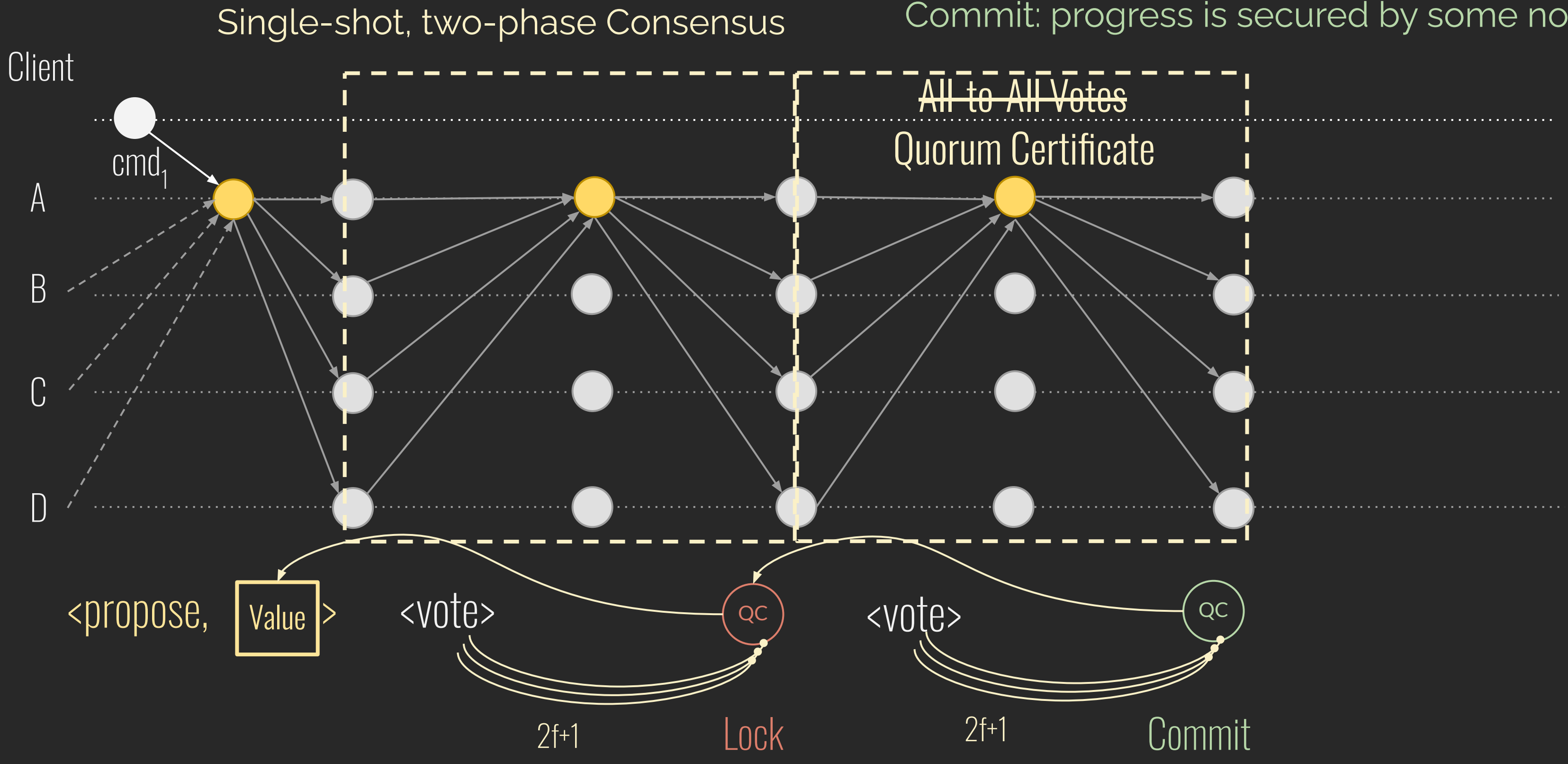
Lessons from HotStuff

Dahlia Malkhi, Maofan “Ted” Yin

HotStuff, revisited.

Lock: value is consistent, but info may be lost

Commit: progress is secured by some node (?)

