BFT in Lens of Blockchain

BFT in Lens of Blockchain

Ted Yin[1,2], Dahlia Malkhi[2], Michael K. Reiter[2,3]
Guy Golan Gueta[2] and Ittai Abraham[2]
[1]Cornell University, [2]VMware Research, [3]UNC-Chapel Hill

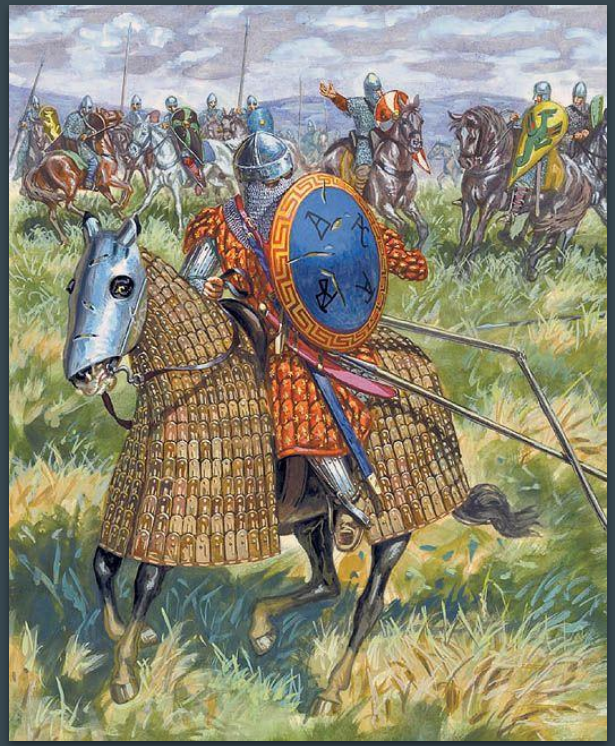
About Me

- {Design,prov,build}ing **practical** distributed systems with **fundamental** (algorithmic) improvements

- Major work:
  - Avalanche Consensus (permission-less, extremely scalable)
  - HotStuff Consensus (permission-ed, elegant and drop-in replacement for PBFT/PBFT-like use cases)

- Two other on-going projects (Cornell, VMware)

