Dumbo Protocol Family
Making Asynchronous Consensus Real

Qiang Tang
The University of Sydney

THE UNIVERSITY OF
SYDNEY

I Work on Crypto

Cryptocurrency is your chance to get rich

Crypto means cryptography!

Either way is OK for me

Blockchain Technology

Many “mysterious” characterizations

You can read

You can write

You can ask me to execute

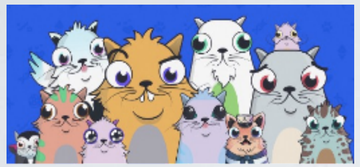
It is secure

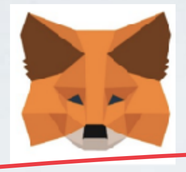
It is robust

It is unalterable

Our Work on Blockchain

CCS15,ICDCS18,20, ESORICS 20,21,…







…

decentralized applications

CCS20, PODC20,NDSS22,CCS22(a),CCS22(b),ICDCS22…

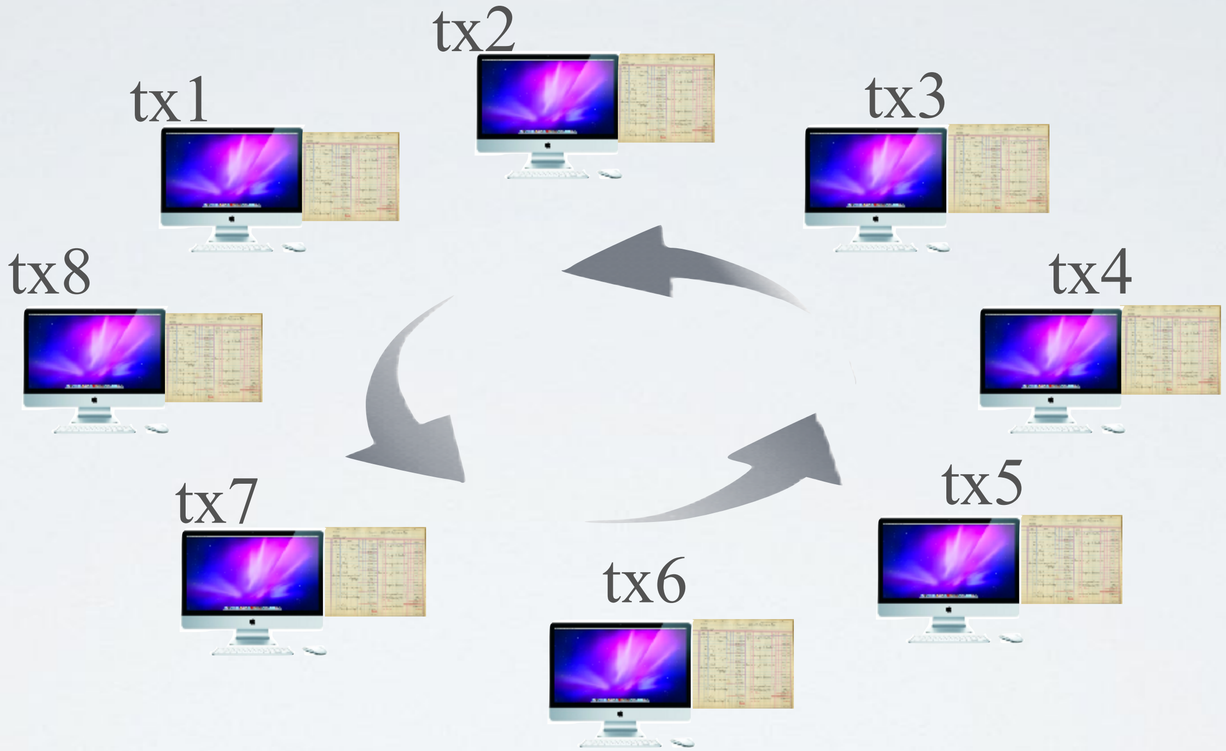

consensus layer

EUROCRYPT16,CRYPTO18,21,NDSS20,AC 20,21, USENIX Sec22…


…

cryptographic supports

The Consensus/Atomic Broadcast Problem



Safety (total order + agreement): all honest ledgers are the "same"

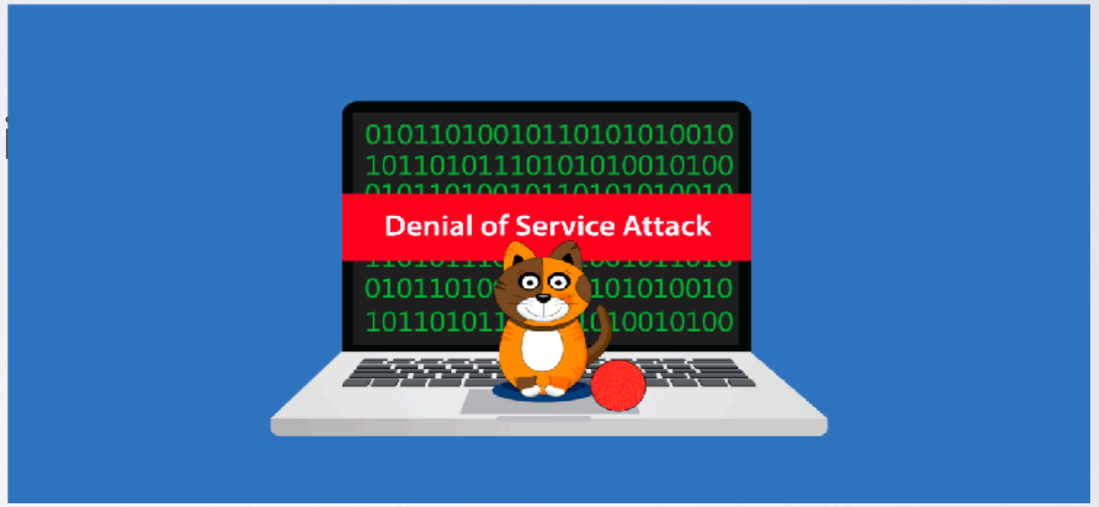
Liveness: every honest transaction will be recorded