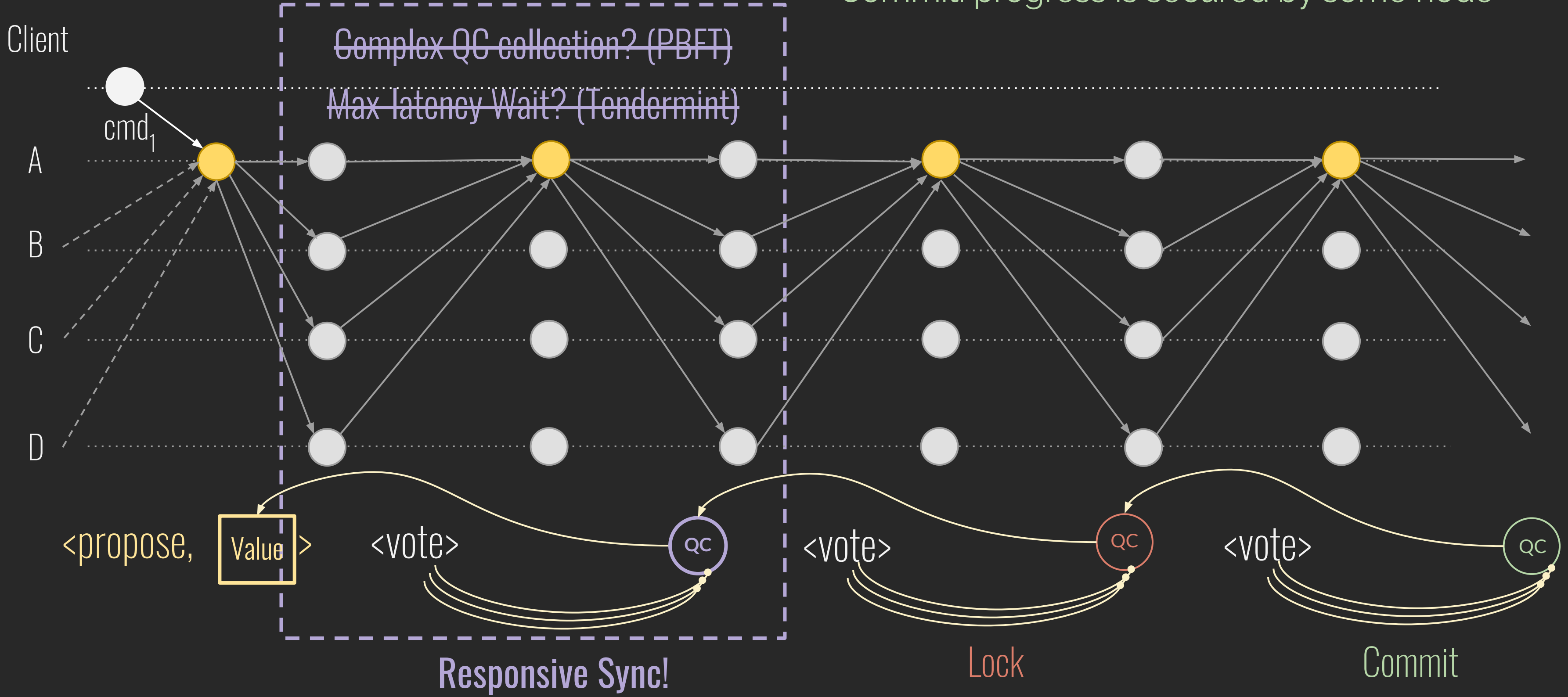


HotStuff, revisited.

