The Obscure Nature of Epidemic Quorum Systems

João Barreto Paulo Ferreira
INESC-ID and Technical University of Lisbon, Portugal

Abstract

Epidemic quorum systems enable highly available agreement even when a quorum is not simultaneously connected, and are therefore very interesting for mobile networks. Although recent work has proposed epidemic quorum algorithms, their properties and trade-offs are not well studied. This paper sheds some light on less known aspects of epidemic quorum systems. With simple counter-examples and combinatorial exercises, we contradict common misbeliefs that are often associated with epidemic quorum systems. Our claims advocate the need for a deeper study of these promising systems.

1 Introduction

Quorum systems are a basic tool for reliable agreement in distributed systems [PW95]. Their applicability is wide, ranging from data replication protocols, distributed mutual exclusion, name servers, selective dissemination of data, to distributed access control and signatures [AW96]. Classical quorum systems (CQS) require agreement on a value to be accepted by a *quorum* of live processes, which are typically assumed to be simultaneously connected in the same network partition. This is not adequate in weakly connected networks, e.g. mobile or sensor networks, where connected quorums may be improbable.

Recent work [Kel99, HSAA03] has proposed epidemic quorum systems (EQS) to eliminate such a shortcoming by allowing unconnected quorums. An EQS tries to ensure agreement by running a finite number of *elections*. Intuitively, on each election, each process may vote for one proposed value. By epidemic propagation of votes, eventually each process should be able to determine, from its local state, whether the EQS has agreed on a given value w, or the current election is inconclusive and, hence, a new one starts. EQSs, hence, seem a powerful tool for coordination in mobile computing.

A common fallacy that may cross one's mind is that EQSs behave similarly to CQSs; with the superficial distinction that the former are better adapted to weakly connected environments and the latter to strongly connected ones. In fact, the absence of any deep study, either empirical or formal, of EQSs (to the best of our knowledge) may consistent with such a misbelief: after all, if both approaches were fundamentally similar, then the extensive studies and results that are available on CQSs [JM90, CW96, IK01, PW95, JM05] would also apply to the epidemic variant.

This paper exposes such a fallacy. Our contribution comprises a series of counter-examples and simple exercises that unveil EQS-specific properties and trade-offs which related literature, to the best of our knowledge, does not document. As corollaries:

1. We show that known, fundamental results for CQSs are not valid with EQSs.
2. We question the environments where EQSs are commonly assumed to be more appropriate than classical quorums, and vice-versa. In particular, we argue that EQSs may not always be advantageous over CQSs in mobile networks, while EQSs can outperform CQSs in high-throughput agreement in strongly-connected environments.
3. We raise open questions that need to be answered before EQSs can be clearly understood.

Section 2 starts by describing epidemic quorum algorithms and coteries, clarifying some less obvious, yet fundamental, aspects that distinguish EQSs from their classical counterpart. Departing from such an analysis, Sections 3 to 6 draw and discuss individual, novel statements about the nature of EQSs, which support our position. Section 7 concludes.

2 Epidemic and Classical Quorum Systems

We consider a set of distributed processes in an asynchronous system. Processes may fail permanently. Failures are non-byzantine. Transient network partitions may also occur, for instance due to mobility, restricting communication to processes inside the same partition.

The distributed processes wish to agree on a single value, taken from a set of values, proposed during the agreement process. For an easier comparison between CQSs and EQSs, we use the voting metaphor to describe both systems. A quorum algorithm tries to agree on a single value by having each process vote for a proposed value. Roughly speaking, the algorithm reaches agreement when it is sure that the system has reached a safe configuration of votes. A *coterie* defines the set of such configurations. As we clarify next, the notion of coterie is different from CQSs to EQSs. The corresponding quorum algorithms are different, too.

