

Communication-Efficient Probabilistic Quorum Systems for Sensor Networks (Preliminary Abstract)

Gregory Chockler Seth Gilbert* Boaz Patt-Shamir†
chockler@il.ibm.com sethg@mit.edu boaz@eng.tau.ac.il
IBM Research MIT CSAIL Tel Aviv University

Abstract

Communication-efficiency is of key importance when constructing robust services in limited bandwidth environments, such as sensor networks. We focus on communication-efficiency in the context of quorum systems, which are useful primitives for building reliable distributed systems. To this end, we exhibit a new probabilistic quorum construction in which every node transmits at most $O(\log^2 n)$ bits per quorum access, where n is the number of nodes in the system. Our implementation, in addition to being communication efficient, is also robust in the face of communication failures. In particular, it guarantees consistency (with high probability) in the face of network partitions. To the best of our knowledge, no existing probabilistic quorum systems achieve polylogarithmic communication complexity and are resilient to network partitions.

1. Introduction

Quorum systems (see e.g., [8, 17, 7, 10]) have long been used for improving robustness, efficiency and availability of distributed systems. Among their applications are data replication, mutual exclusion, data dissemination, and many others (see e.g., [1, 3, 5, 4, 2, 11, 9, 7, 12, 14]). In a typical quorum-based distributed system, the *servers* store data on behalf of *clients*, which access the data by interacting with subsets of servers, called *quorums*, such that every pair of quorums intersect.

In this paper, we explore the problem of designing efficient quorum systems for environments with limited computational and networking resources, such as sensor net-

works (sensornets). We consider a typical sensornet setting where small battery-powered devices (sensors) with wireless connectivity are deployed in an environment without pre-existing networking infrastructure. The sensors may act as servers that collect and store data intended to be accessed by more powerful client machines. They can also act as clients that store and access information at other sensors for better availability and/or load balancing as discussed in [16]. Due to the ad hoc deployment, the communication network is formed by the devices themselves so that nearby nodes communicate directly, and the nodes residing outside of each other's communication ranges communicate indirectly through other nodes (see Figure 1.a). In addition, due to collisions and other electromagnetic interference, the network can experience frequent message loss and connectivity changes.

Traditionally, the goodness of quorum systems has been evaluated in terms of their load, availability and probe complexity [15]. One limitation of these metrics is that they are mainly concerned with the overall system performance and tend to disregard the communication and processing overhead incurred by each individual node. Although this overhead is negligible in environments with abundant resources and ubiquitous connectivity (such as the Internet), it is one of the dominant factors affecting sensornet performance. In particular, the energy and bandwidth consumption of each node are both directly proportional to the number of bits it has to transmit for each quorum access. In addition, to cope with intermittent connectivity, the messages in sensornets are typically disseminated by gossip. As a result, most nodes participate in every quorum access. This further limits the applicability of the load and probe complexity metrics, leaving communication complexity as the primary performance measure.

To understand the challenges associated with designing communication-efficient quorum systems in sensornets, let us consider a probabilistic quorum system \mathcal{Q} of [13]. This quorum system employs quorums of size \sqrt{n} where n is the number of nodes in the system, and is optimal in terms of

*This work is supported by MURI-AFOSR SA2796PO 1-0000243658, USAF-AFRL #FA9550-04-1-0121, NSF Grant CCR-0121277, NSF-Texas Engineering Experiment Station Grant 64961-CS, and DARPA F33615-01-C-1896.

†Supported in part by a grant from Israel Ministry of Science and Technology.



(a) Quorum access in an arbitrary network, such as a sensor-net.

(b) Quorum access in the presence of a networking infrastructure (e.g., Internet).

Figure 1. Accessing quorums in different network topologies.

both load and availability. First, it is easy to see that \mathcal{Q} (and in fact any quorum system) has an optimal communication cost—with respect to the servers—in a network that supports direct node-to-node connectivity, such as the one in Figure 1.b. Indeed, assuming that the node identifiers and the request payload are both of size $O(\log n)$, the number of bits transmitted by each server in each quorum access can be as low as $O(\log n)$. However, if $Q \in \mathcal{Q}$ is accessed over a sensor-net (see Figure 1.a), then some servers (e.g., S_3 in Figure 1.a) can transmit as many as $O(\sqrt{n} \log n)$ bits per quorum access.