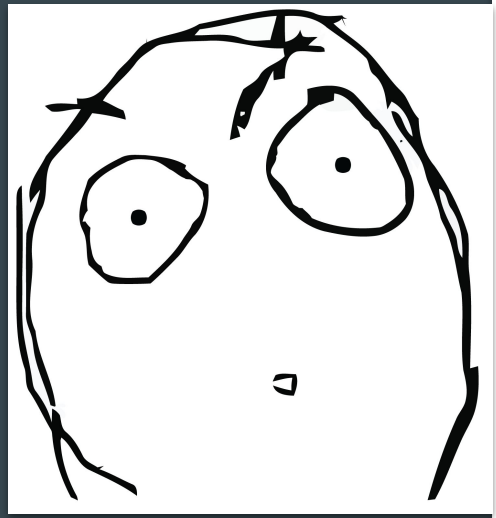
BFT Consensus: Research In Our Eyes

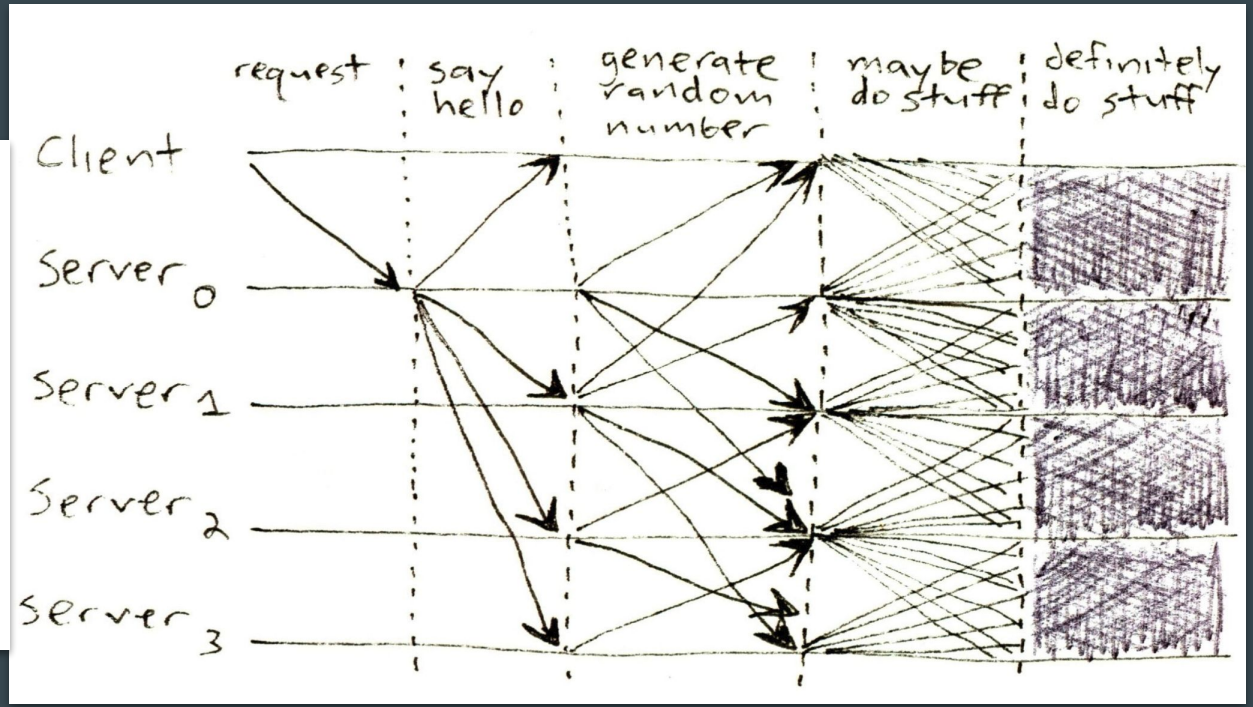


What we think we do

What others think we do

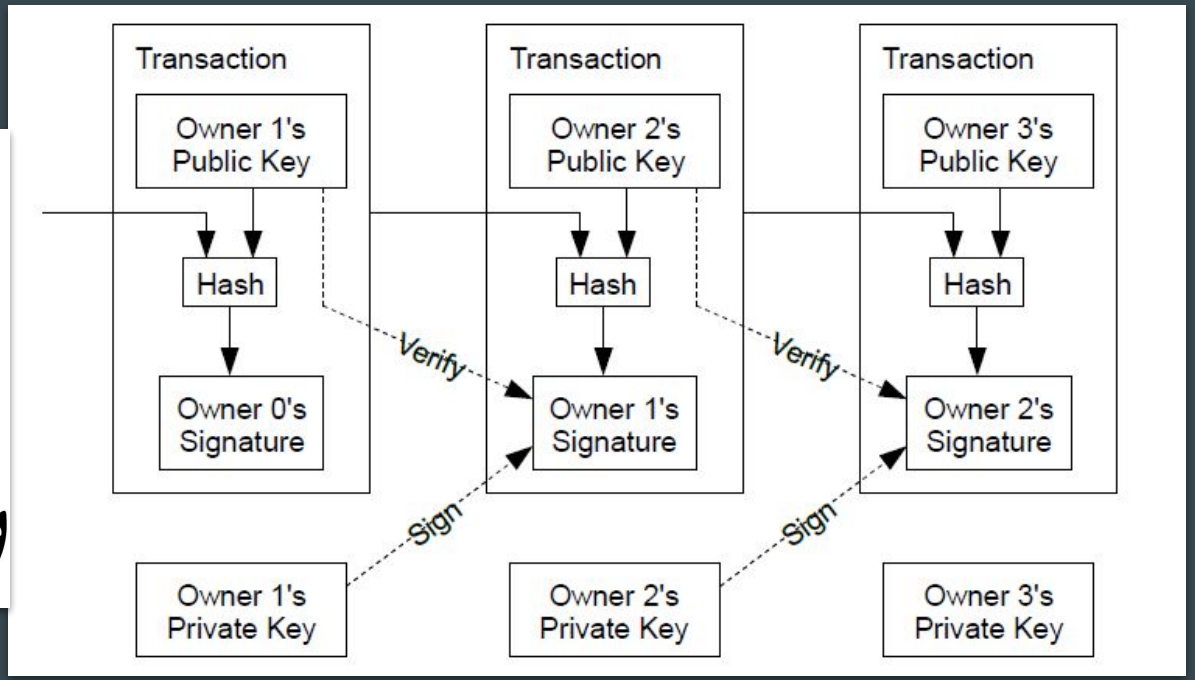
BFT Consensus: Why Another Protocol?