We begin with the classical case. A classical coterie is a set of sets of processes (quorums), $Q_1, ..Q_m$, such that, for all Q_i and Q_j, $Q_i \cap Q_j \neq \emptyset$ (intersection property).[1] In a CQS, a process proposing a value tries to make a quorum of processes vote for its value. Such a step must complete atomically. A common solution is to employ an atomic commit protocol, such as two-phase commit (2PC), for quorum obtention (e.g. [Gif79]). Taking the example of 2PC, a coordinator (typically the process proposing the value) requests each other accessible process to vote for the value (or pre-commit). While having a vote cast (i.e., pre-committed), a process cannot vote for any other value. Once the coordinator has obtained a quorum of processes that voted for its value, it then requests each of them commit, hence deciding the value. If, otherwise, the coordinator is not able to obtain a quorum of votes, it may, at any time, request the processes voting for its value to abort. Upon abort, a process withdraws its vote and regains the right to vote for any value.

EQSs adopt a radically different strategy to obtain a quorum. Here, processes are no longer allowed to withdraw votes. Such a restriction has important consequences. A coordinator is no longer required. Instead, the proposer process votes for its value and propagates information about such a vote to other processes. Upon receiving vote information about a proposed value, a process that has not yet voted in the current election casts its vote for the value. Therefore, votes flow epidemically, and the election decision is taken in a purely decentralized manner. Each process inspects its local voting knowledge and determines if a vote configuration in the coterie has already been *reached* (see below), in which case a value is decided. It may also occur that a process determines that no configuration in the coterie may ever be reached, even after the missing votes arrive; in such a case, the process regards the current election as *indecisive* and starts a new one.

Epidemic coteries (ECs), in contrast to classical coteries, may not only take the set of voters for a given value (the *quorum*) into account, but also the set of voters for the competing values (the *anti-quorums*[2]). The reason for this is the impossibility of vote withdrawal. Note that, in contrast, the coordinator of a CQS, when inspecting its vote knowledge, cannot reason about the known votes for other values, as those votes may change at any time.

We have proposed a formal definition (and proven it safe), which is out the scope of this paper, in [BF07]. We instead give an intuitive description. Each configuration of votes in an EC includes a quorum and a set of anti-quorums. We say that a given vote configuration in the EC has been *reached* once its quorum has voted for a value; while all the anti-quorums of the vote configuration have each voted for a distinct value.

Intuitively, the condition that determines which vote configurations may co-exist in an EC ensures the following. Consider that, given a set of votes, a vote configuration whose quorum votes for a value w is reached. No arbitrary assignment of the missing votes may reach another vote configuration whose quorum votes for a value other than w. This ensures the safety of deciding w. (Note that other vote configurations may be reached, as long as their quorums vote for w.)

[1] For simplicity, and without loss of generality, we omit the minimality property from coterie definitions.

[2] The expression is due to Holliday et al [HSAA03]

Quorum	Anti-Quorums
p_1, p_2, p_3	
p_1, p_2, p_4	
p_1, p_2, p_5	
p_1, p_3, p_4	
p_1, p_3, p_5	
p_1, p_4, p_5	
p_2, p_3, p_4	
p_2, p_3, p_5	
p_2, p_4, p_5	
p_3, p_4, p_5	

Quorum	Anti-Quorums
p_1, p_2	$\{p_3\}, \{p_4\}$
p_1, p_2	$\{p_3\}, \{p_5\}$
p_1, p_2	$\{p_4\}, \{p_5\}$
p_1, p_3	$\{p_2\}, \{p_4\}$
p_1, p_3	$\{p_2\}, \{p_5\}$
p_1, p_3	$\{p_4\}, \{p_5\}$
p_1, p_4	$\{p_2\}, \{p_3\}$
p_1, p_4	$\{p_2\}, \{p_5\}$
p_1, p_4	$\{p_3\}, \{p_5\}$
p_1, p_5	$\{p_2\}, \{p_3\}$
p_1, p_5	$\{p_2\}, \{p_4\}$
p_1, p_5	$\{p_3\}, \{p_4\}$

Quorum	Anti-Quorums
p_2, p_3	$\{p_1\}, \{p_4, p_5\}$
p_2, p_3	$\{p_1\}, \{p_4\}, \{, p_5\}$
p_2, p_4	$\{p_1\}, \{p_3, p_5\}$
p_2, p_4	$\{p_1\}, \{p_3\}, \{, p_5\}$
p_2, p_5	$\{p_1\}, \{p_3, p_4\}$
p_2, p_5	$\{p_1\}, \{p_3\}, \{, p_4\}$
p_3, p_4	$\{p_1\}, \{p_2\}, \{, p_5\}$
p_3, p_5	$\{p_1\}, \{p_2\}, \{, p_4\}$
p_4, p_5	$\{p_1\}, \{p_2\}, \{, p_3\}$
p_1	$\{p_2\}, \{p_3\}, \{, p_4\}, \{, p_5\}$

Figure 1: The plurality EC, $\mathcal{E}_{Plur}$.