One possible way around this problem is to maintain a spanning tree over the nodes in the system, and to use broadcast/convergecast for propagating requests and collecting responses. However, due to intermittent connectivity, maintaining a spanning tree in a sensor-net is difficult, if at all feasible. In fact, intermittent connectivity is one of the defining characteristics of sensor networks. Connectivity changes continuously, as nodes turn themselves off to hibernate and save power, or as nodes move from one location to another. To build a spanning tree would require continually monitoring the topology and updating the tree, a potentially expensive use of energy.

It is therefore, a natural question to ask whether the per-node communication overhead can be reduced *regardless* of the topology of the network, the method of quorum access, and intermittent connectivity. Specifically, is logarithmic or polylogarithmic bit complexity achievable?

In this paper, we answer this question in the affirmative. We present a new, probabilistic quorum system that achieves polylogarithmic— $O(\log^2 n)$ —communication complexity at the cost of a non-zero—but polynomially small—probability of accessing non-intersecting quorums. The main idea of our construction is to use probabilistic sampling to eliminate the need for accessing too many individual nodes in the system. It works roughly as follows: For each quorum access, a client first chooses a sample of node identifiers of size $O(\log n)$, uniformly at random, and then initiates gossip to disseminate

this sample—along with the payload for the access—to the nodes in the system. Upon receiving a request, each individual sensor first updates its state if necessary, and then checks whether its identifier is included in the sample. If so, it initiates a gossip to return a response to the client. The client continues to gossip its request until the responses from a *subset* of the original sample have been received. Once this happens, the quorum access is regarded as completed.

Note that in our protocol, the quorum access can terminate even if the client has not received responses from *every* node in the sample. Thus, a quorum is effectively any subset—of a sufficient size—of a random subset of size $O(\log n)$. The actual size of this subset is determined by the desired failure threshold, which can be polynomially close to $n/2$.

It might be possible to tolerate more failures (e.g., as many as $n - \sqrt{n}$ in [13]) by repeating the sampling step if too many of the nodes in the original sample appear to have failed. There are two problems with this approach though: First, since reliable failure detection cannot be guaranteed, the client might end up selecting too many nodes, potentially resulting in too many messages being sent. Second, when intermittent connectivity causes the network to become partitioned for sufficiently long, the samples being drawn by the client become increasingly biased towards the connected nodes (due to the reduced sample space) eventually leading to violation of the probabilistic intersection guarantee. Our protocol avoids these problems by employing quorums which are subsets of the initial sample so that failure resilience is achieved without repeating the sampling step.

Of course, if too many nodes in the sample are indeed inaccessible, it might take an arbitrarily long time—at least until the connectivity is restored—for the quorum access to complete. The continuous gossip ensures that this will happen with high probability once the network connectivity is restored. An obvious problem with continuous gossip is that it can result in too many messages being transmitted when the network is unstable. In practice however, the per-

formance can easily be tuned up (e.g., by adjusting timeouts between consecutive gossip rounds) to ensure quick termination under normal network conditions.

Lastly, another property of our construction which is useful for applications invoking many rounds of quorum access throughout their execution is that the quorum intersection is guaranteed with *high probability* (as opposed to *constant probability*). To the best of our knowledge, none of the existing probabilistic quorum systems are simultaneously communication-efficient, resilient to network partitions and guarantee quorum intersection with high probability.

The rest of this paper is organized as follows: In Section 2, we briefly outline our system model. In Section 3, we describe our quorum system and sketch its correctness argument. We conclude and discuss the future directions in Section 4.

2. System Model

We consider an ad hoc sensor network of an arbitrary topology consisting of n (sensor) nodes, $S = \{s_1 \dots s_n\}$. Some fraction p of the nodes may fail, while $(1-p)n$ nodes operate correctly. (We do not make any probabilistic assumption about the failures.) For the sake of this paper, we assume that $0 \leq p < 1/4$. (In fact, p can be polynomially close to $1/2$, at the cost of larger hidden constants.)