Most Natural Challenges

Timing/Network Assumptions

Synchronous: all messages delivered within time d

es deli



Asynchronous: all messages will be eventually delivered

Further Benefits of Asynchronous Protocols

- Without time bound, the protocol will proceed as network delivers — responsiveness
- No manual timeout, it saves engineers' lives

Most of existing platforms make timing assumptions

Asynchronous Permissioned Consensus

FLP Impossibility: no deterministic protocol exists for asynchronous consensus

Mostly thereotical

Can it be practical?

Yes, we can
Need some Efforts

The Evolving of Dumbo Protocols

Cachin et al
MVBA
Crypto' 01

const running time

HoneyBadger
CCS '16

optimal comm / tx

Dumbo
"Classic"
(CCS '20)

optimal comm / tx
const running time

Dumbo
"MVBA"
(PODC '20)

asymp optimal everything

Speeding
Dumbo
(NDSS '22)

optimal comm / tx
const running time
optimal msg
small # of rounds

Dumbo-NG
(CCS'22)

optimal comm / tx
const running time
optimal msg
small # of rounds
~ bandwidth limit
tps-oblivious latency

Bolt-Dumbo-
Transformer
(CCS '22)

optimal comm / tx
const running time
optimal msg
small # of rounds
~ bandwidth limit
as fast as HotStuff