The Saddest Moment, Mickens 2013

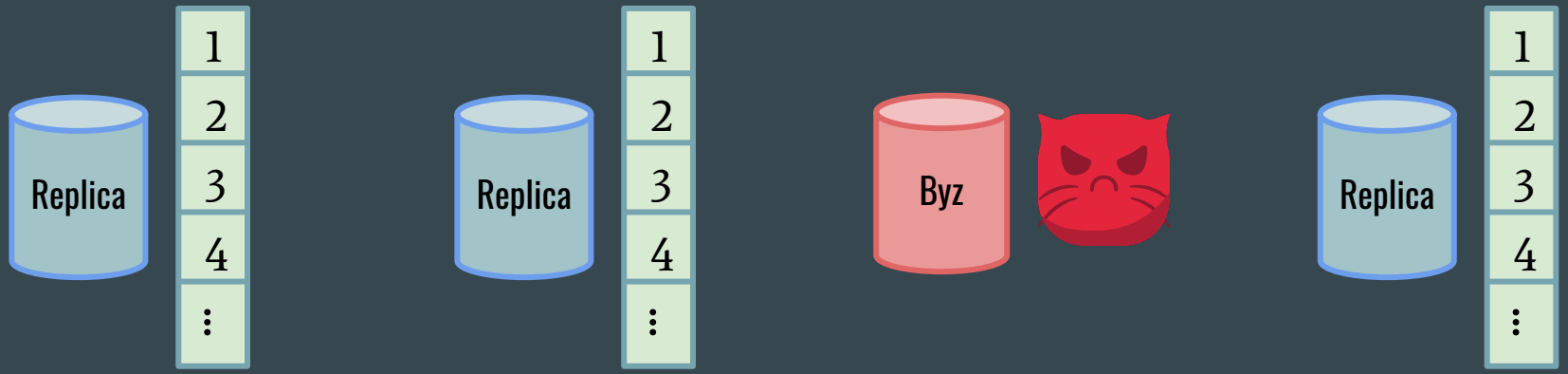
BFT Consensus: Why Another Protocol?



Bitcoin: a Peer-to-Peer Electronic Cash System, Nakamoto 2008

BFT Consensus: Problem Definition

- N nodes replicate the same sequence of commands
- Consistent in asynchronous network (safety)
- During period of synchrony, it'd better progress (liveness)
- When the proposer (leader) is correct, it should be fast



Reducing the Complexity

"Communication Complexity"

"Complexity"

"Protocol Complexity"

Possibly the first protocol with **linear cost** during a view change

Network Cost

Protocol Spec

Conferences probably don't care. But we do!

Protocol Complexity

Classical BFT

- **PoW-free**
- **Quorum invariants**
- From single-decree (1)
- (1) => Sequence numbers
- (1) => View numbers
- Hard to comprehend

Nakamoto's Consensus

- PoW based
- **Longest chain**
- **Naturally multi-decree** (2)
- **(2) => Block heights**
- **(2) => Views == Forks**
- **Easy to understand**

HotStuff: Protocol Framework & Simplicity

Framework

- Classical BFT variant (same/better guarantee)
- Bridges classical BFT and blockchain
- View change is everywhere, and nowhere
- Locking mechanism (reducing protocol state space)
- Decouples safety and liveness
- “Liveness gadget” could be RR, PoW based, etc.