We exemplify with the plurality EC used by the Deno system [Kel99], which we denote $\mathcal{E}_{Plur}$. Essentially, plurality determines that a value w is decided when, for each (possibly unknown) rival value, x, either (a) w is guaranteed to have more votes than those that x may potentially acquire (with the missing votes); or (b) w is guaranteed to have as many votes as those x may potentially acquire *and* a process p that has voted for w has a lower identifier than any of the processes that may potentially vote for x.

For a system of five processes, Figure 1 enumerates (the minimal) $\mathcal{E}_{Plur}$. To illustrate the safety of $\mathcal{E}_{Plur}$, assume that a given process is aware that a value x has votes from p_1 and p_2, value y has votes from p_3 and value z has votes from p_4. Clearly, such votes reach vote configuration $\langle\{p_1, p_2\}|\{p_3\}, \{p_4\}\rangle$, where x is voted by the quorum; hence, the process decides x. If either y or z obtain the missing vote (from p_5) no other configuration where a quorum votes for either y or z will be reached – this condition ensures that any other process will eventually decide x.

The universe of ECs is larger than the classical one. On the one hand, the former includes coteries to which CQSs cannot resort to. On the other hand, it is easy to prove that the intersection property is stronger than the condition for ECs [BF07]. Hence, any classical coterie may also be used by an EQS, as proposed in [HSAA03]. We designate the EC that directly results from a classical coterie as the latter's *epidemic-equivalent*.

As a final remark before the next sections, it may be helpful to look at EQSs from the perspective of CQSs. EQSs may be regarded as an enhanced variant of a CQS using One-Phase Commit. In the absence of faults, a CQS based on 1PC can decide in one round and cannot abort (i.e., withdraw votes), similarly to EQSs. The advantage of EQSs is the ability to exploit the impossibility of vote withdrawal, and hence use a wider universe of coteries, with both quorums and anti-quorums.

3 EQSs aren't always better in mobile networks

CQSs are normally regarded as working in partition-free conditions. However, saying that CQSs may only decide when the quorum is simultaneously connected is not entirely true. The precise requirement, in fact, is that, at each phase of the vote obtention stage (e.g. via 2PC), each single process in the quorum be accessible to the coordinator at some moment. Simultaneous availability of the quorum as a whole is not necessary. Temporary inaccessibility of any of the processes involved may indeed happen, with the sole consequence of delaying agreement.

Having observed that, in fact, both options are valid for transiently-partitioned networks, which one is the best choice? The answer depends on an important trade-off between two distinctive advantages of each approach. On the one hand, EQSs are truly decentralized, while CQSs base agreement on interactions with the coordinator. Hence, should the coordinator become inaccessible from a quorum that has already completed the first 2PC phase, then agreement on any value will *always* halt until the coordinator becomes accessible again. That is not the case with EQSs, where agreement may (although possibly not always) complete in spite of the inaccessibility of any individual process.

On the other hand, multiple proposed values, along with the inaccessibility of one or more processes, may prevent EQSs from deciding in situations where CQSs do decide. This is a direct consequence of the possibility of vote withdrawal in a CQS and of the corresponding impossibility on the epidemic side. We illustrate this with an example. Consider a partition that holds 4 of the 5 processes of the system. Let us compare a CQS and an EQS, both using majority coteries (i.e. a value must obtain 3 votes to be decided). Assume that two values, x and y, have been proposed and each has obtained 2 votes so far. In this situation, the vote that is missing for a majority of either proposal is held by the process outside the partition. In fact, the EQS would wait for such a process to become accessible again, even if that takes a long time or if the process is permanently failed. We call this situation a *tied indecision*.