Asynchronous Permissioned Consensus

Major insights from HBBFT

Asynchronous Common Subset is good for batching, to get linear per tx comm

ACS can be built from binary Byzantine Agreement

Decoupling HB-BFT

RBC: if one honest node receives a tx, all honest nodes will receive tx

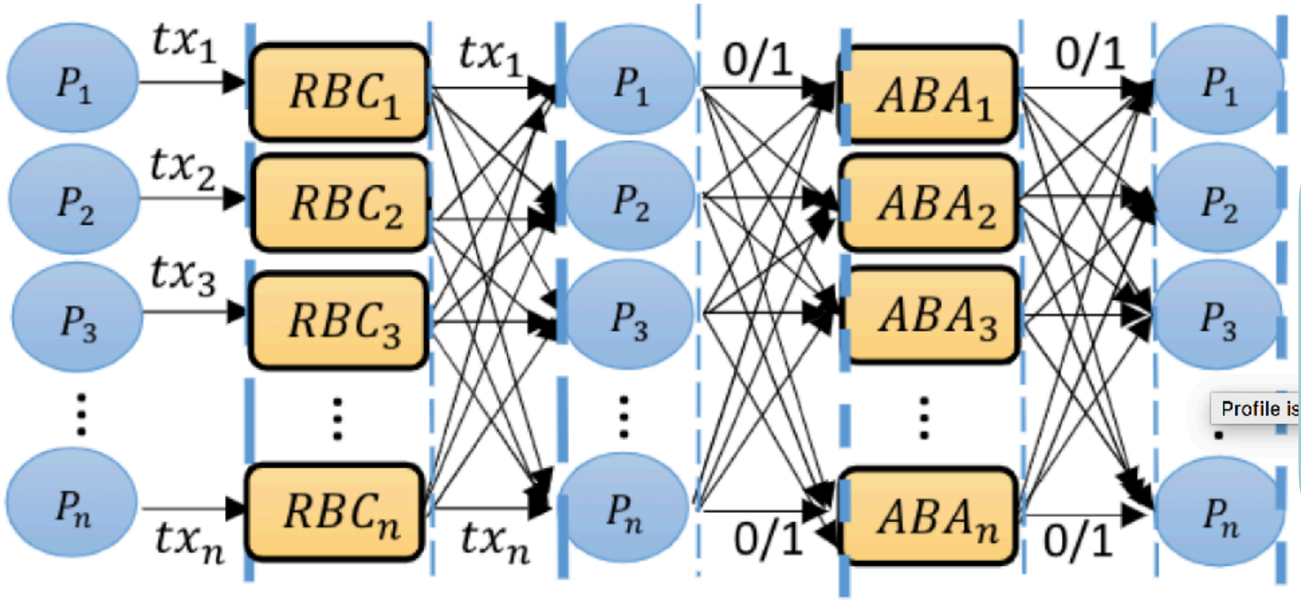


Figure 1: The structure of ACS in HoneyBadgerBFT

ABA: if honest nodes output 1, at least one honest node inputs 1

Identifying the Bottleneck

FLP I

ensus

ons of

hms

stance

w one

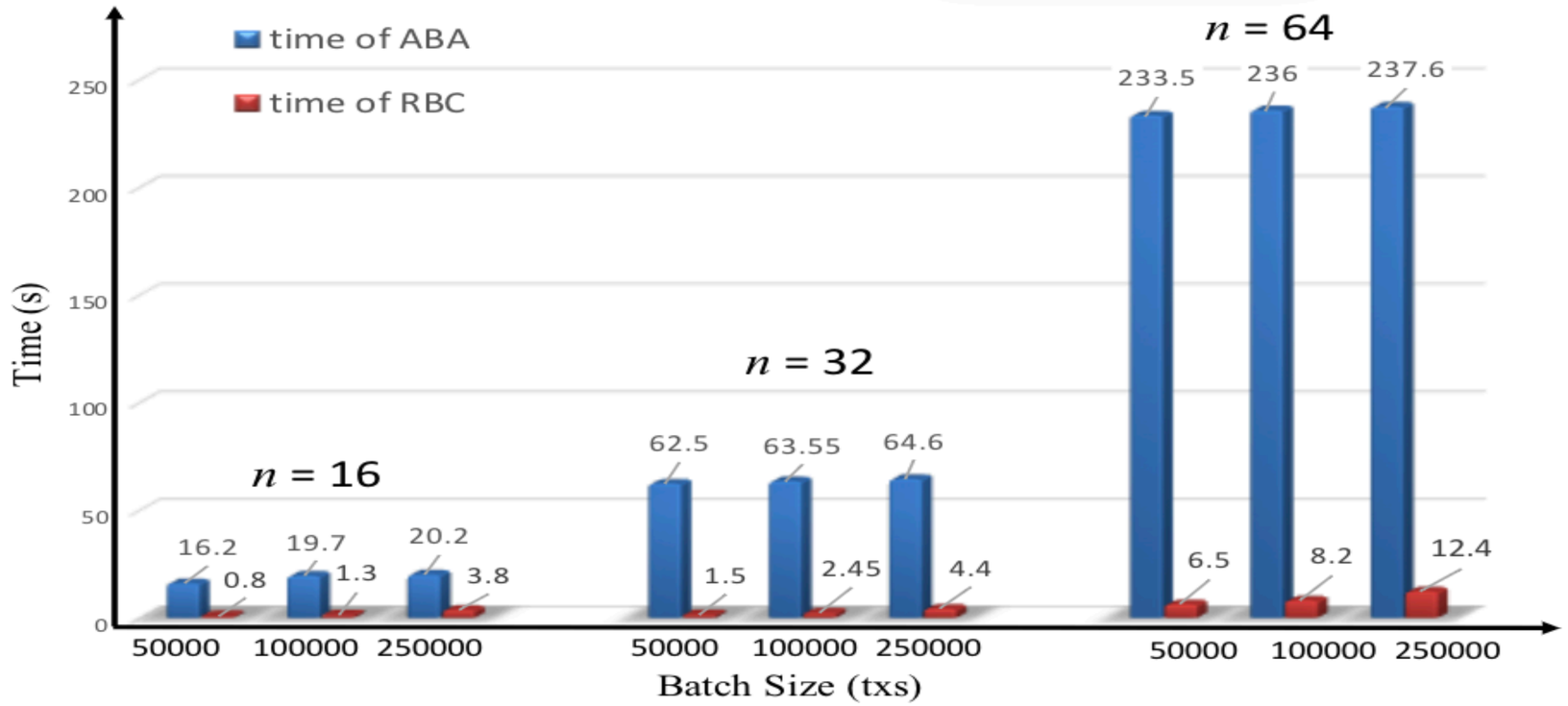


