Asymptotically Optimal Validated Asynchronous Byzantine Agreement

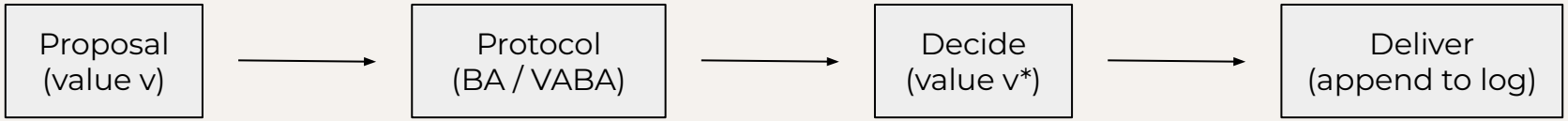
Vincent Huynh, Steven Trujillo, Billy Ouattara, Yu-Chieh Chao

Introduction

General Goal: Make a distributed system behave like one server

Standard approach

- **Atomic Broadcast:** All replicas get the same operations in the same order
- **State Machine Replication:** All replicas move through the same sequence of states

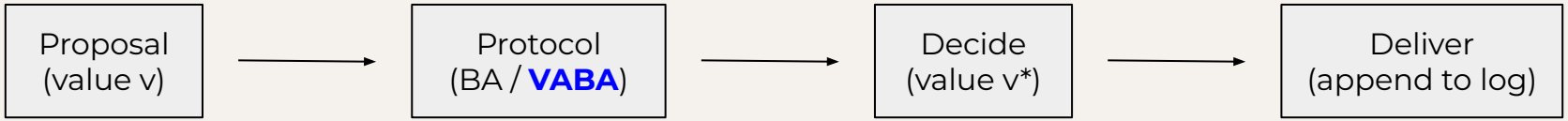


Introduction

General Goal: Make a distributed system behave like one server

Standard approach

- **Atomic Broadcast:** All replicas get the same operations in the same order
- **State Machine Replication:** All replicas move through the same sequence of states



Validated Asynchronous Byzantine Agreement

Introduction

Cachin et al. [10]

- Defined Atomic Broadcast & VABA in the authenticated asynchronous model.
- Resilience: $n = 3f + 1$ ⟶ Optimal
- Running Time: $O(1)$ ⟶ Optimal
- Expected Word Communication: $O(n^3)$ ⟶ Not optimal

Specific Goal:

- Achieve $O(n^2)$ on Expected Word Communication while maintaining optimality on Resilience and Time.

[10] Secure and efficient asynchronous broadcast protocols

Introduction

Byzantine Agreement (BA)

- **Validity:** If all honest parties propose value v, then they must decide value v when terminate.
  - **Weakness**
    - Decide on a value proposed by faulty party.
    - Decide a constant value => No progress
- **Synchronous:** All messages are delivered within Δ
  - Synchronous
  - Eventual synchronous
  - Partially synchronous
  - Weakly synchronous (e.g. PBFT)
  - **Weakness**
    - Real-World networks can have unpredictable message delays

Introduction

Byzantine Agreement (BA)

- **Validity:** If all honest parties propose value v, then they must decide value v when terminate.
- **Synchronous:** All messages are delivered within Δ

Validated Asynchronous Byzantine Agreement (VABA)