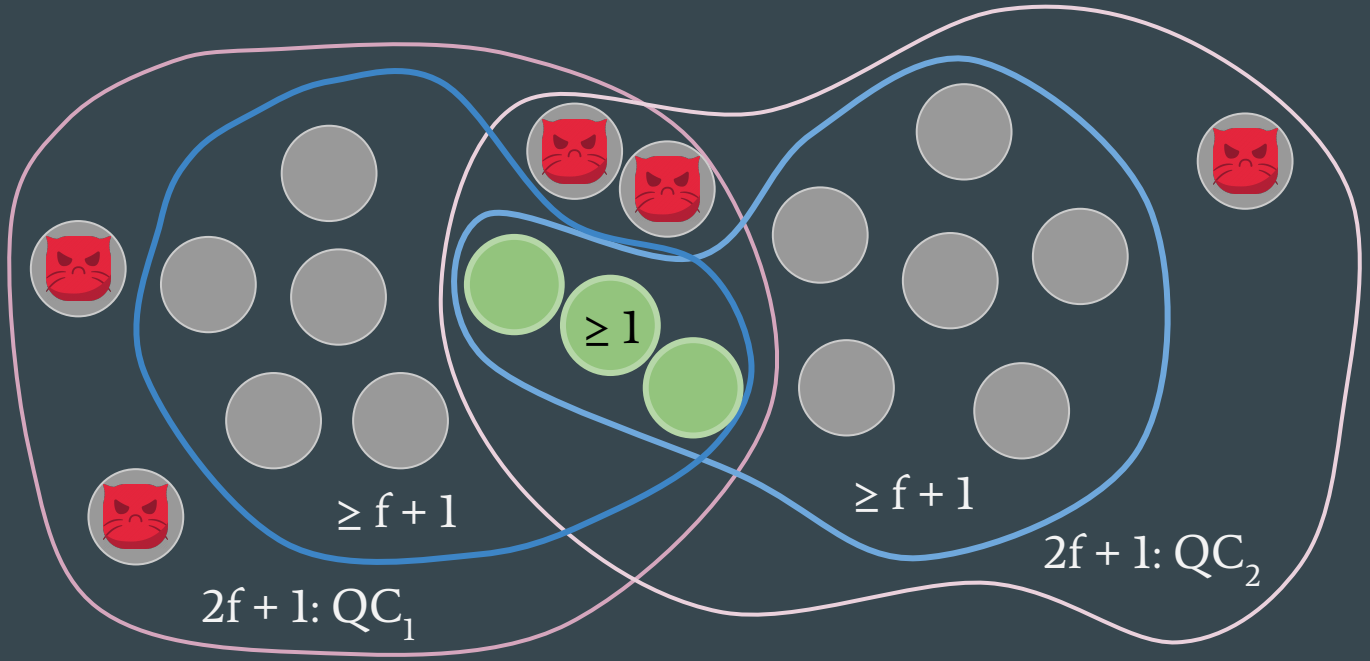
Challenge: BFT consensus in 10 min

Ingredients to Make a 2-step HotStuff

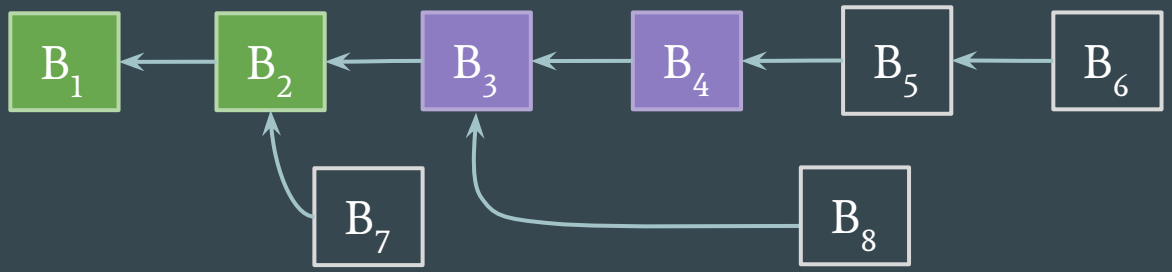
- Protocol state variables
- Message types
- Voting rule
- Commit rule

Quorum Certificate

QC: Proof of the Existence of 2f+1 (positive) Votes



Blockchain!

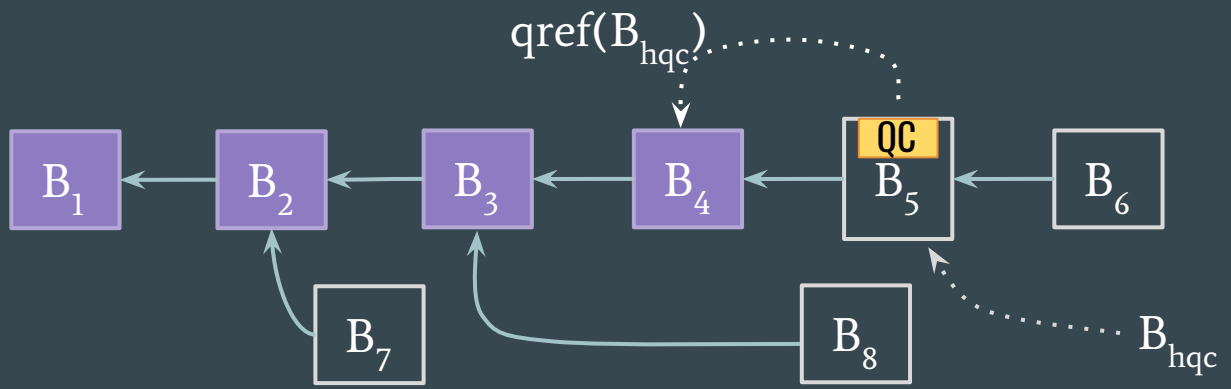


Branch Preference

B_{hqc}: block containing QC for the preferred block

"Preferred block" or qref(B_{hqc}): highest block receives a QC

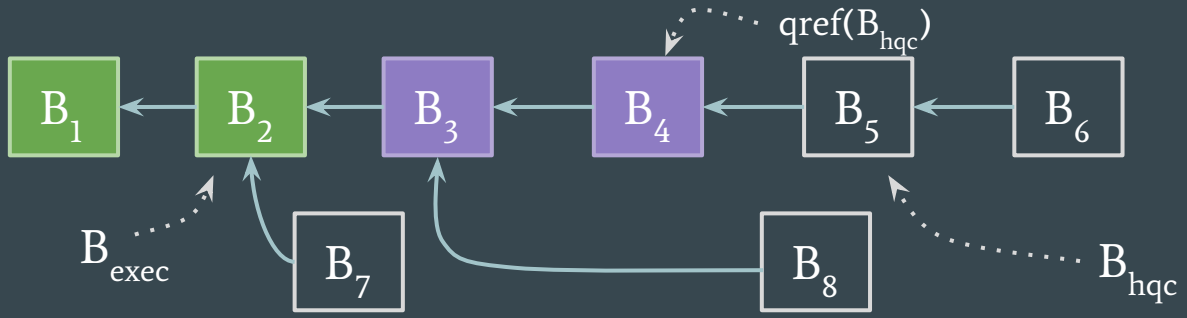
Locking mechanism: a replica sticks to qref(B_{hqc}) unless...