Figure 2: Time costs of RBC and ABA in HoneyBadgerBFT

Dumbo2: Pushing ABA to Minimum

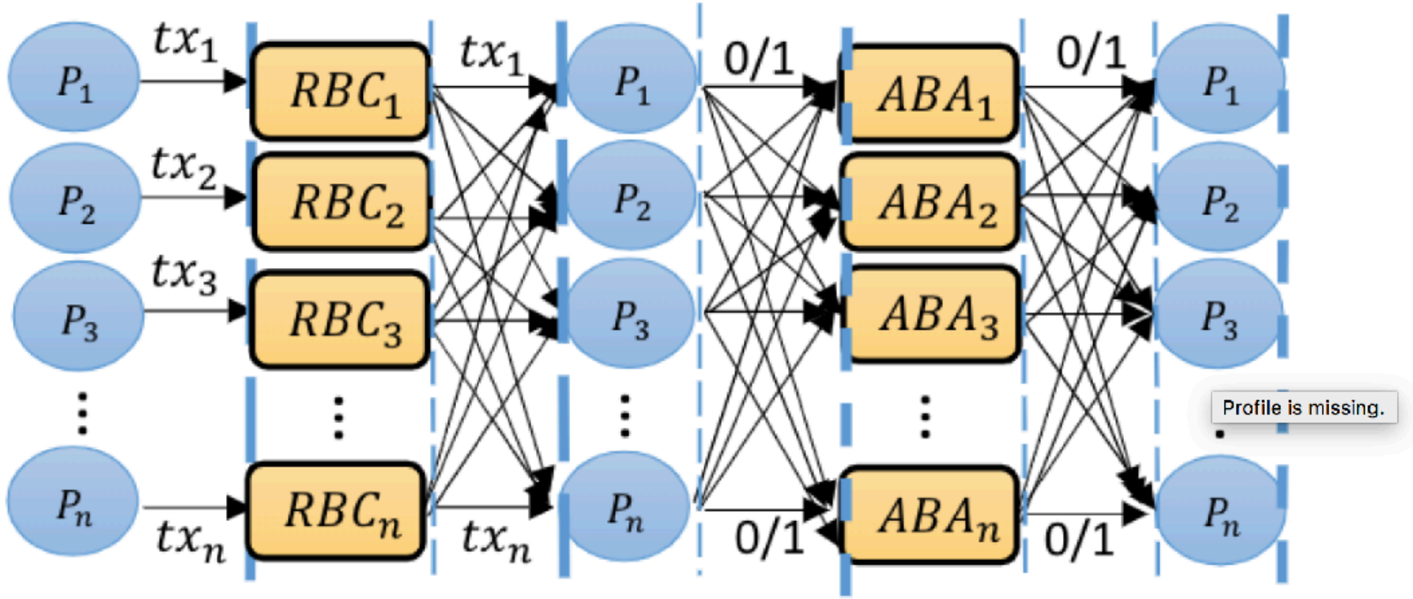


Figure 1: The structure of ACS in HoneyBadgerBFT

Dumbo2: Pushing ABA to Minimum

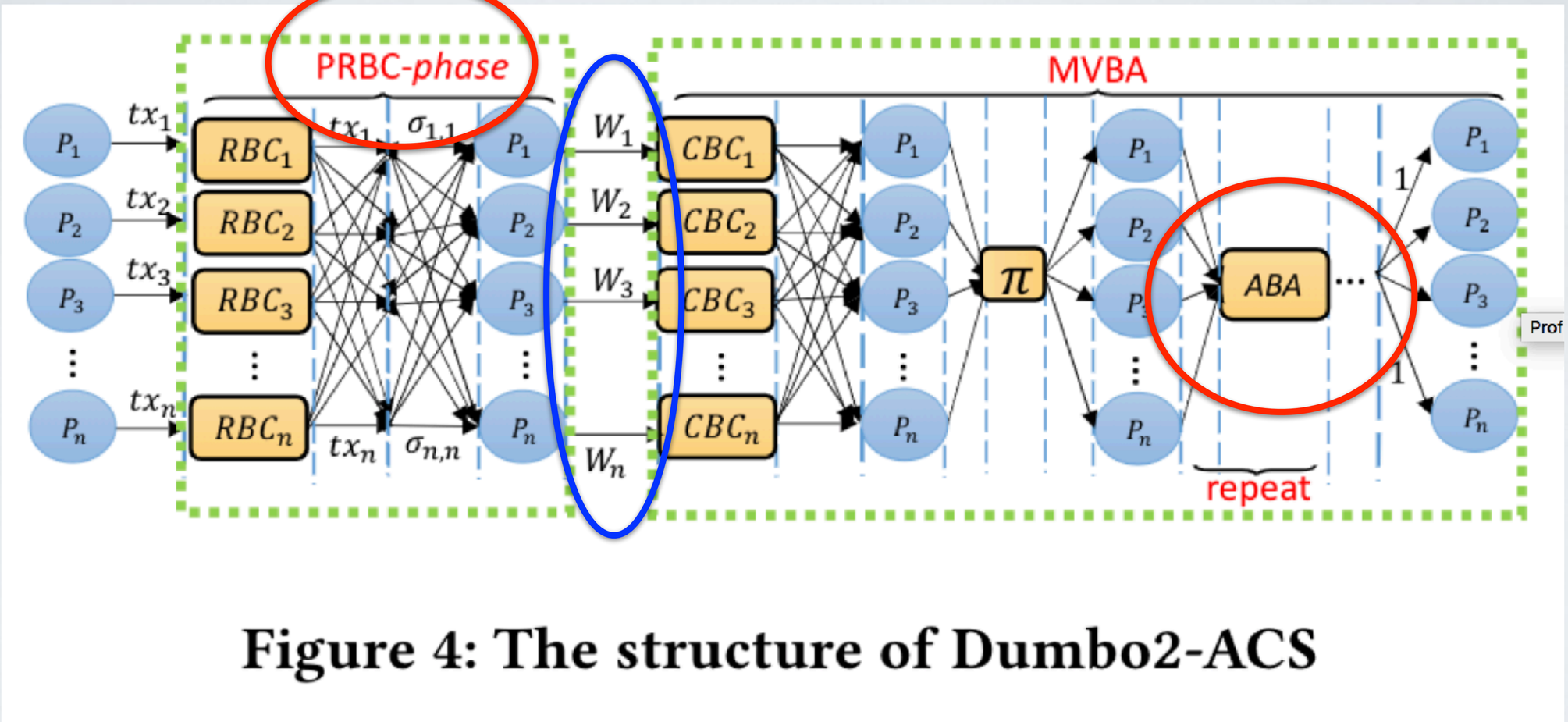
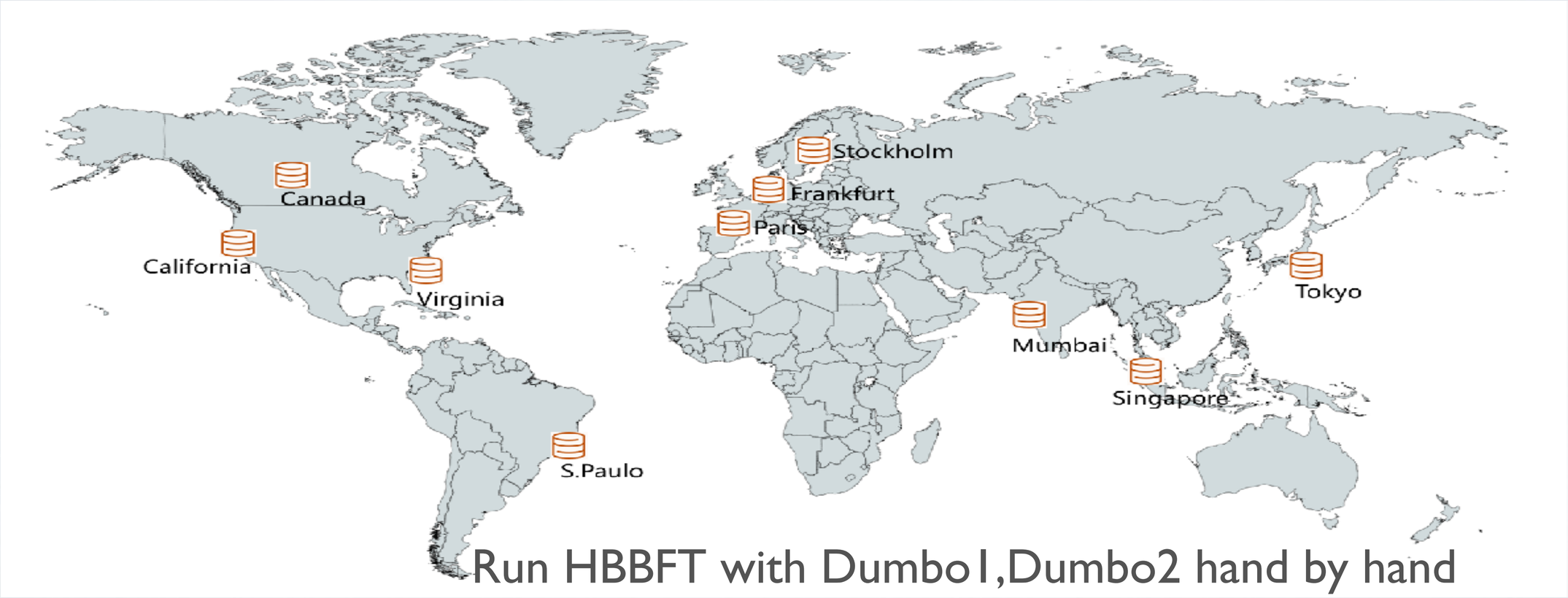


