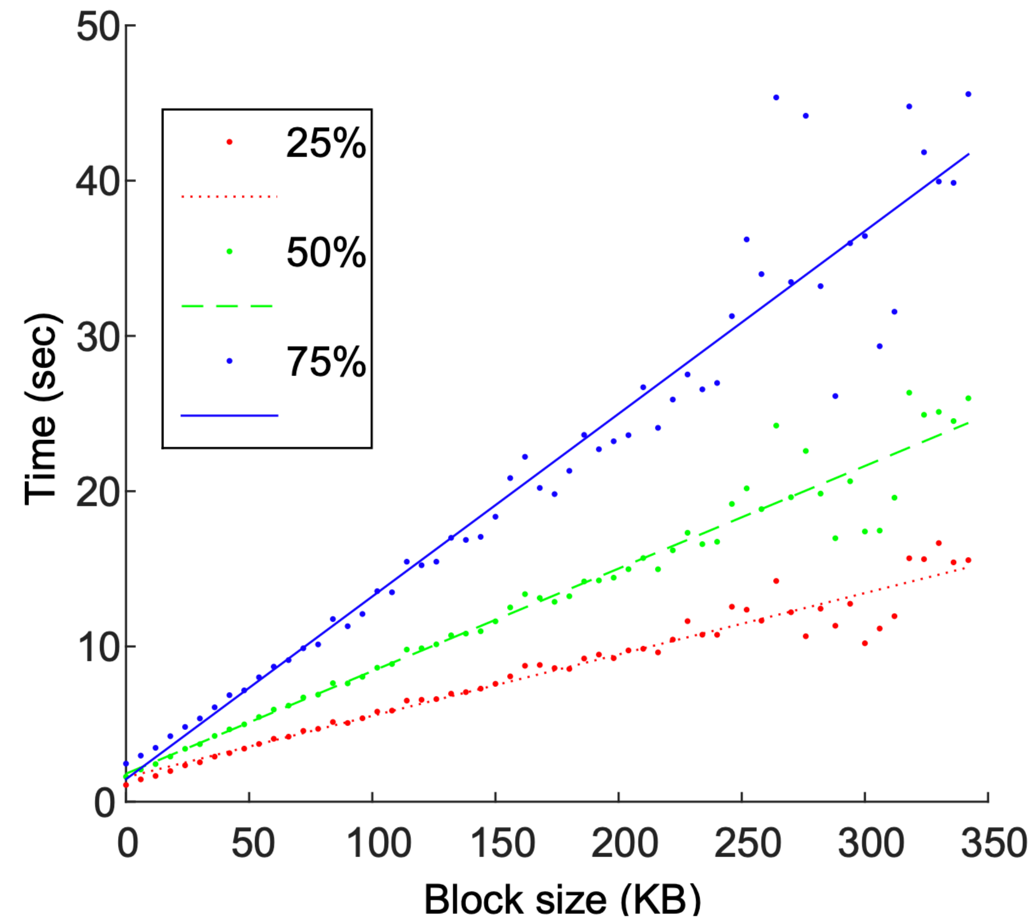
A Glance at BBFT

Where are we?

- Draft released (v1.0)
    - https://github.com/bystackcom/BBFT-Whitepaper/blob/master/whitepaper.pdf
- Implementation in-progress
- Many details to figure out
- Open to suggestion/diss