Challenge: BFT consensus in 10 min

Protocol State Variables

- B_{hqc} = block containing a reference to the preferred branch
- B_{exec} = last committed block
- vheight = height of the block last voted for



Challenge: BFT consensus in 10 min

Message Types

<propose, v, B_{new}, $B_{hqc'}$>

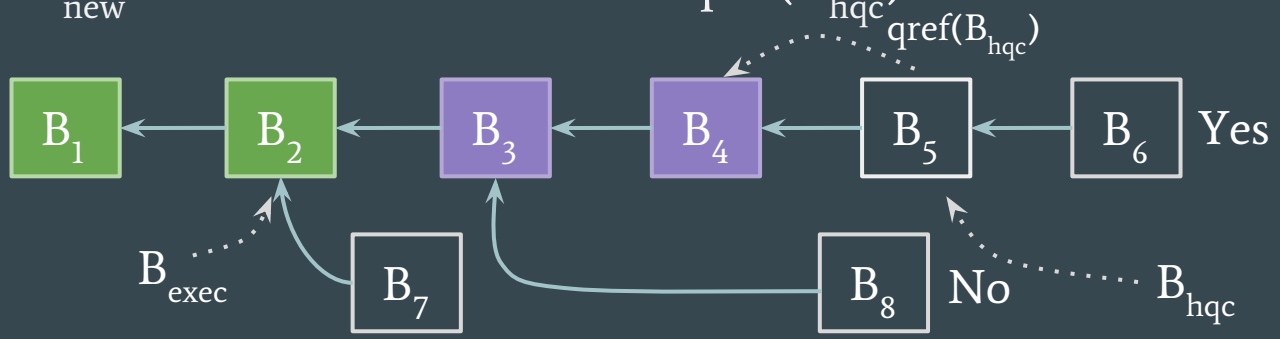
<vote, <v, B_{new}>$_{signed\ by\ v}$, $B_{hqc'}$>

- Proposer broadcasts the propose message for block B_{new}
- Voters give back their opinions to the next proposer via votes
- **Only one type of messages for voting/view change, etc.**

Challenge: BFT consensus in 10 min

How to Vote?

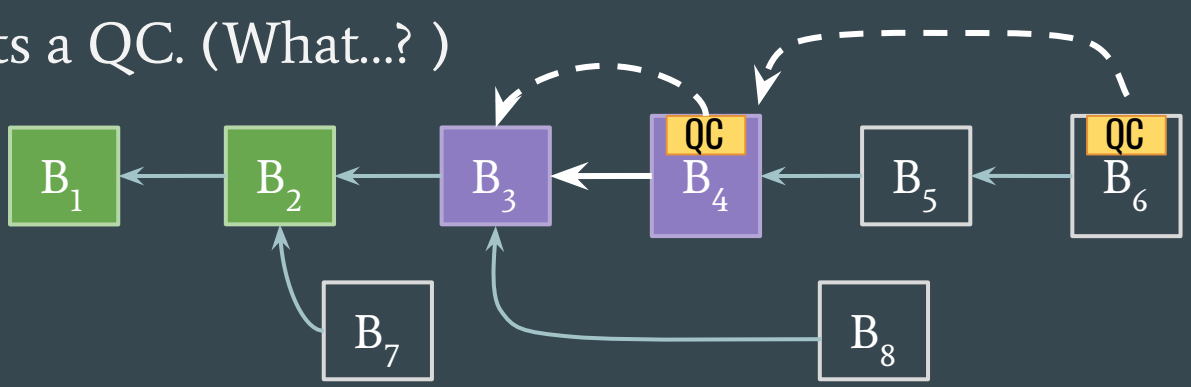
- Only vote positively for B_{new} if the following constraints hold:
  - B_{new}.height > vheight
  - B_{new} is on the same branch as qref(B_{hqc})



Challenge: BFT consensus in 10 min

When to Commit?