Figure 4: The structure of Dumbo2-ACS

MVBA could blow up communication [HBBFT]

Experiments

100 AWS EC2 t2.medium instances uniformly spread in 10 regions world-wide



Run HBBFT with Dumbo1,Dumbo2 hand by hand

Performance

System Scal	nbo2
n=32	↑ 79%
n=64	↑ 320%
n=100	↑ 819%

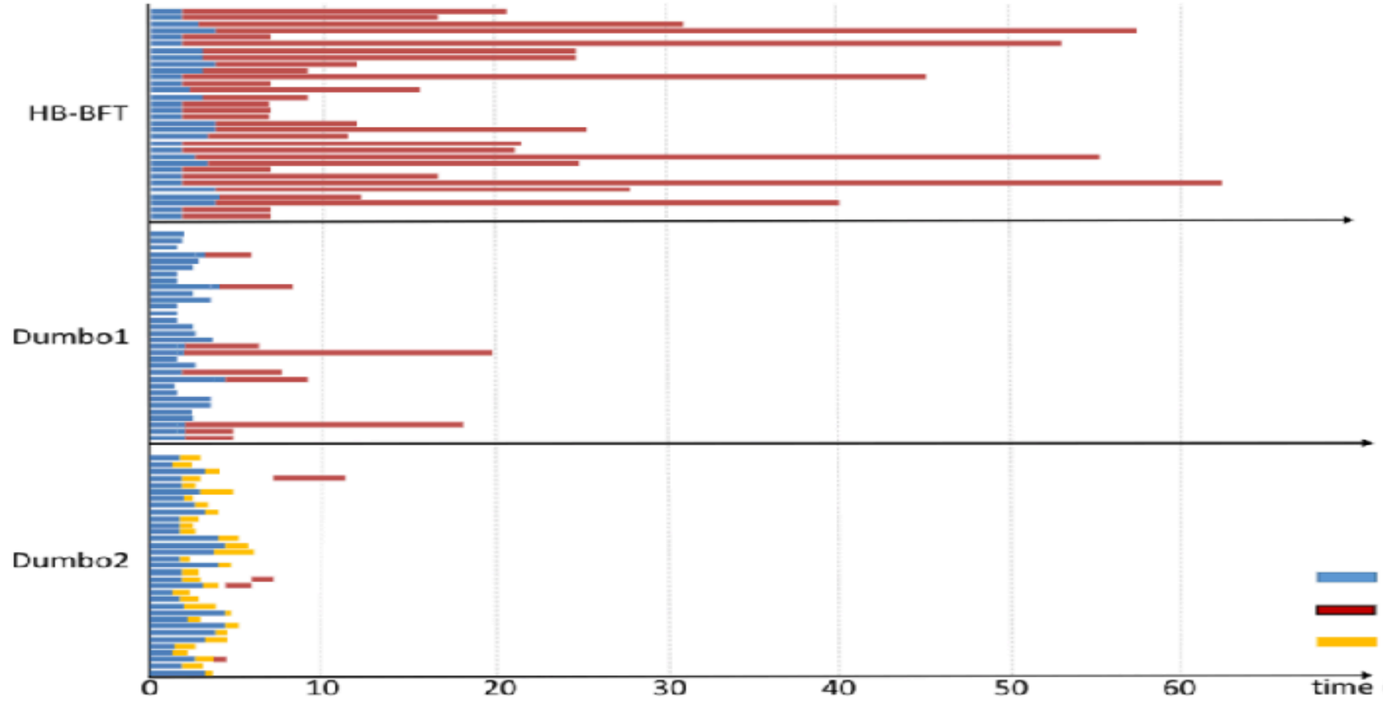


Figure 5: Running time breakdown of Dumbo1/2 and HoneyBadgerBFT on one random node.