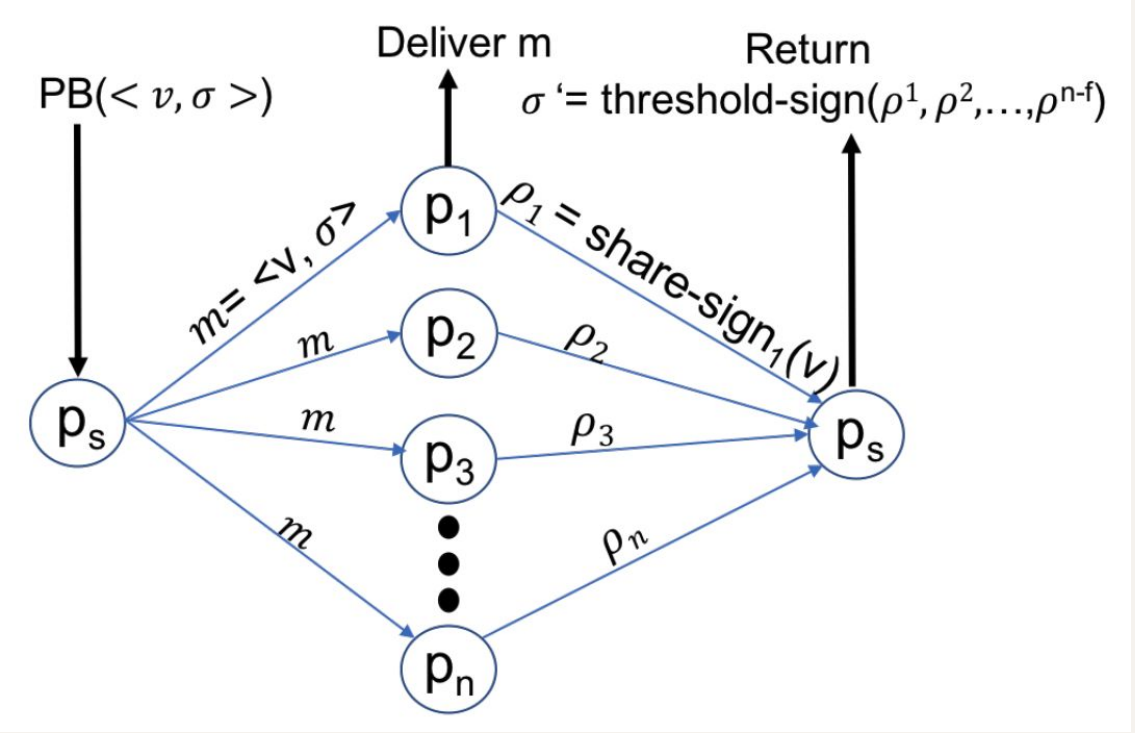
- **Quality:** The decided value v was proposed by an honest party with $p \geqq 0.5$
- **External Validity:** If an honest party decides a value v, then EX-PB-VAL(v) = True.
- **Asynchronous:** No timing assumption on the network.

Provable-Broadcast

Sender P_s
- v: value
- σ: proof

Receiver (other parties)
- ρ: signature share

Sender P_s
- wait until receiving n-f signature shares
- σ': threshold signature



The Structure of a VABA View

A VABA view is comprised of three core phases:

- **Leader-nomination**
  - Provide all parties/replicas the opportunity to promote a value.
- **Leader-election**
  - Randomly selects a party as leader to promote their value.
- **View-change**
  - Synchronize all parties on the chosen leader’s progress to safely finalize the value or move to the next view.

Leader Nomination

- Every party acts as a leader of its own **Proposal-Promotion** instance.
- All Proposal-Promotion instances run in parallel.
- Each party tries to promote its own value.

Goal: Ensure that at least one honest value makes sufficient progress to be safely used later.

Proposal-Promotion Overview

- Each leader runs four **Provable-Broadcast** steps:
  - Initial broadcast
  - Key
  - Lock
  - Commit
- What strengthens at each stage is the cryptographic evidence attached to the value.