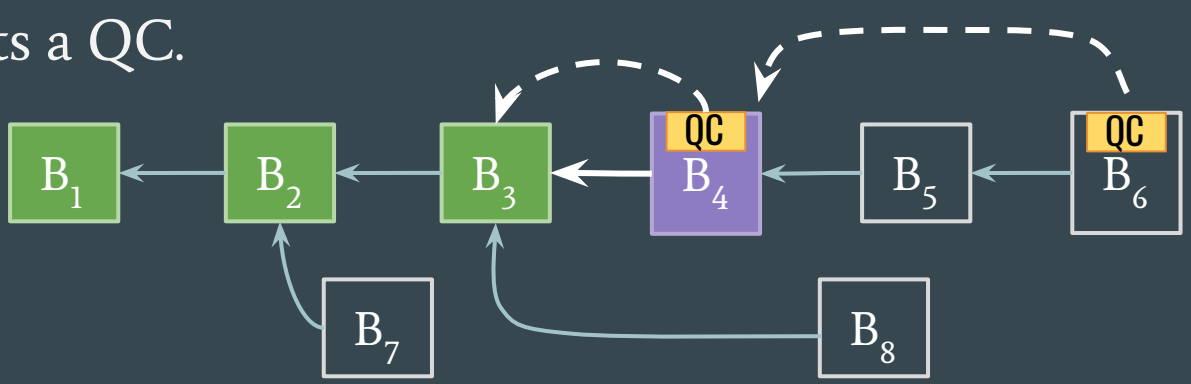
- Every block could contain a QC for some previous block
- Block B will be committed when a child having QC for B also gets a QC. (What...?)



Challenge: BFT consensus in 10 min

When to Commit?

- Every block could contain a QC for some previous block
- Block B will be committed when a child having QC for B also gets a QC.



HotStuff: Protocol in a Single Slide (2-step version)

Pseudo-code for replica u

```
1:  // begin: rules specific to 2-step HotStuff in framework
2:  function getPref() := qref(B_hqc)
3:  function checkCommit
4:      // check for a Commit 2-chain
5:      B' := qref(B_hqc)
6:      B := qref(B')
7:      if B = B'.parent then
8:          onCommit(B); return true
9:      else return false
10: // end
11: // begin: generic HotStuff framework logic
12: procedure finishQC(B)
13:     certs[B] := makeQC(ref = B, votes[B])
14: procedure onCommit(B)
15:     if B_exec.height < B.height then
16:         onCommit(B.parent)
17:         execute(B.cmd)
18: procedure update(B'_hqc)
19:     if qref(B'_hqc).height > qref(B_hqc).height then
20:         B_hqc := B'_hqc
21:     if checkCommit then B_exec := B
22: procedure onReceiveProposal(⟨propose, v, B_new, B'_hqc⟩)
23:     update(B'_hqc)
24:     if B_new.height > vheight ∧ getPref() ←* B_new then
25:         vheight := B_new.height
26:         vote := ⟨vote, ⟨u, B_new⟩_σu, B_hqc⟩
27:         send(nextProposer(v), vote)
28: procedure onReceiveVote(⟨vote, ⟨v, B_new⟩_σv, B'_hqc⟩)
29:     update(B'_hqc)
30:     if ∃⟨v, B_new⟩_σ'v ∈ votes[B_new] then return
31:     // collect votes for B_new
32:     votes[B_new] := votes[B_new] ∪ {⟨v, B_new⟩_σv}
33:     if |votes[B_new]| ≥ 2f + 1 then finishQC(B_new)
34: procedure onPropose(B_tail, qc, cmd)
        B_new := makeBlock(parent = B_tail,
35:                        height = B_tail.height + 1,
                           qc = qc, cmd = cmd)
36:     // send to all replicas, including u itself
37:     broadcast(⟨propose, u, B_new, B_hqc⟩)
38: // end
```

HotStuff: Protocol in a Single Slide (3-step version)

```
1:  // begin: rules specific to 3-step HotStuff in framework
2:  function GETPREF() := QREF(QREF(B_hqc))
3:  function CHECKCOMMIT
4:      // check for a Commit 3-chain
5:      B'' := QREF(B_hqc)
6:      B' := QREF(B'')
7:      B := QREF(B')
8:      if (B = B'.parent) ∧ (B' = B''.parent) then
9:          ONCOMMIT(B); return true
10:     else return false
11: // end
```

HotStuff vs. State of the Art Performance

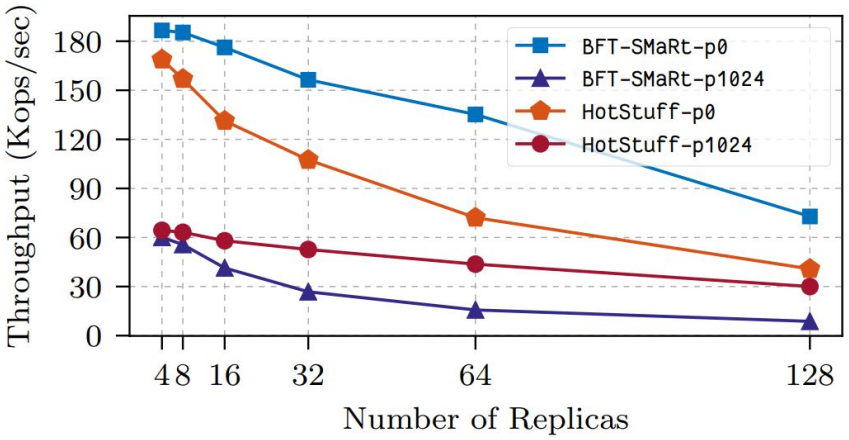


Figure 9: Throughput vs. number of nodes with payload size 0/0 and 1024/1024.