We assume that the network can experience frequent connectivity changes, possibly resulting in network partitions, and that the individual nodes can crash and recover. For lack of space and to simplify the presentation, we also assume that the nodes have unique identifiers and that the set of participants is fixed and known to the clients. Possible ways of removing these assumptions are discussed in Section 4.

Nodes communicate by broadcasting messages on the wireless medium. Normally, each message broadcast by a node p would be received by every node within p 's communication range. Messages can be lost due to collisions and other disruptions of the wireless medium. We do not assume that nodes have any way of detecting lost messages.

3. Communication-Efficient Probabilistic Quorums

In this section, we describe the new probabilistic quorum system. Throughout the description of the algorithm, fix a constant c , for some $c \geq 1$, where $1 - n^{-\Theta(c)}$ is the desired probability of quorum intersection. Fix $\tau = (1-p)/5$, where p is the fraction of nodes in the network that may fail.

A quorum system supports two types of operations: update and query. An update(*value*) operation propagates

```

Sensors
1  repeat forever at sensor  $s$ :
2    do  $\langle op, sample, val \rangle \leftarrow$  Gossip-receive()
3    if  $s \in sample$ 
4      then if  $op = \text{update}$ 
5        then  $data \leftarrow data \cup val$ 
6          Gossip(response,  $s, \perp$ )
7      else if  $op = \text{query}$ 
8        then Gossip(response,  $s, data$ )

```

Figure 2. Pseudocode for sensors. Each sensor gossips information, updating its *data* based on update requests and returning its *data* based on query requests.

data to a *write quorum* of size at least $(1-p-2\tau)n$, with high probability. A query() operation retrieves data from a *read quorum* of size at least $\Theta((1-p-\tau) \log n)$. Each read quorum intersects each write quorum with high probability, i.e., with probability n^{-c} for some $c > 1$.

We assume that the *value* being disseminated to the quorum system is itself of size $O(\log n)$; if the data being disseminated is large, it is obviously impossible to replicate it while transmitting only a small number of bits¹. (Typically, packet size is constant, even as the scale of the network grows.)

The main idea of our protocol is to use sampling to minimize the cost of quorum accesses. We will uniformly choose sets of size $r = \Theta(c \log n)$. (The constant hidden in the Θ notation does depend on p .) The initiator then waits for responses from members of the sample. It cannot, of course, wait for responses from every one of the r nodes in the sample, since some may have failed. Since the sample is chosen uniformly at random, however, we can be (relatively) certain that at least $(1-p-\tau)r$ nodes in the r -sized sample have not failed. On receiving a response of size $(1-p-\tau)r$, the initiator can determine that the value has been sufficiently propagated, in the case of an update, or that sufficient responses have been collected, in the case of a query. Even though only a fraction of the sampled nodes respond, the sample is large enough to ensure that even adversarially-chosen message delays cannot trick the initiator.

The pseudocode for the sensors (i.e., servers) appears in Figure 2, and the quorum access routines appear in Figure 3. An update(*value*) begins by choosing a random sample of

¹For the case of large data, it has been suggested previously that it is possible to use quorum systems to maintain small "metadata" which can then be used to ensure data consistency [6].

<pre> Update(<i>value</i>) 1 <i>sample</i> ← Random(<i>S</i>, <i>r</i>) 2 <i>responses</i> ← ∅ 3 while <i>responses</i> < (1 - <i>p</i> - τ)<i>r</i> 4 do <i>responses</i> ← Gossip(update, <i>sample</i>, <i>value</i>) 5 return </pre>	<pre> Query() 1 <i>sample</i> ← Random(<i>S</i>, <i>r</i>) 2 <i>responses</i> ← ∅ 3 while <i>responses</i> < (1 - <i>p</i> - τ)<i>r</i> 4 do <i>responses</i> ← Gossip(<i>query</i>, <i>sample</i>, ⊥) 5 return <i>responses</i> </pre>
---	--

Figure 3. Pseudocode for quorum access routines. Random(S, r) chooses r elements uniformly at random from S . Gossip(y) uses gossip to disseminate the value y to quorums.