Performance was achieved without any system optimization

Paths for Practical Asynchronous Atomic Broadcast/Consensus

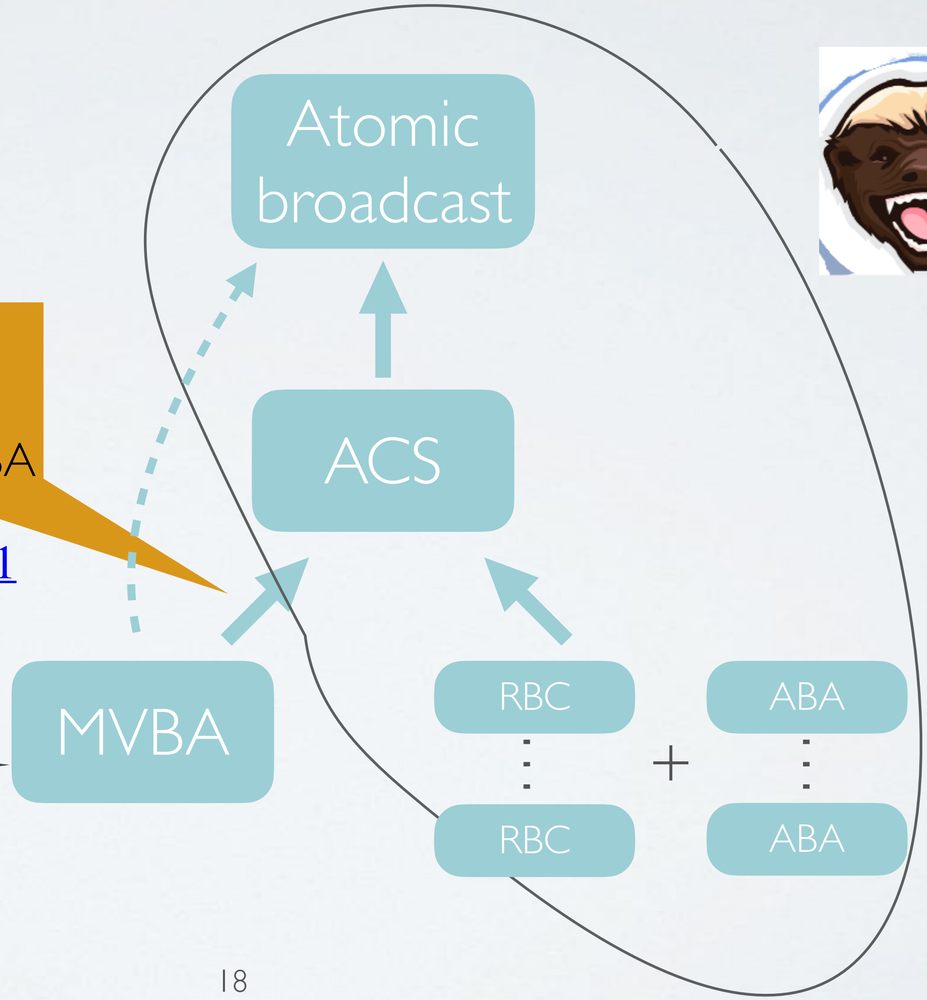
Reclaim the glory of MVBA for async consensus

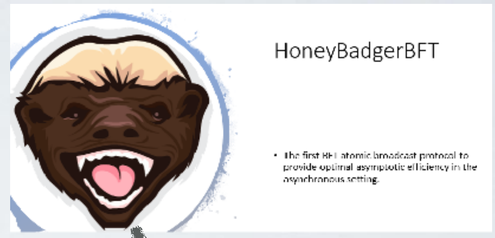
Dumbo2
CCS'20
RBC+MVBA

https://eprint.iacr.org/2020/841

Dumbo-MVBA
PODC'20

https://eprint.iacr.org/2020/842





To avoid heavy MVBA w. large input

Are we Done?

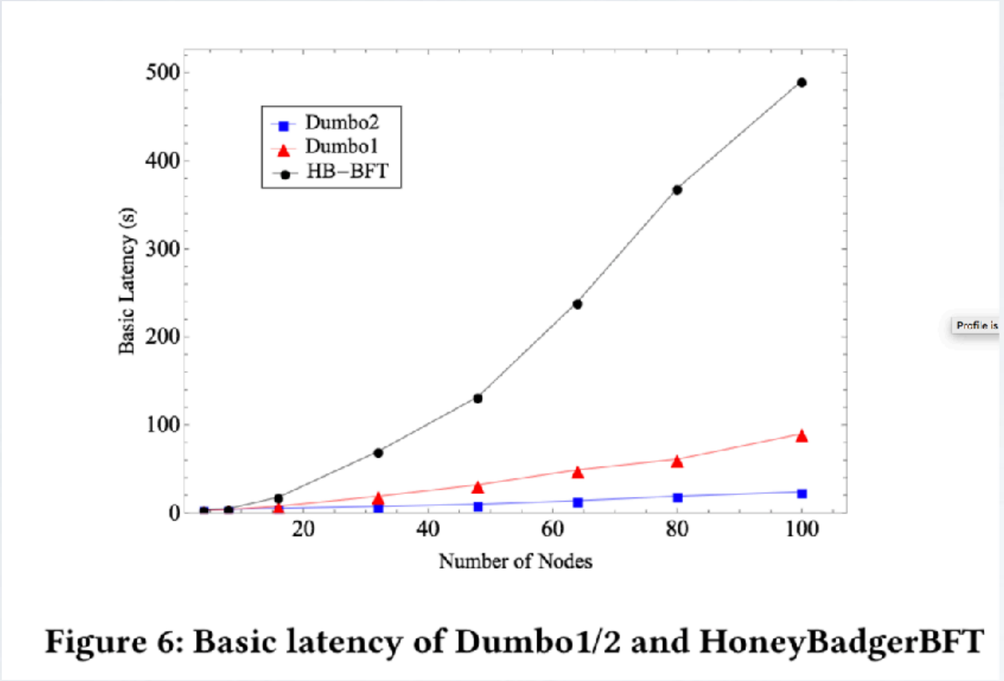
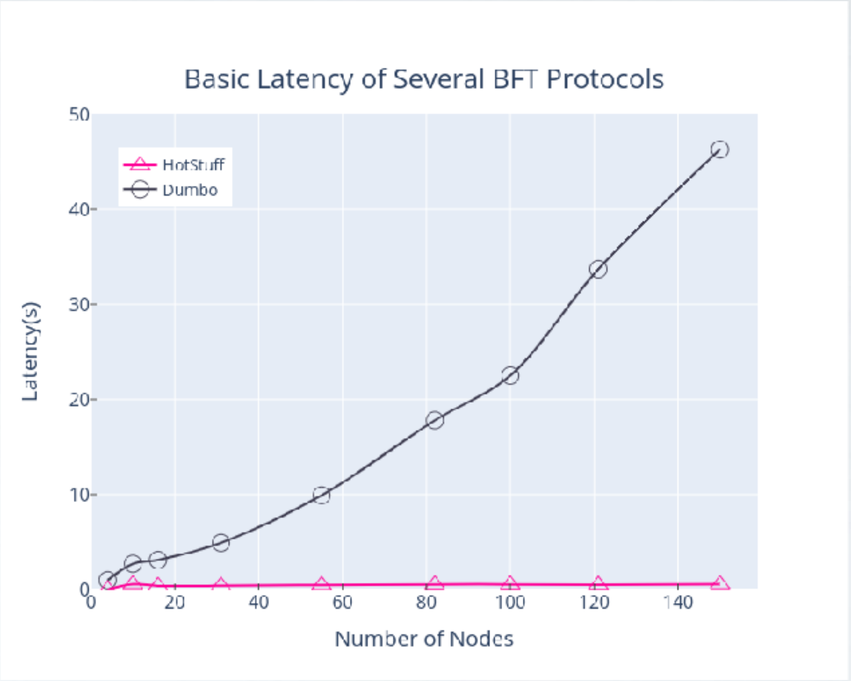