In the classical approach, however, a given coordinator (e.g. x's proposer) may decide, for instance once a reasonable timeout expires, to withdraw the votes for x. Having withdrawn such votes, the coordinator of y may now obtain the missing vote from the free processes, and decide y.

Summing up, the fully decentralized nature of EQSs removes the need for a distinguished coordinator process (per value proposed) with relatively good connectivity to the remaining processes. However, should processes with such connectivity conditions exist, tied indecisions may cause EQSs to halt (possibly forever) when CQSs will not. In mobile networks where a subset of processes has a sufficiently high probability of having contact with the remaining processes, even if such contacts are interleaved with periods of lack of connectivity, the classical approach may indeed perform better. Both in terms of time to decide and availability. Without a precise analysis of both approaches, the boundary between the scenarios where each approach is more appropriate remains unclear.

4 Coterie inclusion does not necessarily mean lower availability

It is a known result that, given two classical coteries, C_1 and C_2, such that $C_1 \subset C_2$, a CQS using C_1 will have a lower availability than one using C_2 [PW95]. As an example, C_1 might be the classical coterie that includes the quorums with more than 65% of processes, while C_2 is the classical majority coterie. It is intuitive that a system using C_2 will decide more frequently, as not only will it obtain a quorum in the same situations as C_1, but as well in others.

Perhaps surprisingly, the same reasoning may no longer be valid in the context of EQSs. As before, an EC, E_2, that is a superset of another, E_1, will have a superset of opportunities to decide. However, in the epidemic case, a new phenomenon comes into play, which increases availability as ECs become smaller.

We illustrate it with the following example. Consider a system of 10 processes in a single partition, two of which have permanently failed. Further, assume values x and y have been proposed and each one one has obtained four votes. If we consider an EQS using the epidemic-equivalent of the majority coterie, then the system is in a tied indecision. Since the missing votes still allow both candidates to obtain a majority, the system has no choice but to wait (in this case, forever).

Now, consider that the system was, instead, using the epidemic-equivalent of the smaller coterie, C_1 (quorums with more than 65% of processes). In the previous situation, the EQS would then determine that neither x nor y would ever obtain more than 65% votes from the missing processes. Hence, the tied indecision would no longer hold. Instead of halting, the EQS can safely conclude that the current election is indecisive and start a new election. Possibly, a different vote distribution of the new election would then decide one of the values.

The example above exposes an inherent trade-off of EQSs that is not present in CQSs. On the one hand, larger ECs leverage availability by increasing the probability of deciding in a single election. On the other hand, smaller ECs contribute to a higher availability by preventing tied indecisions (thus repeating indecisive elections). To the best of our knowledge, the previous trade-off is neither documented nor studied in literature. Precisely determining the total availability that results from both vectors is a non trivial question that remains open.

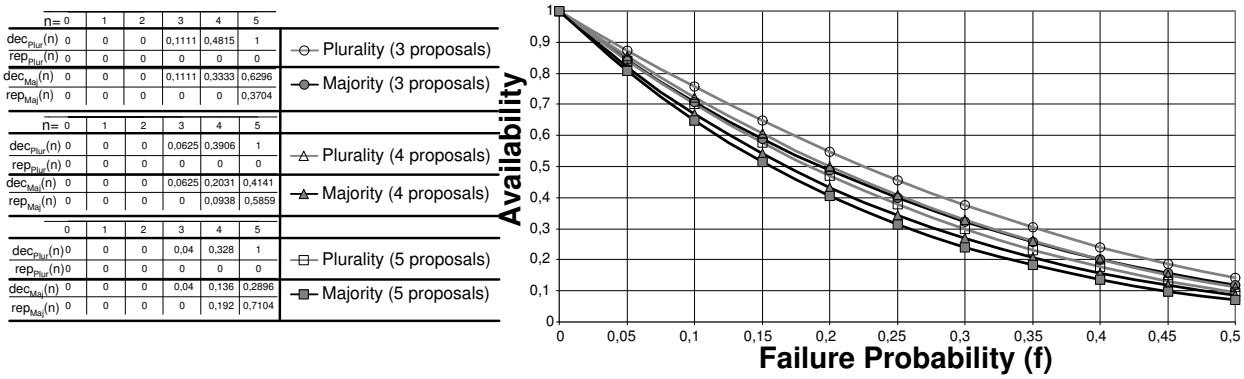

Figure 2: Availability of $\mathcal{E}_{Maj}$ and $\mathcal{E}_{Plur}$ in a system with five processes.