size $r = \Theta(c \log n)$. The value, along with the identity of the random sample, is propagated throughout the network. Each node gossips the value to its neighbors. When a sensor discovers that it is part of the sample, it updates its *data*, and sends a response². Finally, when the initiator of the quorum access has received a response from $(1-p-\tau)r$ nodes in the sample, it can conclude that the quorum access is complete. At this point, we can show the following lemma:

Lemma 1. *When a write access is complete, at least $(1 - p - 2\tau)n$ nodes in the network have received the value, with high probability.*

Proof (sketch). Assume that no more than $(1 - p - 2\tau)n$ nodes have received the value. Then, since the sample is chosen uniformly at random, the expected number of nodes in the sample that have received the value is $\leq (1-p-2\tau)r$. Choose $\delta = \tau/(1 - p - 2\tau) = 1/3$. Using a Chernoff bound³, we can conclude that the probability that $(1 - p - \tau)r$ nodes in the sample have received the value—even though no more than $(1 - p - 2\tau)n$ nodes received the value—is no more than $n^{-c\delta^2/4}$. That is, if $(1 - p - \tau)r$ nodes have responded, then with high probability at least $(1 - p - 2\tau)n$ nodes have received the *value*. \square

We refer to the nodes that have received the value by the time the write completes as the *write quorum* for the write access.

A query access simply accesses an r -sized sample of the network. Again, the initiator chooses uniformly at random r nodes from the network, and disseminates the random sample through the network. As in the case of an update access, each of the sampled nodes sends a response back to the initiator, including its *data* and its identifier. When the initia-

tor has received $(1 - p - \tau)r$ responses, it can complete the query access. We refer to the nodes that have responded as the *read quorum* for the read access. We can then conclude the following lemma:

Lemma 2. *Each read quorum intersects each write quorum with high probability.*

Proof (sketch). Consider the case where at least $(1 - p - 2\tau)n$ nodes are in the write quorum; recall from Lemma 1 that this occurs with probability at least $1 - n^{-c/36}$. If for every subset of the read sample of size at least $c(1 - p - \tau) \log n$, one node in the subset intersects with the prior write quorum, then intersection is achieved. We therefore calculate the probability that there exist any $c(1 - p - \tau) \log n$ nodes in the read sample that have not received the value.

The expected number of nodes in the read sample of size $c \log n$ that have not received the value is $c(p + 2\tau) \log n$. Choose $\delta = (1 - 2p - 3\tau)/(p + 2\tau)$. Using a Chernoff bound, we can conclude that the probability that $c(1 - p - \tau) \log n$ nodes have not received the value is no more than $n^{-c\delta^2/4}$. Thus, by a union bound, the total probability of non-intersection is $\leq n^{-c/36} + n^{-c\delta^2/4}$, i.e., polynomially small in n . \square

Notice that neither the update nor the query access requires any sensor to send more than $O(\log^2 n)$ bits of information: the sample size is of size $O(\log n)$ and each identifier requires $O(\log n)$ bits.

A final claim is that each quorum access completes, even if a p fraction of the nodes fail. Since the samples are chosen uniformly at random, we can expect that no more than a p fraction of each sample will have failed, leading to the desired result:

Lemma 3. *If there exists a time after which the network is connected, a read or write access will eventually complete with high probability.*

²In many cases, updating the data may involve applying some “aggregation” function; for example, often a time-stamp is associated with each update, and only the data with the largest time-stamp is recorded. In this way, the size of the data is prevented from growing linearly with the number of updates.

³That is, if μ is the expected sum of independent trials X_i and $\delta < 2e - 1$, then $\Pr(\sum X_i > (1 + \delta)\mu) \leq e^{-\mu\delta^2/4}$.

Proof (sketch). The expected number of nodes in a sample that have not failed is $\leq c(1-p)\log n$. Choose $\delta = (2 - 2p - \tau)/(1-p)$. Using a Chernoff bound (yet again) we can conclude that the probability of more than $c(1-p-\tau)\log n$ nodes in the sample have failed is polynomially small in n . \square

One implication of this lemma is that initiator s_i should not repeat an operation simply because it has not received a response for a period of time. (That is, s_i should not “time-out” operations.) If there is no response, then the most likely cause is unreliable communication, rather than failures. This allows the quorum system to meet our goal of tolerating partitions.