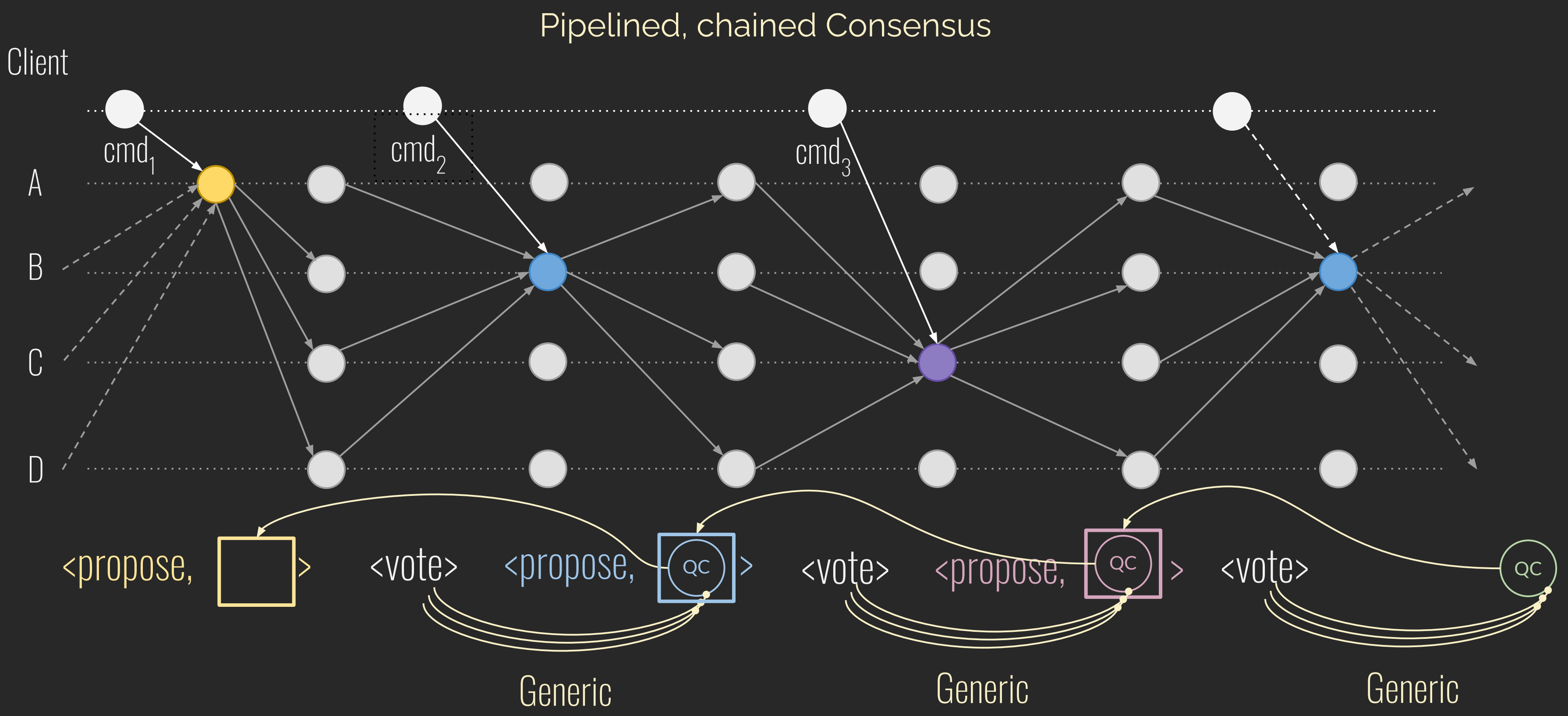
Sync: the highest locked value is found

Lock: value is consistent, but info may be lost

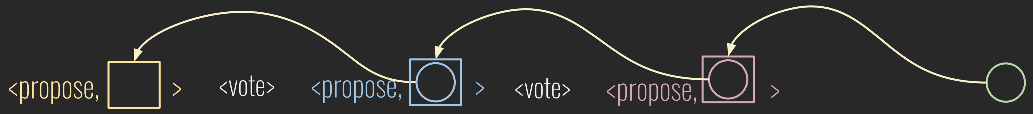
Commit: progress is secured by some node



HotStuff, revisited.



Key Contributions



- Pipelining
    - No explicit phases: a QC is an implicit phase
    - Towards “zero-cost” consensus: minimal protocol state
- Linearity
    - First to achieve O(n) complexity (optimal)
    - Inspired other works
- “Pacemaker”
    - Decouples view-synchronization from the agreement (view-change)
    - Developer-friendly

Fundamental Problems Solved!

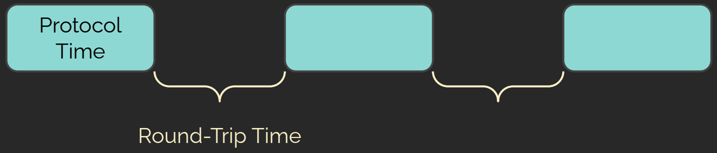
- Asynchronous Byzantine Agreement
  - VABA [2]
    - Runs n parallel HotStuff instances
    - first optimal solution: O(n^2)
- Optimistically Asynchronous BA
  - Bolt-Dumbo [26], Jolteon and Ditto[15]
    - Two-phase HotStuff as fast path: O(n)
    - O(n^2) asynchronous as fallback
- Partially Synchronous Consensus
  - HotStuff's linearity is for a single view-change
    - At most f view-changes: O(n^2)
    - Pacemaker's complexity?
  - Multiple failures: towards lowering worst-case complexity!

Fundamental Problems Solved!

- Pacemaker
  - Cogsworth [32]
    - Expected linearity in Pacemaker, worst-case O(n^3)
    - f+1 “backup” leaders
  - RareSync [10] and Lewis-Pye [25]
    - Worst-case O(n^2) with O(nΔ) latency
- Two-phase HotStuff
  - Fast HotStuff [18], DiemBFT-v4 [40], Jolteon and Ditto [15]
    - Two-phase “fast-path”
    - Reverts to PBFT-style O(n^2) per view-change
  - Wendy [16] and MSCFCL [3]
    - Similar, but focuses on compressing the leader proof
  - HotStuff-2 [28]: “well, the vanilla HotStuff is very close…”
    - A “bad day” could use Δ timed wait
    - But a node can tell if it is on a good day! (then optimistically, only need δ)

