The Honey Badger of BFT Protocols

Authors: Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, Dawn Song

Minqiang Hu
10/29/2018
Timing assumptions considered harmful

- Weak Synchrony and Asynchronous BFT Protocol
- Why not weak synchrony
  - The liveness will fail when expected time assumptions are violated
  - Less throughput when network is unpredictable
- HoneyBadgerBFT guarantees liveness without making time assumptions
Asynchronous networks are the “harsh climates” of distributed computing

Full Synchrony: all messages are delivered within $\Delta$ time

Eventual Synchrony: after unknown time $GST$, all messages delivered within $\Delta$

Partial Synchrony: $\Delta$ is unknown to the protocol

Weak Synchrony: $\Delta t$ is time varying, but grows polynomially in $t$

Asynchronous: all messages are eventually delivered
Fault models

Timing assumptions

Crash faults

Fully-Synchronous

Paxos

Weakly-synchronous

Raft

Asynchronous

Byzantine faults

Dolev-Strong

PBFT

SINTRA
Cachin and Portiz, 2002
The Approach

Adapt synchronous BFT for efficiency in the batch setting

1. Improve by $O(N)$ by “refactoring” with known (but overlooked) primitives
2. Improve by another $O(N)$ by using random selection and threshold encryption
Refactor of the transactions to mitigate node bandwidth bottleneck

PBFT broadcast standard way

P: leader

Leader bandwidth $O(n^3)$
Refactor of the transactions to mitigate node bandwidth bottleneck

Improved broadcast way

Leader bandwidth $O(n)$
Avoid sending redundant transactions—random selection and threshold encryption
Results: Optimal resilience and efficiency

Choose a large enough batch size, of $B = \Omega(\lambda N^2 \log N)$.

Total Bandwidth per transaction (for each node) is $O(1)$. Expected # of rounds is $O(\log N)$. 
Implementation

Python protocol implementation, using gevent

Threshold cryptography: Charm/PBC library

  Signature: Boldyreva ‘03

  Encryptions: Baek and Zhang ‘03

Experiments on local cluster, worldwide EC2, & over Tor
Thank you!