4. Conclusions and Future Work

In order for quorum systems to be a practical solution for sensor networks, it is necessary to reduce the amount of information which must be transmitted. Probabilistic quorum systems provide one such mechanism for accomplishing this in a network with wholly unreliable communication.

There are three major directions for improvement. First, we can reduce even further the number of bits required: instead of $O(\log^2 n)$ bits per quorum access, it seems possible to achieve $O(\log n \log \log n)$ bits per quorum access through better data encodings. Similar techniques may allow us to reduce the dependency on knowing the size of the network n , and knowing unique identifiers.

Second, in this paper we make very weak assumptions about the wireless network, allowing arbitrary message loss. In some wireless networks, most message losses are due to collisions, and by using a backoff protocol it is often possible to reduce the contention on the wireless channels. In such a network, it is possible to analyze the performance of these quorum protocols, when combined with a randomized backoff protocol, resulting in a polylogarithmic number of bits transmitted, regardless of adversarial choices.

Third, in this paper we assume only benign failures. There are interesting possibilities in extending the probabilistic quorum systems in this paper to tolerate byzantine failures. There are two challenges involved: byzantine quorum systems require increased intersection to ensure correctness, and byzantine nodes may create problems by propagating messages incorrectly through the network. When cryptographic techniques are available, these problems can be overcome. It remains an open question whether it is possible with minimal cryptographic tools.

In addition to the theoretical directions mentioned above, we also intend to carry out experimental studies to evaluate the performance of our quorum systems in both simulations and real sensor networks.

References

- [1] D. Agrawal and A. E. Abbadi. Resilient logical structures for efficient management of replicated data. Technical report, University of California Santa Barbara, 1992.
- [2] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. *Journal of the ACM*, 42(1):124–142, 1995.
- [3] M. Bearden and R. P. B. Jr. A fault-tolerant algorithm for decentralized on-line quorum adaptation. In *Proceedings of the 28th International Symposium on Fault-Tolerant Computing Systems*, Munich, Germany, 1998.
- [4] A. El Abbadi, D. Skeen, and F. Cristian. An efficient fault-tolerant protocol for replicated data management. In *Proc. of the 4th Symp. on Principles of Databases*, pages 215–228. ACM Press, 1985.
- [5] A. El Abbadi and S. Toueg. Maintaining availability in partitioned replicated databases. *Transactions on Database Systems*, 14(2):264–290, 1989.
- [6] R. Fan and N. Lynch. Efficient replication of large data objects. In *Proceeding of the 17th International Conference on Distributed Computing*, October 2003.
- [7] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, 1985.
- [8] D. K. Gifford. Weighted voting for replicated data. In *Proceedings of the seventh symposium on operating systems principles*, pages 150–162, 1979.
- [9] S. Gilbert, N. Lynch, and A. Shvartsman. RAMBO II:: Rapidly reconfigurable atomic memory for dynamic networks. In *Proc. of the Intl. Conference on Dependable Systems and Networks*, pages 259–269, June 2003.
- [10] M. Herlihy. A quorum-consensus replication method for abstract data types. *ACM Transactions on Computer Systems*, 4(1):32–53, feb 1986.
- [11] N. Lynch and A. Shvartsman. RAMBO: A reconfigurable atomic memory service for dynamic networks. In *Proc. of the 16th Intl. Symp. on Distributed Computing*, pages 173–190, 2002.
- [12] M. Maekawa. A \sqrt{N} algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3(2):145–159, 1985.
- [13] D. Malkhi, M. Reiter, and R. Wright. Probabilistic quorum systems. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 267–273, 1997.
- [14] M. Naor and U. Wieder. Access control and signatures via quorum secret sharing. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):909–922, 1998.
- [15] D. Peleg and A. Wool. How to be an efficient snoop, or the probe complexity of quorum systems. *SIAM Journal of Discrete Mathematics*, 15(3):416–433, 2002.
- [16] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Gvoindan, L. Yin, and F. Yu. Data-centric storage in sensor networks with GHT, a geographic hash table. In *ACM MONET*, 2003.
- [17] R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *Transactions on Database Systems*, 4(2):180–209, 1979.