, M.: Byzantine quorum systems. In: Proceedings of the 29th ACM Symposium on Theory of Computing, (May 1997)
9. Malkhi, D., Reiter, M., Wool, A., Wright, R.: Probabilistic Quorum Systems. The Information and Computation Journal **170**(2), 184–206 (2001)
10. Naor, M., Wieder, U.: Scalable and dynamic quorum systems. In: Proceedings of the ACM Symposium of Principles of Distributed Computing, (July 2003)
11. Naor, M., Wool, A.: The load, capacity, and availability of quorum systems. SIAM Journal on Computing **27**(2), 423–447 (1998)
12. Peleg, D., Wool, A.: The availability of quorum systems. Information and Computation **123**(2), 210–223 (1995)
13. Peleg, D., Wool, A.: How to be an efficient snoop, or the probe complexity of quorum systems. In: Proceedings of the ACM Symposium of Principles of Distributed Computing, (May 1996)

14. Savage, S., Anderson, T., Aggarwal, A., Becker, D., Cardwell, N., Collins, A., Hoffman, E., Snell, J., Vahdat, A., Voelker, G., Zahorjan, J.: Detour: A Case for Informed Internet Routing and Transport. IEEE Micro **19**(1), 50–59 (1999)
15. Thomas, R.H.: A majority consensus approach to concurrency control for multiple copy databases. ACM Transactions on Database Systems **4**(2), 180–209 (1979)
16. Yao, A.: Probabilistic computations: Towards a unified measure of complexity. In: Proceedings of the 17th Annual Symposium on Foundations of Computer Science, (October–November 1977)
17. Yu, H.: Overcoming the majority barrier in large-scale systems. In: Proceedings of the 17th International Symposium on Distributed Computing, (October 2003)
18. Yu, H., Vahdat, A.: The costs and limits of availability for replicated services. In: Proceedings of the 18th ACM Symposium on Operating Systems Principles, (October 2001)

Haifeng Yu is currently a Researcher at Intel Research Pittsburgh. He is also an Adjunct Assistant Professor at the Department of Computer Science, Carnegie Mellon University. His research interests cover the general area of distributed systems, as well as related fields such as operating systems, database systems, fault-tolerance and large-scale peer-to-peer systems. Haifeng receives his Ph.D. and M.S. from Duke University, and his B.E. from Shanghai Jiaotong University, China. More information about his research is available at http://www.cs.cmu.edu/~yhf.