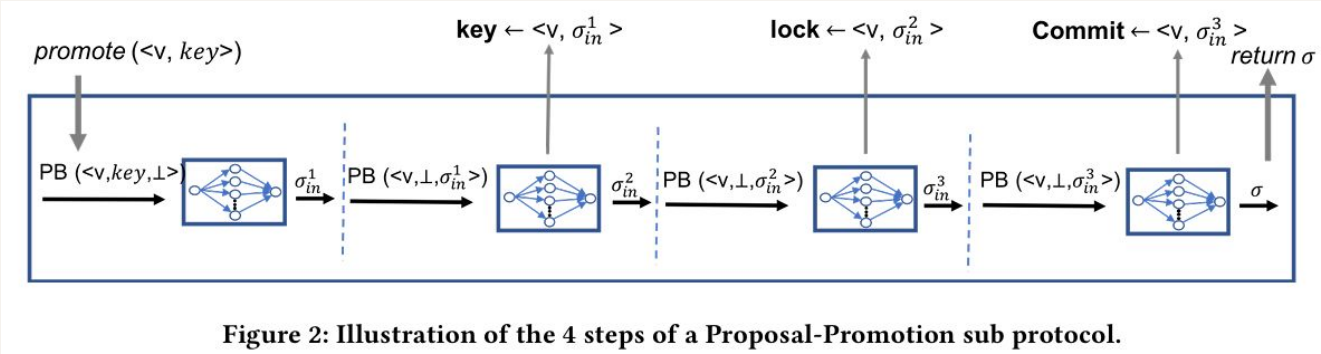


Figure 2: Illustration of the 4 steps of a Proposal-Promotion sub protocol.

Proposal-Promotion: Initial Broadcast

- Leader distributes its value with external validity proof.
- Parties verify and send back signature shares.
- Leader builds a certificate after **2f + 1** signature shares proving the value is valid.

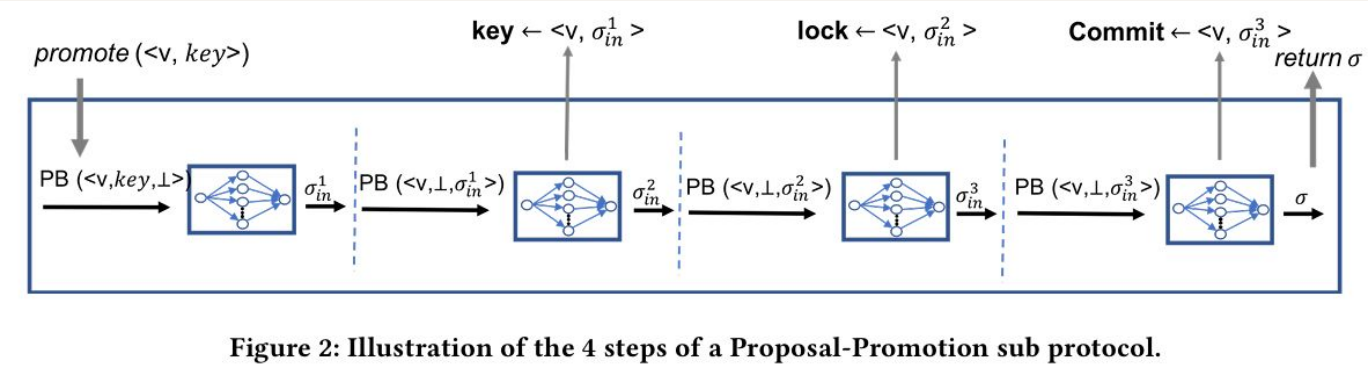


Figure 2: Illustration of the 4 steps of a Proposal-Promotion sub protocol.

Proposal-Promotion: Key Stage

- Parties deliver the value and store it as key.
- Leader creates Key certificate after receiving **2f + 1** signature shares which act as proof:
  - “Enough honest parties have seen this value.”
- This marks the value as a legitimate candidate.