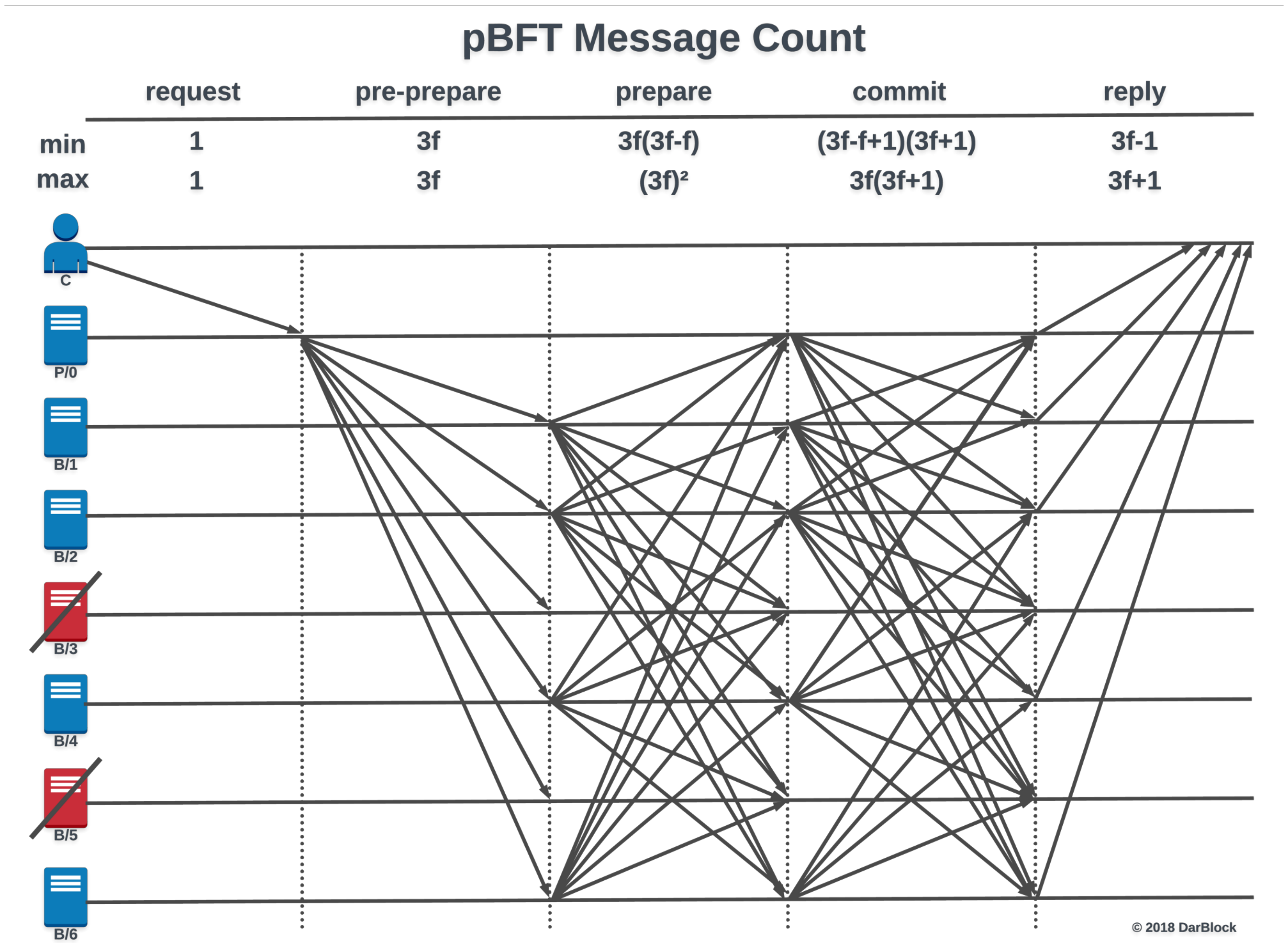
Why BFT?

- Why not POW?
  - Energy waste, greenhouse effect?
  - A tale of TPS: BTC: 3-7; ETH: 7-15; VISA: 2k+
  - Block size: storage, fork
  - Block time: similar effect
  - Full decentralization
    - Any node can generate block
    - Network perf = node perf
- Finality = No more fork