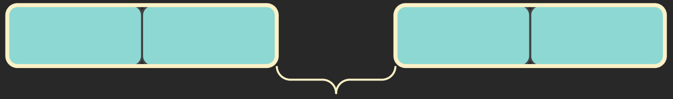
Scalability Lessons

- Computational & network resources
- Parallel Computation
  - Signature verification
  - Transaction dissemination (“mempool”)
- Large Blocks
  - Protocol’ = b * Protocol
  - Increase utilization, but not indefinitely…

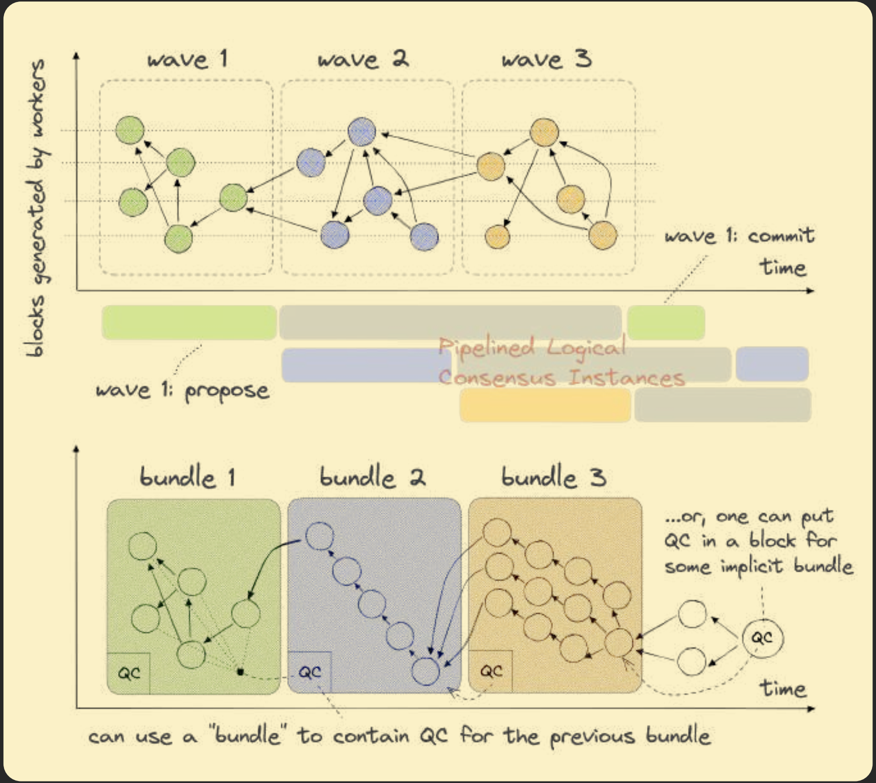


Protocol / (Protocol + Network)

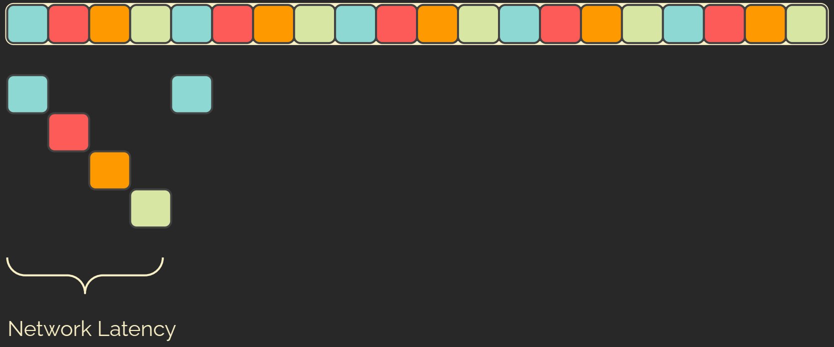
Protocol ~ microseconds
Network ~ dozens-hundreds of milliseconds



Scalability Lessons


wave 1
wave 2
wave 3
blocks generated by workers
wave 1: commit
time
Pipelined Logical Consensus Instances
wave 1: propose
bundle 1
bundle 2
bundle 3
...or, one can put QC in a block for some implicit bundle
QC
QC
QC
QC
time
can use a "bundle" to contain QC for the previous bundle

- Block Waves
  - Narwhal & Bullshark [12, 37], …
  - Idea: separate network propagation from the core loop
  - A Tn phase drives a “wave” of multiple instances of Tp
- Concurrent Instances
  - FairLedger [19], Mir-BFT [38], …
  - “Shard/Slice” the replicated log/chain into parallel instances
  - Challenge: fault-tolerance for the instance allocation



Check out our paper!

- Interesting theoretical & practical details
- Future research directions