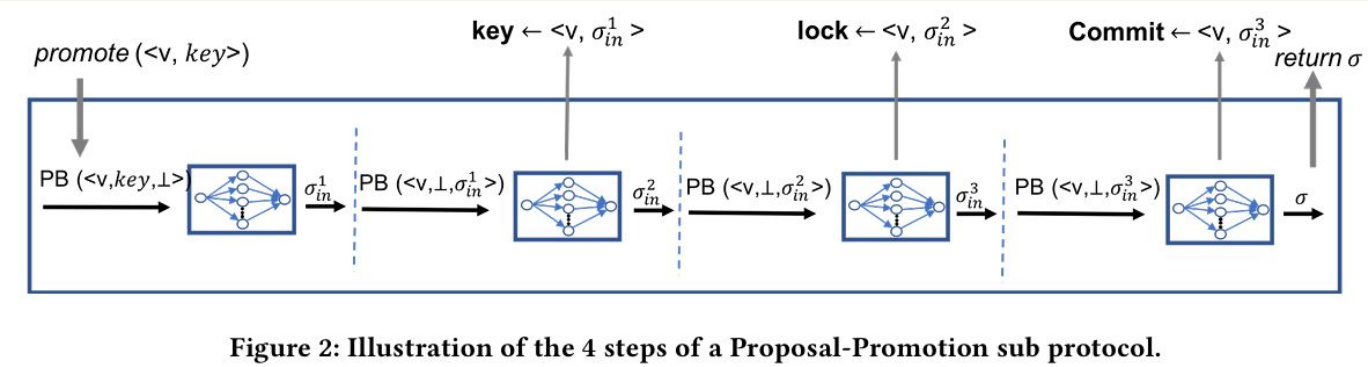


Figure 2: Illustration of the 4 steps of a Proposal-Promotion sub protocol.

Proposal-Promotion: Lock Stage

- Parties deliver and store the lock.
- Lock certificate proves enough parties are committed to protecting this value.
- Conflicting values cannot progress safely after this point.

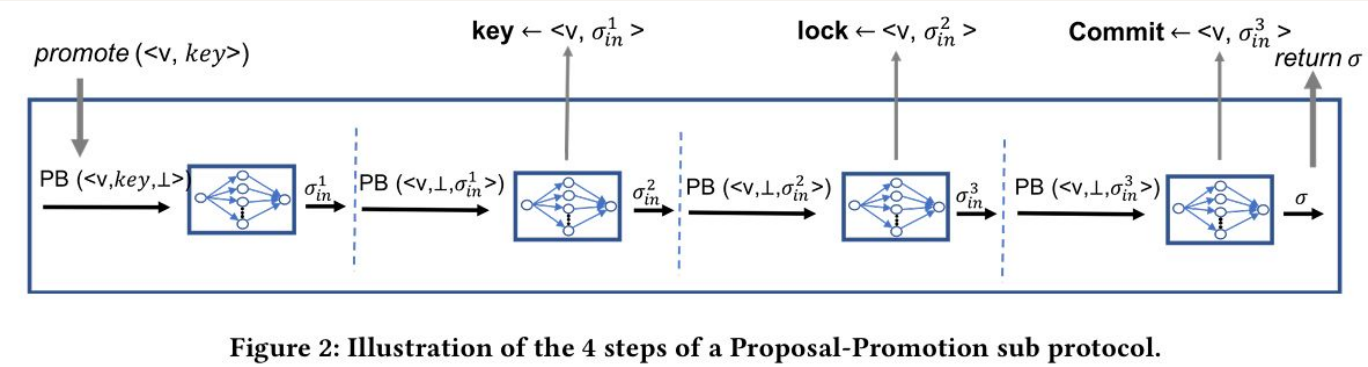


Figure 2: Illustration of the 4 steps of a Proposal-Promotion sub protocol.

Proposal-Promotion: Commit Stage

- Parties deliver and store the commit.
- Commit certificate proves the value is now finalizable.
- No other value can compete with a committed value.