Sompolinsky et al. 2013

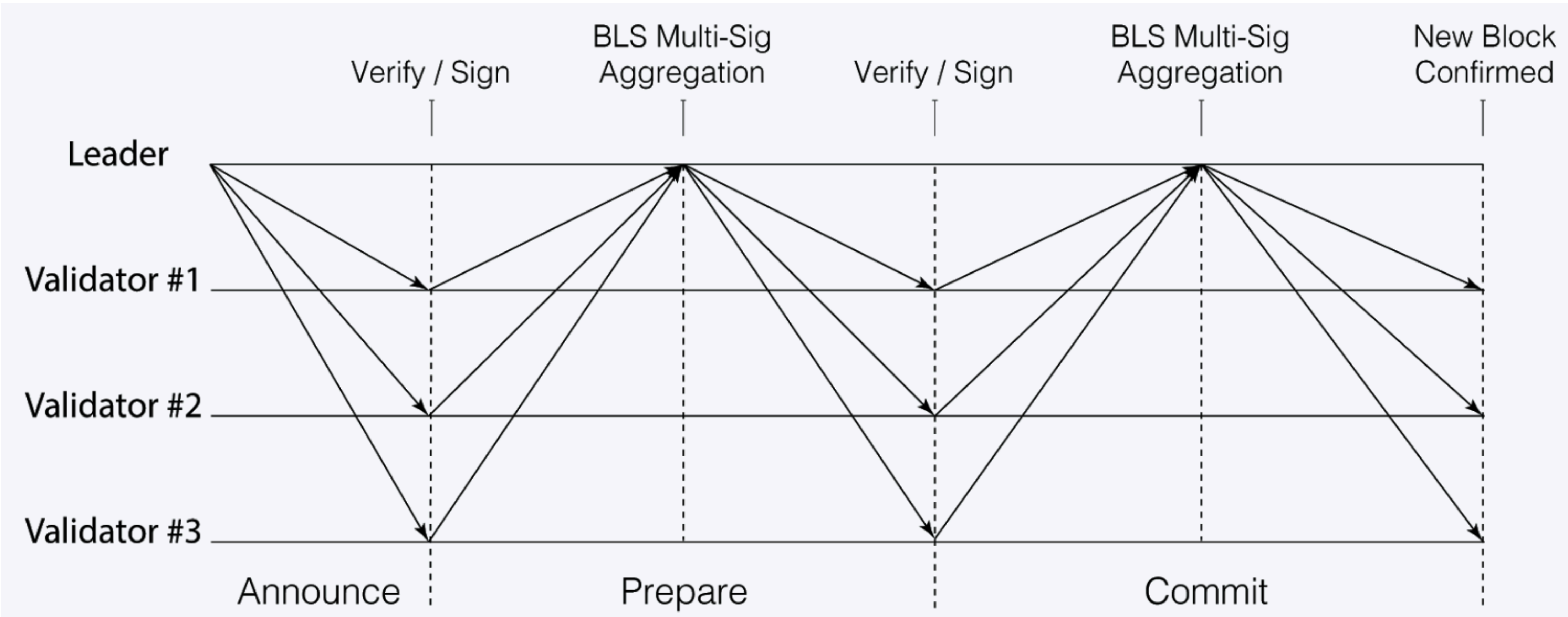
Why another BFT?



PBFT: do not scale @ O(n^2)

ref: https://medium.com/coinmonks/pbft-understanding-the-algorithm-b7a7869650ae

Why another BFT?