5 Optimal classical coteries do not necessarily produce optimal epidemic coteries

One might be tempted to believe that the epidemic-equivalent of the optimal classical coterie under a given system model is also the optimal EC. This section contradicts such an hypothesis with a simple combinatorial exercise. We analyze a system of five processes, p_1, p_2, p_3, p_4 and p_5, all in a single partition, with a uniform process failure probability $f < 0.5$. It is well known that the optimal classical coterie in this case is the majority coterie [PW95].

Let us denote the epidemic-equivalent of the majority coterie by $\mathcal{E}_{Maj}$. $\mathcal{E}_{Maj}$ comprises all the vote configurations with majority quorums (and no anti-quorums). It is easy to show that $\mathcal{E}_{Maj}$ is a subset of $\mathcal{E}_{Plur}$; it consists of the leftmost column of configurations in Figure 1.

Let us now try to determine the availability of $\mathcal{E}_{Maj}$, i.e. the probability that an EQS using $\mathcal{E}_{Maj}$ eventually decides. At each election, an EQS using $\mathcal{E}_{Maj}$ in a single partition model with n correct processes will either (a) decide a value, with probability $dec_{\mathcal{E}_{Maj}}(n)$; (b) determine that the current election is indecisive and start a new one, with probability $rep_{\mathcal{E}}(n)$; or (c) halt. The values of $dec_{\mathcal{E}_{Maj}}(n)$ and $rep_{\mathcal{E}_{Maj}}(n)$ may be obtained by simple combinatorics, depending on the number of proposed values. Figure 2 presents such probabilities for 3, 4 and 5 values proposed, along with the resulting availability (1) of $\mathcal{E}_{Maj}$. (We omit the cases of 0, 1 and 2 values proposed since both ECs behave identically in such conditions.)

Therefore, for each $n = 1..5$, the probability of an eventual decision corresponds to the probability of any finite (possibly empty) sequence of indecisive elections, followed by one decisive election. Such a probability is the sum of a geometric series, given by $dec_{\mathcal{E}_{Maj}}(n) \sum_{i=0}^{\infty} (rep_{\mathcal{E}_{Maj}}(n))^i = (dec_{\mathcal{E}_{Maj}}(n))/(1 - rep_{\mathcal{E}_{Maj}}(n))$. Since the probability of having exactly n correct processes (out of five) is $\binom{5}{n}(1-f)^n\ f^{(5-n)}$, then the availability of $\mathcal{E}_{Maj}$ is given by:

$$\sum_{n=0}^{5} \frac{\binom{5}{n} (1-f)^n\ f^{(5-n)} dec_{\mathcal{E}_{Maj}}(n)}{1 - rep_{\mathcal{E}_{Maj}}(n)} \tag{1}$$

As an alternative to $\mathcal{E}_{Maj}$, consider the plurality EC, $\mathcal{E}_{Plur}$. Obtaining the availability of an EQS running $\mathcal{E}_{Plur}$ is simpler than for $\mathcal{E}_{Maj}$, since indecisive elections never occur in $\mathcal{E}_{Plur}$ (as $rep_{\mathcal{E}_{Plur}}(n)$ is always null). In other words, $\mathcal{E}_{Plur}$ either decides in the first election or halts.

Again, by simple combinatorics, we present $dec_{\mathcal{E}_{Plur}}(n)$ and the availability obtained by (1) in Figure 2. A comparison of availability of both ECs shows that, for a system of 5 processes, $\mathcal{E}_{Plur}$ always achieves a higher availability (for $0 \leq f < 0.5$). Hence, $\mathcal{E}_{Maj}$ is not the optimal EC in a single partition model. This evidence raises, of course, the question of whether $\mathcal{E}_{Plur}$ is the optimal epidemic coterie or not; not only in the single partition model, but also in other models.