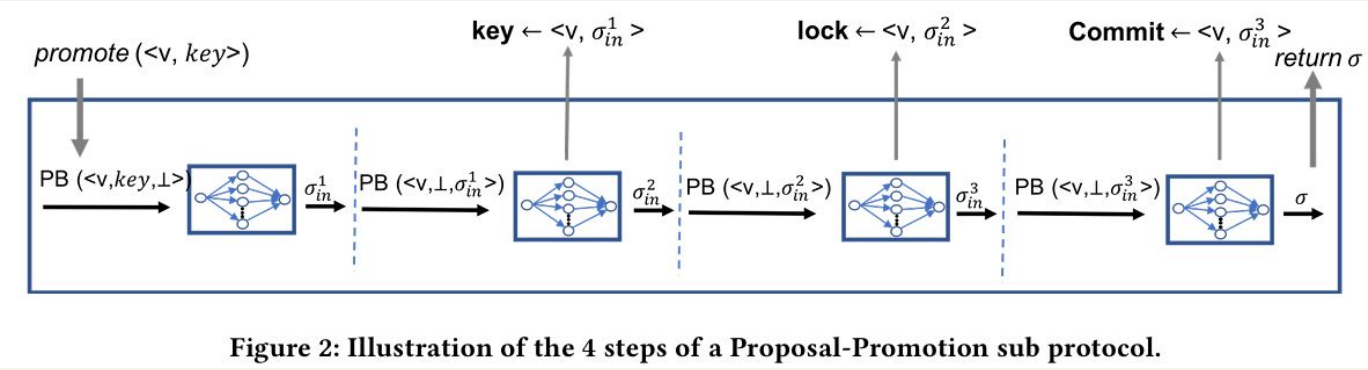


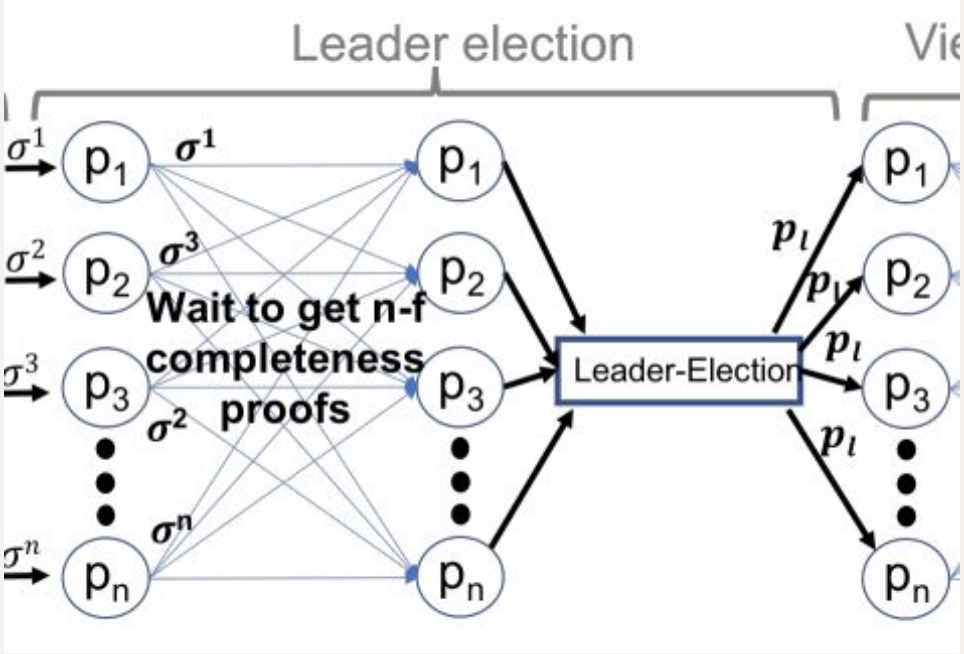
Figure 2: Illustration of the 4 steps of a Proposal-Promotion sub protocol.

The Coin Toss

- Includes private function coin-share$_i$
- Includes two public function coin-share-validate & coin-toss
- Given f + 1 valid coin shares, coin toss return a unique & pseudorandom number
    - Range 1 to n

