

# Understanding Epidemic Quorum Systems

**More details in the technical report:**

<http://www.gsd.inesc-id.pt/~jpbbarreto>

**João Barreto**

[joao.barreto@inesc-id.pt](mailto:joao.barreto@inesc-id.pt)

**Paulo Ferreira**

[paulo.ferreira@inesc-id.pt](mailto:paulo.ferreira@inesc-id.pt)

**INESC-ID Lisbon/Technical University of Lisbon**

<http://www.gsd.inesc-id.pt/>



# Why Epidemic Quorum Systems?

---

- **Epidemic Quorum Systems** [1,2] seem advantageous for **agreement** in weakly-connected environments
  - e.g. mobile and sensor networks.
  - agreement of a value from a set of proposed values is often required in distributed systems (e.g. replication protocols, distributed mutual exclusion, ..)
- **Whereas Classical Quorum Systems** require a *quorum* of processes to be simultaneously connected to agree...
- **Epidemic Quorum Systems** do not require simultaneous connection of quorum
  - Processes vote locally for one of the proposed values
  - Votes propagate epidemically
  - Processes decide one value based on the votes they collect



# Example (from the viewpoint of process $p_1$ )

	time →				
<p>5 processes with temporary partitions</p>	<p>Proposes X</p>	<p>Proposes Y</p>	<p>Proposes Z</p>		
<p><b>Epidemic Plurality</b> Winner must have more votes than the votes that any other value may collect in the worst case; ties are broken in favour of the process with the lowest identifier.</p>	<p><math>p_1</math> votes for X. No plurality yet.</p>	<p><math>p_1</math> knows that <math>p_1</math> and <math>p_2</math> have voted for X. No plurality yet.</p>	<p><math>p_1</math> knows that: <math>p_1</math> and <math>p_2</math> have voted for X, <math>p_5</math> for Y, and <math>p_4</math> for Z. X has a plurality of votes. <b>Hence <math>p_1</math> decides X.</b></p>		
<p><b>Epidemic Majority</b> Winner must have more than half the votes.</p>	<p><math>p_1</math> votes for X. No majority yet.</p>	<p><math>p_1</math> knows that <math>p_1</math> and <math>p_2</math> have voted for X. No majority yet.</p>	<p><math>p_1</math> knows that: <math>p_1</math> and <math>p_2</math> have voted for X, <math>p_5</math> for Y, and <math>p_4</math> for Z. No majority yet.</p>	<p><math>p_1</math> knows that: <math>p_1</math>, <math>p_2</math> and <math>p_3</math> have voted for X, <math>p_5</math> for Y, and <math>p_4</math> for Z. X has a majority of votes. <b>Hence <math>p_1</math> decides X.</b></p>	
<p><b>Classical Majority</b> Quorum of more than half the processes must be acquired.</p>	<p><math>p_1</math> fails to obtain quorum of 3 accessible processes.</p>	<p><math>p_1</math> fails to obtain quorum of 3 accessible processes.</p>	<p><math>p_1</math> fails to obtain quorum of 3 accessible processes.</p>	<p><math>p_1</math> fails to obtain quorum of 3 accessible processes.</p>	<p><math>p_1</math> obtains quorum of 3 accessible processes. <b>Hence <math>p_1</math> decides X.</b></p>

# However, Epidemic Quorum Systems are not well studied...

---

Previous work does not answer the following questions:

1. Under **which conditions** do Epidemic Quorum Systems become **advantageous** over Classic Quorum Systems?
2. What are the **inherent trade-offs** of Epidemic Quorum Systems?
3. How can we precisely **characterize the liveness** of Epidemic Quorum Systems?
  - In particular, how do proposed Epidemic Quorum Systems compare in terms of availability and performance?
4. What is the **best** Epidemic Quorum System?

**And known results for Classical Quorum Systems [3] are not valid for Epidemic Quorum Systems.**



# Defining an Epidemic Coterie (EC)

Intuitively, we define an EC as:

- a collection of **<Quorum; Anti-Quorums>** pairs
- where:
  1. for each pair, the Quorum and every Anti-Quorum are disjoint sets of processes
  2. and, given a pair in the EC, we cannot:
    1. pick one of its Anti-Quorums and make it the new Quorum (swapping it with the original Quorum),
    2. add arbitrary processes (not yet included in the pair) to any Quorum/Anti-Quorum,
    3. and obtain a different pair of the EC.

**Definition 1.** *Vote set, vote configuration and q-vote configuration*

A vote set  $s$  is a set of processes,  $s \subseteq U$ .

A vote configuration  $c$  is a set of non-empty vote sets,  $c = \{s_1, s_2, \dots, s_n\}$ , such that  $\forall s_i, s_j \in c, s_i \cap s_j = \emptyset$ .

A q-vote configuration  $qc$  is a pair,  $qc = \langle Q_{qc}, \{A_{qc}^1, A_{qc}^2, \dots, A_{qc}^{n_{qc}}\} \rangle$ , where  $\{Q_{qc}, A_{qc}^1, \dots, A_{qc}^{n_{qc}}\}$  is a vote configuration.

**Definition 2.** *Potential vote set*

Let  $s$  be a vote set. The potential vote set of  $s$  in a vote configuration  $c$ , denoted  $[s]_c$ , is defined as  $[s]_c = s \cup (U \setminus (\bigcup \{s : s \in c\}))$ . In particular,  $[\emptyset]_c = (U \setminus (\bigcup \{s : s \in c\}))$ .

Further, the potential vote set of  $s$  in a q-vote configuration  $qc$ , denoted  $[s]_{qc}$ , is defined as  $[s]_{qc} = s \cup (U \setminus (Q_{qc} \cup A_{qc}^1 \cup \dots \cup A_{qc}^{n_{qc}}))$ .