6 EQSs may outperform CQSs also in partition-free settings

EQSs have been proposed as a solution to provide higher availability in frequently partitioned environments such as mobile networks. However, a careful look at EQSs shows that, under certain circumstances, EQSs are also able to decide faster than CQSs. Most importantly, such circumstances include also partition-free settings such as fixed networks.

Consider an EQS using the epidemic majority coterie and a CQS using the classical majority coterie. Further, assume a situation of no contention (i.e. a single value is proposed) where a majority of processes is accessible. In an EQS, the process proposing the single value will have round of contacts with the quorum processes. After such a single round, the proposer will know that a quorum has voted for the value and will, thus, decide the value. In contrast, the CQS requires a number of rounds equal to the number of phases of the corresponding vote obtention protocol (for instance, two with 2PC).

The above argument suggests that the scope of EQSs is wider than the domain of mobile networks, and therefore one should also consider employing EQSs in strongly-connected networks. Provided that the probability of contention is low, as happens with many distributed systems that resort to quorum systems (e.g. write sharing is rare in replicated file systems [LS90]), EQSs are a faster alternative than CQSs in such networks. Should contention be sufficiently frequent, then the possibility of tied indecisions can be too high as a penalty, and the choice should return to CQSs.

7 Concluding Remarks

Although recent work has proposed EQSs, they are neither well defined nor well studied. We have shown evidence of previously undocumented properties and trade-offs that are inherent of EQSs and are not observable in their classical counterpart. We believe our claims call for a deeper analysis of this promising field of distributed computing and, in particular, mobile computing. Addressing the open questions that our discussion exposes is a first step in that direction. Namely, (1) under which conditions are EQSs effectively advantageous over CQSs; and (2) which are the optimal ECs.

References

[AW96] Yair Amir and Avishai Wool. Evaluating quorum systems over the internet. In *Symposium on Fault-Tolerant Computing*, pages 26–35, 1996.

[BF07] J. Barreto and P. Ferreira. The availability and performance of epidemic quorum algorithms. Technical Report 10/2007, INESC-ID, February 2007.

[CW96] Ing-Ray Chen and Ding-Chau Wang. Analyzing dynamic voting using petri nets. In *SRDS '96: Proceedings of the 15th Symposium on Reliable Distributed Systems (SRDS '96)*, page 44, Washington, DC, USA, 1996. IEEE Computer Society.

[Gif79] D. K. Gifford. Weighted voting for replicated data. In *Proceedings of the Seventh Symposium on Operating System Principles SOSP 7*, pages 150–162, Asilomar Conference Grounds, Pacific Grove CA, 1979. ACM, New York.

[HSAA03] JoAnne Holliday, Robert Steinke, Divyakant Agrawal, and Amr El Abbadi. Epidemic algorithms for replicated databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1218–1238, 2003.

[IK01] K. Ingols and I. Keidar. Availability study of dynamic voting algorithms. In *ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems*, page 247, Washington, DC, USA, 2001. IEEE Computer Society.

[JM90] S. Jajodia and David Mutchler. Dynamic voting algorithms for maintaining the consistency of a replicated database. *ACM Trans. Database Syst.*, 15(2):230–280, 1990.

[JM05] Flavio Paiva Junqueira and Keith Marzullo. Coterie availability in sites. In *DISC*, pages 3–17, 2005.

[Kel99] P. Keleher. Decentralized replicated-object protocols. In *Proc. of the 18th Annual ACM Symp. on Principles of Distributed Computing (PODC'99)*, 1999.

[LS90] Eliezer Levy and Abrahram Silberschatz. Distributed file systems: Concepts and examples. *ACM Computing Surveys*, 22(4):321–374, December 1990.

[PW95] David Peleg and Avishai Wool. The availability of quorum systems. *Information and Computation*, 123(2):210–223, 1995.