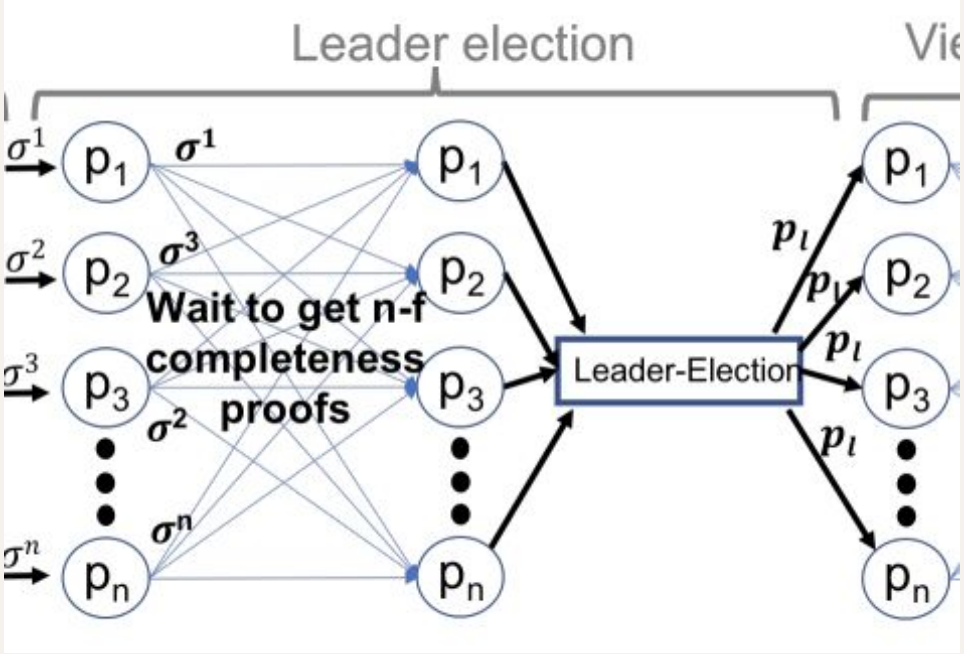
Entering Leader Election

- Broadcast Done(value, proof)
- Collect **n - f** Done messages
- Send Skip-share message
- Collect **n - f** valid Skip-shares
- Send Skip message
- Collect valid Skip message and move to Leader Election



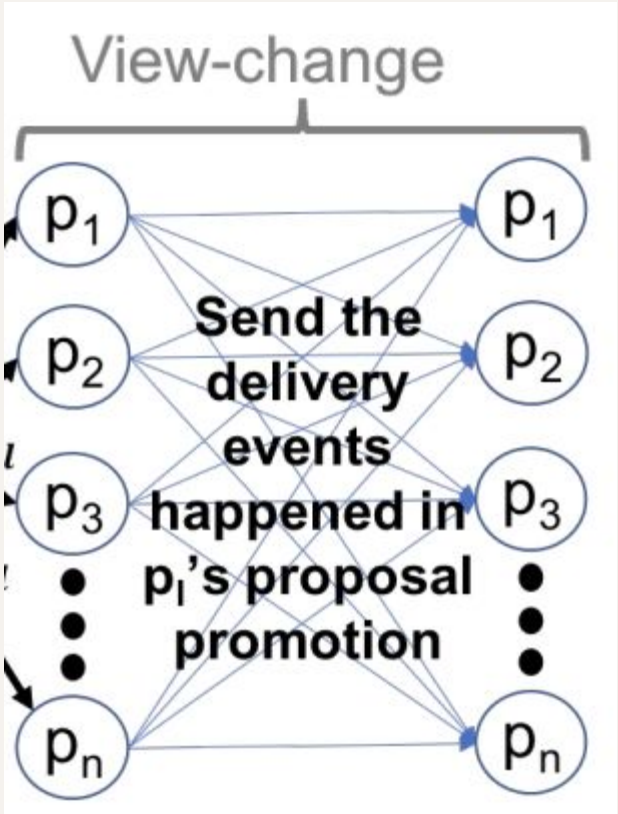
Leader Election

- Collect **f + 1** valid coin shares
- Then Coin toss
  - Unique & pseudorandom value
  - Range 1 to n
- Move to View-Change Phase