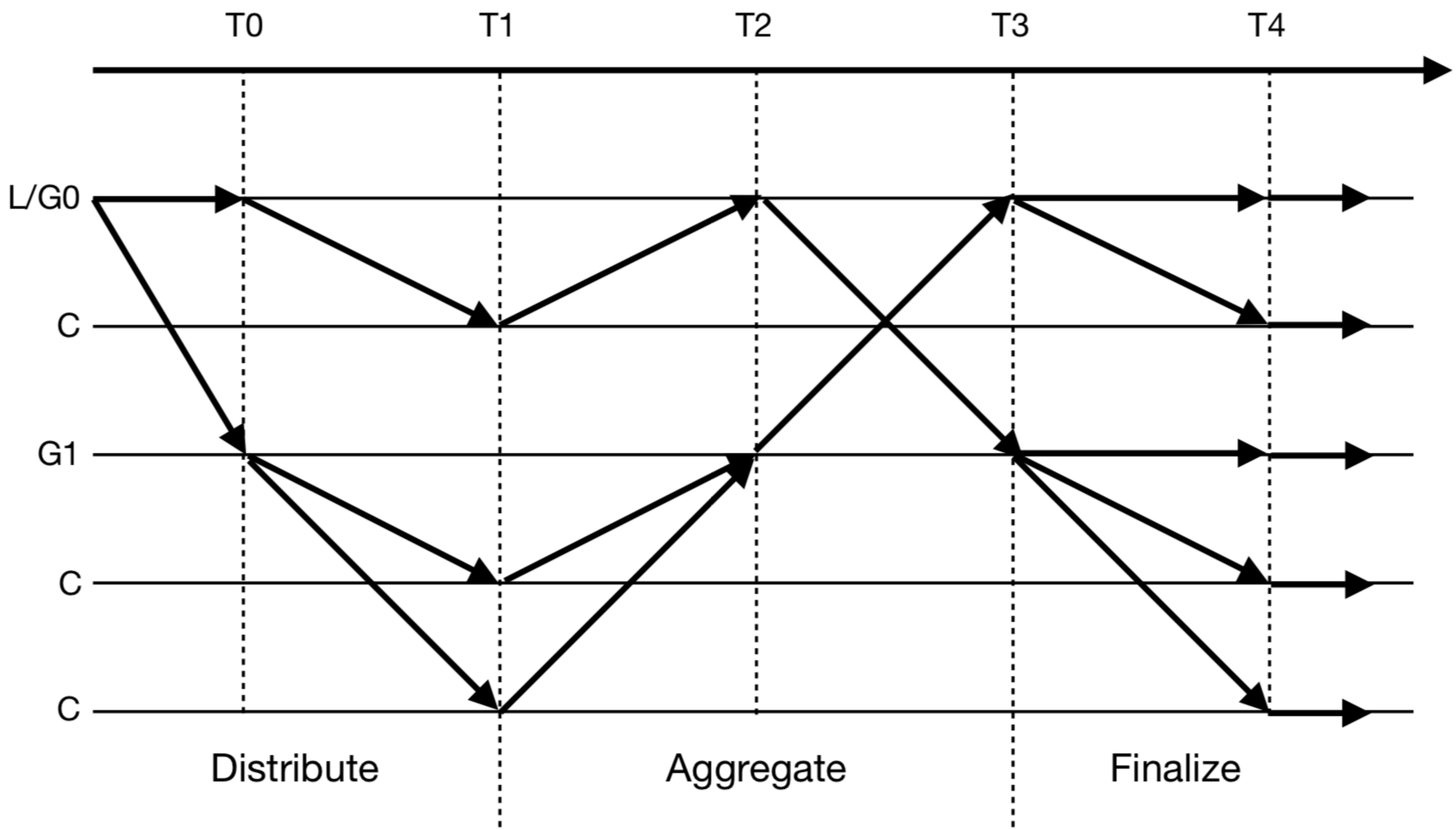
ref: https://harmony.one/whitepaper.pdf

FBFT: everything goes through the pivotal leader

What the heck is BBFT?

- Message passing
- Signature aggregations
- Network zoning

Consensus flow



Consensus flow

- Total messages: $m^2 - 2m + 3n - 2$
- Complexity under condition: $1 < = m \leq \sqrt{n} \Rightarrow O(n)$
- Special cases:
  - m = n: PBFT
  - M = 1: FBFT

BLS key aggregation

- Setup: We choose an efficiently computable non-degenerate bilinear pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$[3], where $\mathbb{G}_0$, $\mathbb{G}_1$, and $\mathbb{G}_T$ are groups of prime order q. Let g_0 and g_1 be the generators of $\mathbb{G}_0$ and $\mathbb{G}_1$ respectively. We also choose two hash functions $H_0 : M \to \mathbb{G}_0$, a mapping from message space to $\mathbb{G}_0$, and $H_1 : \mathbb{G}_1^n \to \mathbb{Z}_q$.

- KeyGen(): Every participant picks a secret key $sk \in \mathbb{Z}_q$, and shares its public key $pk = g_1^{sk \cdot H_1(g_1^{sk})}$ with each other. $pk \in \mathbb{G}_1$.

- Sign(sk, m): Participant with secret key sk signs m and outputs signature $\sigma = H_0(m)^{sk \cdot H_1(g_1^{sk})}$. $\sigma \in \mathbb{G}_0$.