**Definition 3.** *Coverage and potential coverage between q-vote configurations*

Let  $qc$  and  $qd$  be q-vote configurations.

We say  $qc$  covers  $qd$ , or  $qc \succ qd$ , if and only if:

1.  $Q_{qc} \supseteq Q_{qd}$ , and
2. There exists an injective function  $f : \{1, \dots, n_{qd}\} \rightarrow \{1, \dots, n_{qc}\}$  such that  $\forall 1 \leq k \leq n_{qd} : A_{qc}^{f(k)} \supseteq A_{qd}^k$ .

We say  $qc$  may cover  $qd$ , or  $qc \triangleright qd$ , if and only if:

1.  $[Q_c]_{qc} \supseteq Q_{qd}$ , and
2. There exists an injective function  $f : \{1, \dots, n_{qd}\} \rightarrow \{1, \dots, n_{qc}\}$  such that  $\forall 1 \leq k \leq n_{qd} : [A_{qc}^{f(k)}]_{qc} \supseteq A_{qd}^k$  or  $[\emptyset]_{qc} \supseteq A_{qd}^k$ .

**Definition 4.** *Epidemic Coterie*

Let  $\mathcal{E}$  be a non-empty set of q-vote configurations.  $\mathcal{E}$  is an Epidemic Coterie (EC) if,  $\forall c, d \in \mathcal{E} : c \neq d$ :

1.  $\forall j : \langle A_c^j, \{Q_c\} \cup \{A_c^i : i \neq j\} \rangle \not\triangleright d$ , and
2.  $\langle \emptyset, \{Q_c\} \cup \{A_c^1, \dots, A_c^{n_c}\} \rangle \not\triangleright d$ , and
3.  $c \not\succeq d$ .

# Epidemic Quorum Algorithm

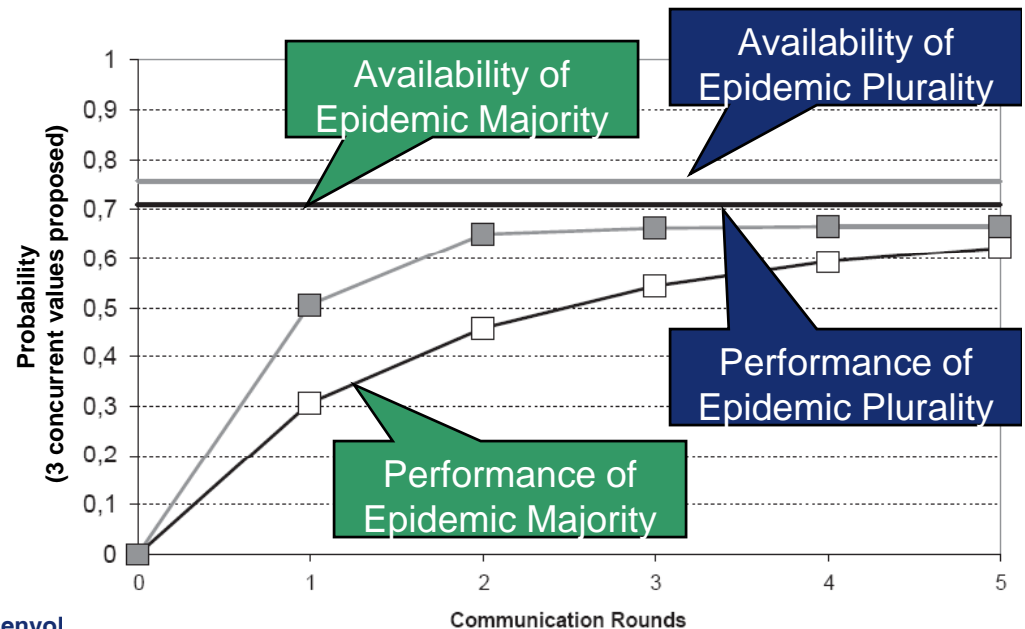
---

- Processes propose and vote for a value by adding their vote to the **local vote set of that value**
  - A process may only vote once and may not retract its vote in each election
- Vote sets propagate epidemically
- Once a process is able to compose a
  - $\langle$ Quorum; Anti-Quorums $\rangle$  pair from the EC using its local vote sets, it **decides the value corresponding to the vote set that is the Quorum**
- Once a process determines that no pair in the EC may ever be composed (with the still unknown votes), it **starts a new election** (with empty vote sets)



# Characterizing Availability and Performance

- We assume:
  - An asynchronous system
  - Probability of non-Byzantine permanent failure is uniform and constant
  - Probability of two processes being inaccessible due to transient partitioning is uniform and constant
  - $Z$  values proposed, with uniform probability of being voted for
- Given an EC, determine the probability distributions:
  - Probability that a process, **having collected  $N$  votes, decides some value**
  - Probability that a process, **having collected  $N$  votes, starts a new election**
- Having the probability distributions we have obtained **analytical** expressions for:
  - Probability that a process **eventually** decides some value (**availability**)
  - Probability that a process decides some value **within  $r$  communication rounds (performance)**



## Future Work

---

- Use the formalism to devise novel ECs and study their availability and performance
- Compare Epidemic and Classical Quorum Systems
- What EC offers optimal availability and/or performance?

## References

---

- [1] JoAnne Holliday, Robert Steinke, Divyakant Agrawal, and Amr El Abbadi. *Epidemic algorithms for replicated databases*. IEEE Trans. on Knowledge and Data Engineering, 2003.
- [2] P. Keleher. *Decentralized replicated-object protocols*. 18th Annual ACM Symp. on Principles of Distributed Computing (PODC'99), 1999.
- [3] Yair Amir and Avishai Wool. *Evaluating quorum systems over the internet*. Symposium on Fault-Tolerant Computing, 1996.