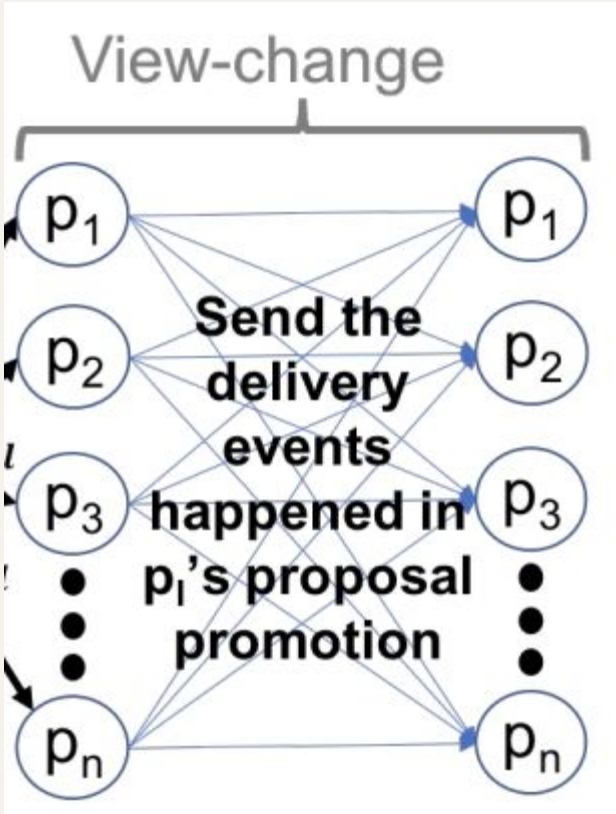
Good View Change

- Broadcast View-change message
- Collect **n - f** view-change messages
- Process these messages
- If valid **COMMIT** proof for value v decide value v



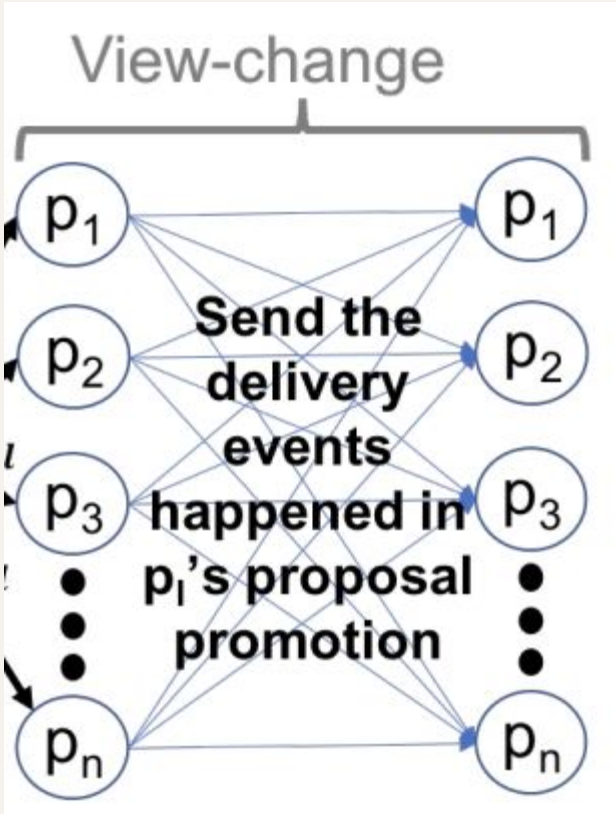
Bad View Change

- Leader was slow, crashed, or malicious
- If Leader contains a LOCK, update local LOCK variable
- If Leader contains no LOCK but has a KEY, update local KEY variable
- If Leader is empty, keep previous state