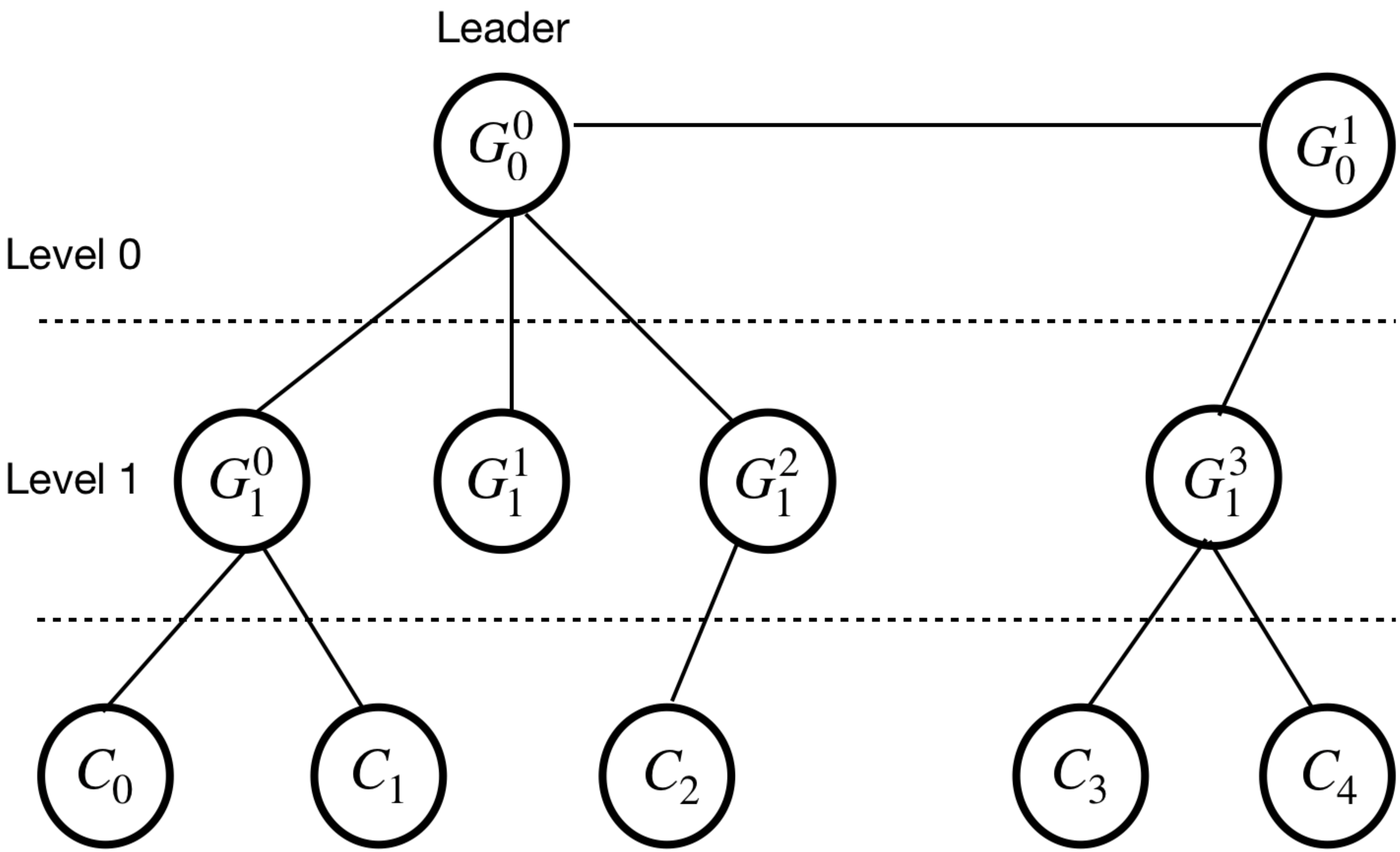
- Aggregate(σ_1, σ_2,...,σ_n): Upon receipt of multiple signatures on message m, the aggregated signature $\sigma = \prod_{i=1}^{n} \sigma_i$ is computed as the product of individual signatures σ_i. $\sigma \in \mathbb{G}_0$.

- Verify(σ, m): Participant verifies an aggregated signature on m by first computing aggregated public key $\kappa = \prod_{i=1}^{n} pk_i$, as the product of individual public key pk_i. $\kappa \in \mathbb{G}_1$. Then verification is done by the two-pairing check $e(\sigma, g_1) = e(H_0(m), \kappa)$.

BLS key aggregation

- Setup: $e : \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T \quad H_0 : M \rightarrow \mathbb{G}_0$
- Key-gen： $pk = g_1 \times sk$
- Sign: $S = sk \times H_0(m)$
- Verify: $e(pk, H_0(m)) = e(g_1 \times sk, H_0(m)) = e(g_1, sk \times H_0(m)) = e(g_1, s)$
- With aggregation: $e(pk_0 \cdot pk_1, H_0(m_0) \cdot H_0(m_1)) = e(g_1, s_0) \cdot e(g_1, s_1)$

Network zoning



Network zoning

- Topology graph: weighted undirected graph
- Topology tree: minimal spanning tree
- Gateway nodes handle message aggregation and relay
- Only top level nodes do full message exchange
- Faulty nodes? incentive, penalty
- Use case:
  - communication cost of top level far exceeds other levels.
  - Not too many levels

Thank You