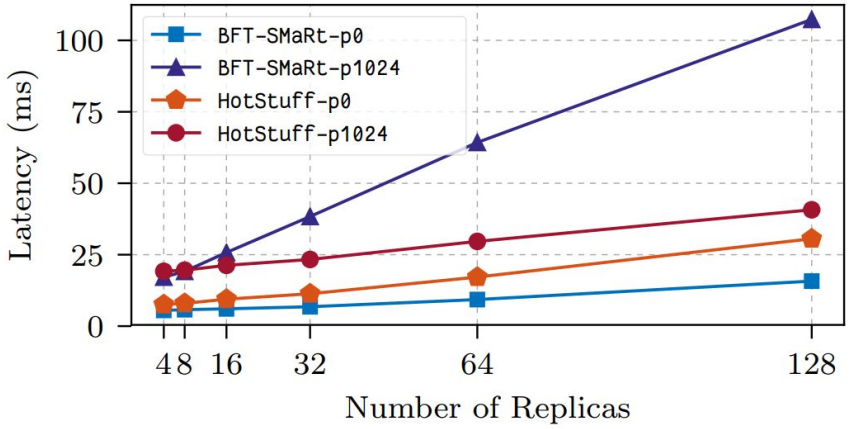


Figure 10: Latency vs. number of nodes with payload size 0/0 and 1024/1024.

HotStuff vs. State of the Art Performance

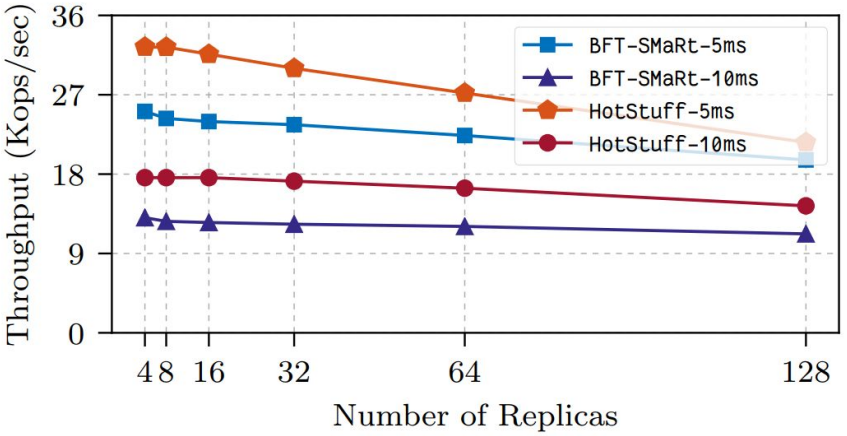


Figure 11: Throughput vs. number of nodes with inter-replica latency 5ms and 10ms.