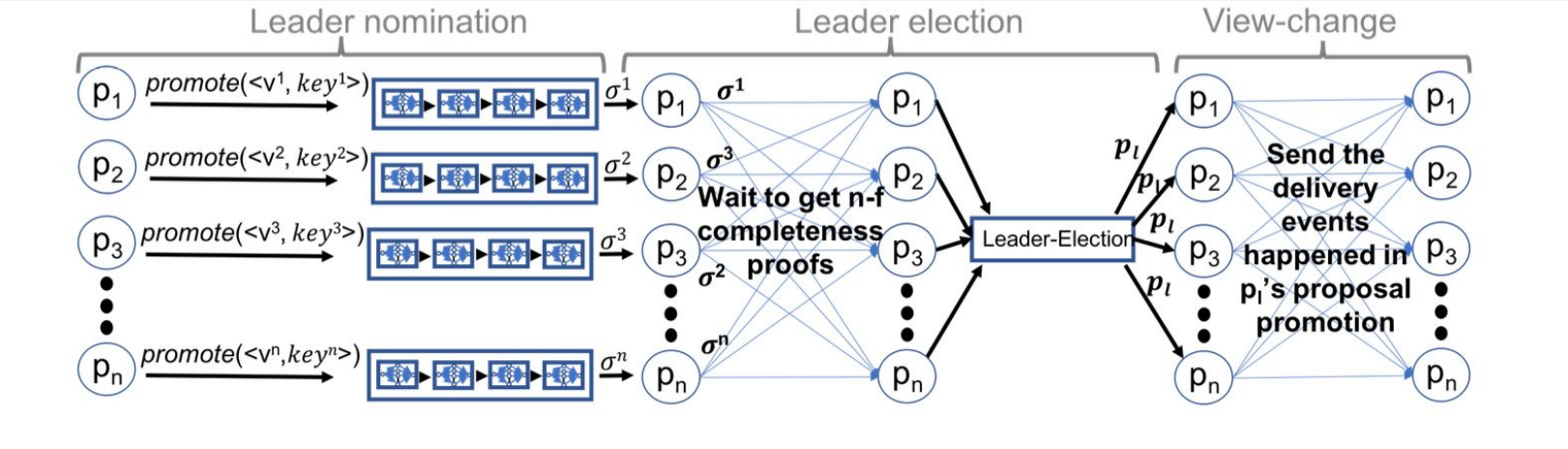
Key Locking Mechanism

- Lock Safety: Prevents Overwriting Commitment
  - If value v is committed LOCK will be set to current view
- Key Safety: Prevents Conflicting Proposals
  - If value v is committed, no party can commit a different value in future views
- Key Progress: Guarantees Liveness:
  - Locked part will always have a valid KEY to promote in the next view



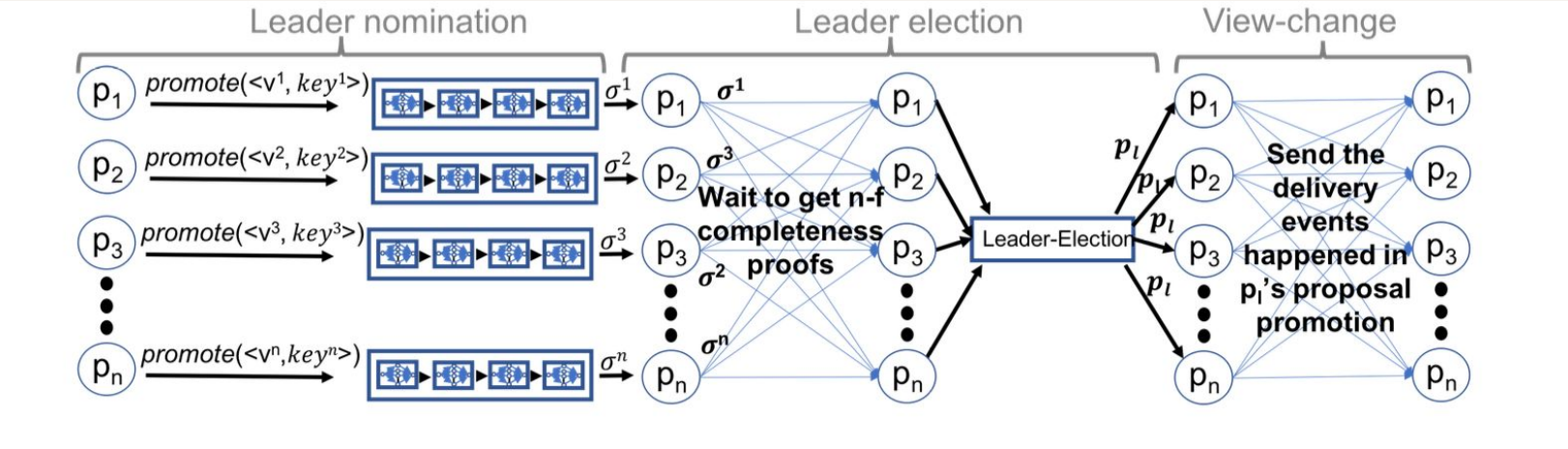
Complexity Analysis

- Resilience: f < n/3
- Time: Expected O(1)
- Communication: Expected $O(n^2)$



Limitations

- Heavy Cryptography (The CPU Bottleneck)
- Probabilistic Latency dependent on a coin toss



Conclusion

- First protocol to achieve Optimal Resilience, Optimal Time, AND Optimal Communication in the Asynchronous authenticated setting
- Replaced timeouts (PBFT) with knowledge of progress (Key/Lock/Commit)
- Replaced leader-generated proofs (PBFT) with replica-enforce history (Lock Safety)