BFT in Lens of Blockchain

Ted Yin\textsuperscript{1,2}, Dahlia Malkhi\textsuperscript{2}, Michael K. Reiter\textsuperscript{2,3}
Guy Golan Gueta\textsuperscript{2} and Ittai Abraham\textsuperscript{2}
\textsuperscript{1}Cornell University, \textsuperscript{2}VMware Research, \textsuperscript{3}UNC-Chapel Hill
About Me

- Designing, prov, building **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?

Transaction
Owner 1's Public Key
Hash
Owner 0's Signature
Owner 1's Private Key

Transaction
Owner 2's Public Key
Hash
Owner 1's Signature
Owner 2's Private Key

Transaction
Owner 3's Public Key
Hash
Owner 2's Signature
Owner 3's Private Key

BFT Consensus: Problem Definition

- N nodes replicate the same sequence of commands
- Consistent in asynchronous network (safety)
- During period of synchrony, it’s 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

<table>
<thead>
<tr>
<th>Classical BFT</th>
<th>Nakamoto’s Consensus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PoW-free</strong></td>
<td><strong>PoW based</strong></td>
</tr>
<tr>
<td><strong>Quorum invariants</strong></td>
<td><strong>Longest chain</strong></td>
</tr>
<tr>
<td>From single-decree (1)</td>
<td>Naturally multi-decree (2)</td>
</tr>
<tr>
<td>(1) =&gt; Sequence numbers</td>
<td>(2) =&gt; Block heights</td>
</tr>
<tr>
<td>(1) =&gt; View numbers</td>
<td>(2) =&gt; Views == Forks</td>
</tr>
<tr>
<td>Hard to comprehend</td>
<td>Easy to understand</td>
</tr>
</tbody>
</table>
## HotStuff: Protocol Framework & Simplicity

<table>
<thead>
<tr>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Classical BFT variant (same/better guarantee)</td>
</tr>
<tr>
<td>● Bridges classical BFT and blockchain</td>
</tr>
<tr>
<td>● View change is everywhere, and nowhere</td>
</tr>
<tr>
<td>● Locking mechanism (reducing protocol state space)</td>
</tr>
<tr>
<td>● Decouples safety and liveness</td>
</tr>
<tr>
<td>● “Liveness gadget” could be RR, PoW based, etc.</td>
</tr>
</tbody>
</table>
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

$2f + 1$: QC

$\geq f + 1$

$\geq 1$

$2f + 1$: QC

$\geq f + 1$

$\geq f + 1$

$2f + 1$: QC

$\geq f + 1$
Blockchain!
Branch Preference

$B_{hq_c}$: block containing QC for the preferred block

“Preferred block” or qref($B_{hq_c}$): highest block receives a QC

Locking mechanism: a replica sticks to qref($B_{hq_c}$) 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

- \(<\text{propose, v, } B_{\text{new}}, B_{\text{hqc}}\)>
- \(<\text{vote, } \langle v, B_{\text{new}} \rangle\text{ signed by } v, B_{\text{hqc}}\)>

- Proposer broadcasts the propose message for block $B_{\text{new}}$
- 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 $B_{\text{new}}$ if the following constraints hold:
  - $B_{\text{new}}$.height > vheight
  - $B_{\text{new}}$ is on the same branch as $\text{qref}(B_{\text{hqc}})$
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 $u$

```plaintext
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($B$)
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, $B_{new}$)$_{σ_u}$, $B_{hqc}$)
27:          send(nextProposer(v), vote)
28:  procedure onReceiveVote(($vote, (u, $B_{new}$)$_{σ_v}$, $B'_{hqc}$))
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, $B_{new}$)$_{σ_v}$\}
33:      if |votes[$B_{new}$]| $≥$ 2$f$ + 1 then finishQC($B_{new}$)
34:  procedure onPropose($B_{tail}$, qc, cmd)
35:      $B_{new} := makeBlock$(parent = $B_{tail}$, height = $B_{tail}.height + 1$
36:      qc = qc, cmd = cmd)
37:      // send to all replicas, including $u$ itself
38:      broadcast(($propose, u, B_{new}, B_{hqc}$))
39:  // 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 ± 0.5ms or 10ms ± 1.0ms.
BFT Solutions & Communication Complexity

PBFT

VC: $O(N^3)$ or $O(N^2)$
Normal: $O(N^2)$

PBFT*

VC: $O(N^2)$
Normal: $O(N)$

DLS

VC: $O(N^3)$
Normal: $O(N^4)$

SBFT

VC: $O(N^2)$
Normal: $O(N)$

Tendermint

VC: $O(N)$
Normal: $O(N)$

Casper

VC: $O(N^2)$
Normal: $O(N^2)$

HotStuff

VC: $O(N^2)$
Normal: $O(N^2)$

HotStuff*

VC: $O(N)$
Normal: $O(N)$

HotStuff+

VC: $O(N^2)$
Normal: $O(N^2)$

VC = View Change
That’d be all.
Special Thanks
VMware Research Group

https://arxiv.org/abs/1803.05069
Open-sourced code coming soon