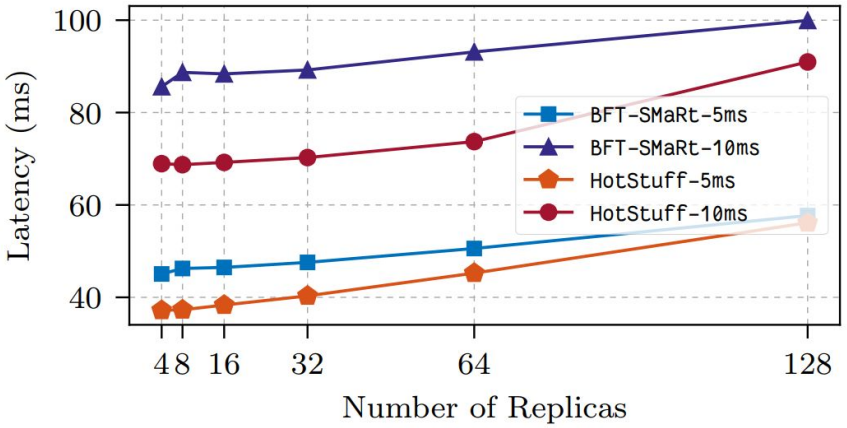
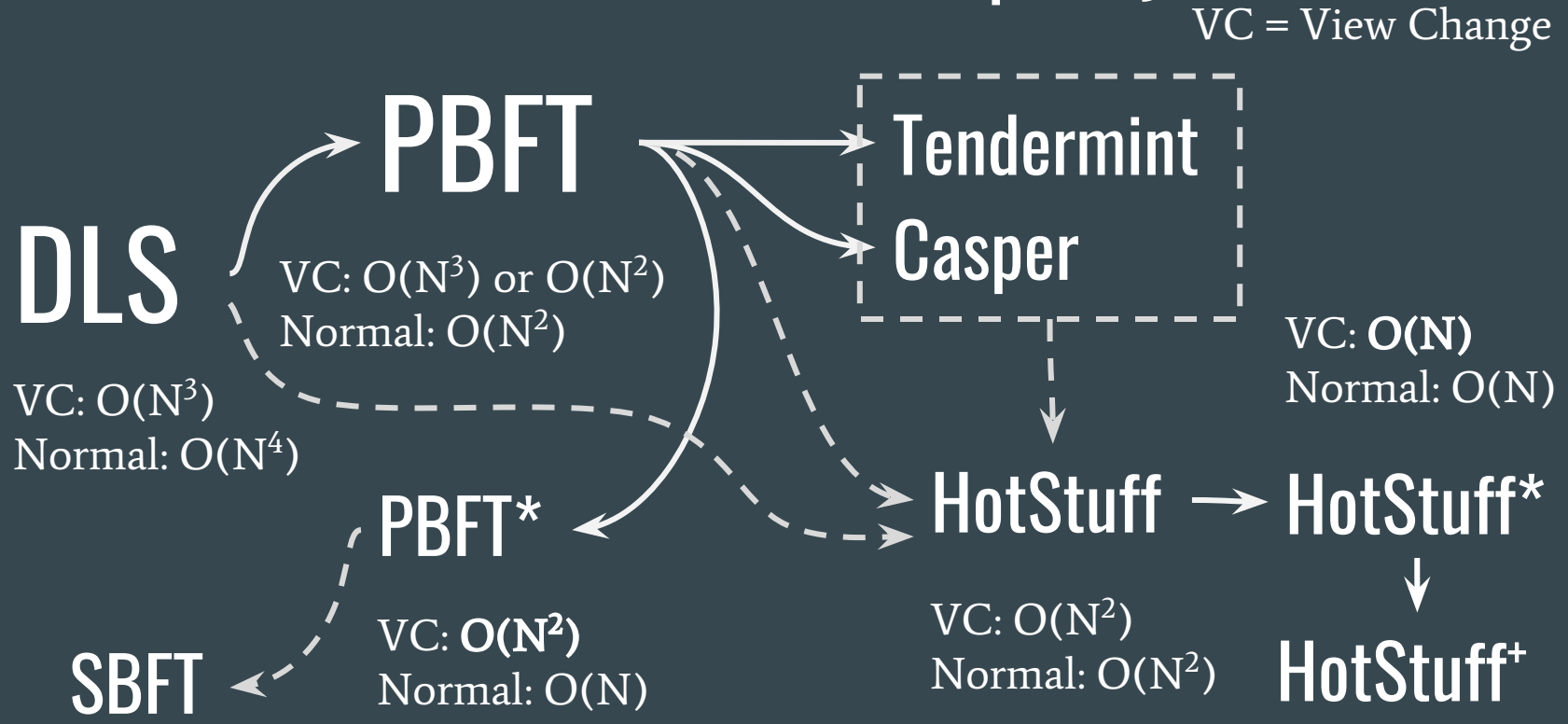


Figure 12: Latency vs. number of nodes with inter-replica latency 5ms $\pm$ 0.5ms or 10ms $\pm$ 1.0ms.

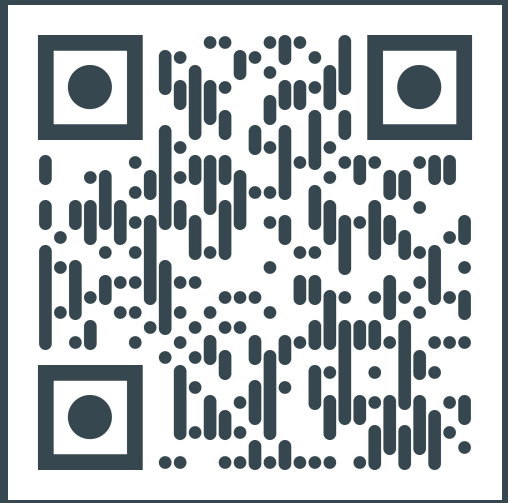
BFT Solutions & Communication Complexity



That’d be all.

Special Thanks

VMware Research Group



https://arxiv.org/abs/1803.05069

Open-sourced code coming soon