Figure 6: Basic latency of Dumbo1/2 and HoneyBadgerBFT



If we in favor of throughput, or, in a very small scale

Security-Latency Dilemma

Dete

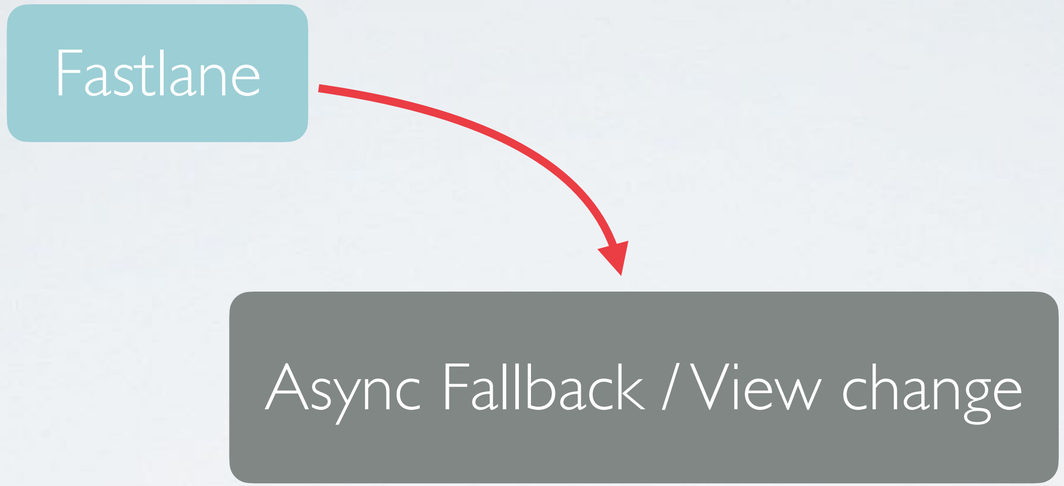
fast, bu
have

ed

still
w

Can we get the best of both? — as robust as right, as fast as left

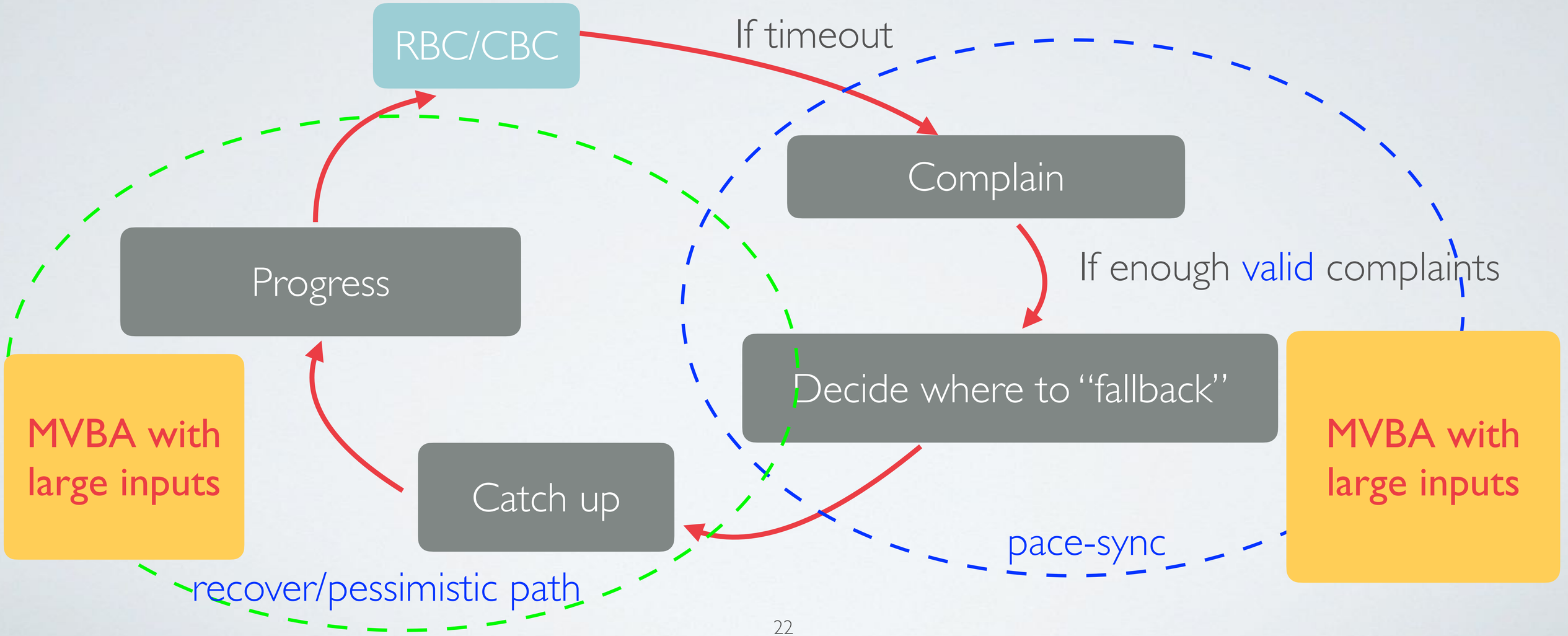
Optimistic Asynchronous Consensus



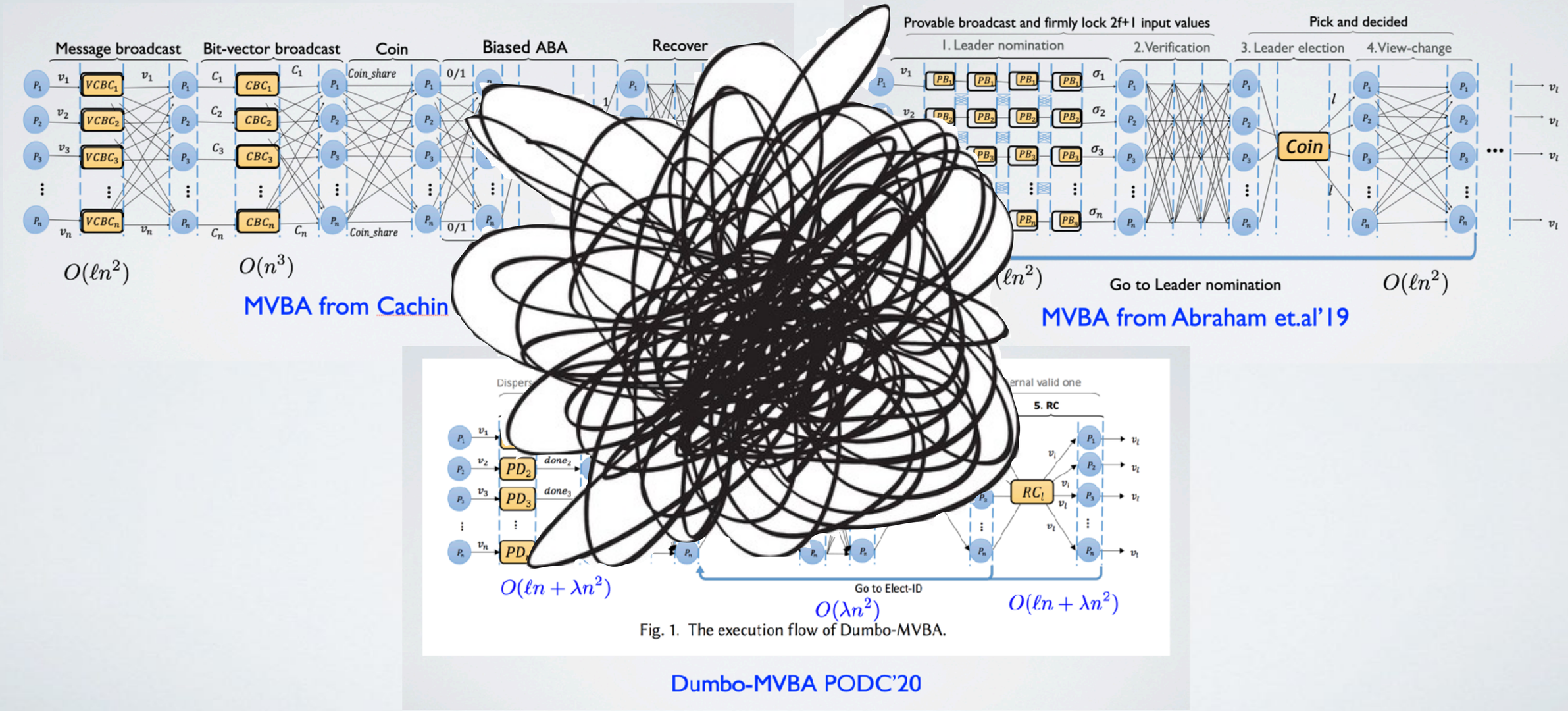
Kursawe-Shoup 02, Ramasamy-Cachin 05

Real-world networks might still be stable for most of the time

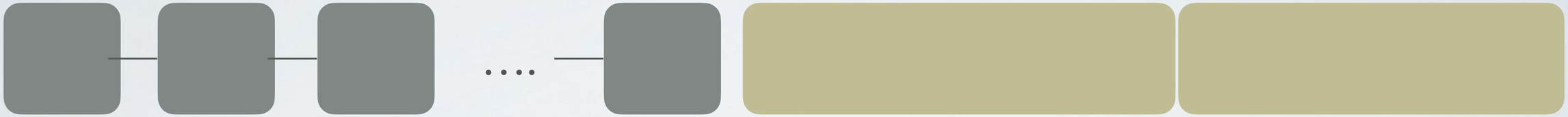
Optimistic Asynchronous Consensus [KS02,RC05]



MVBAs are still Complicated



Consequence of a Slow Pace-sync in Practice

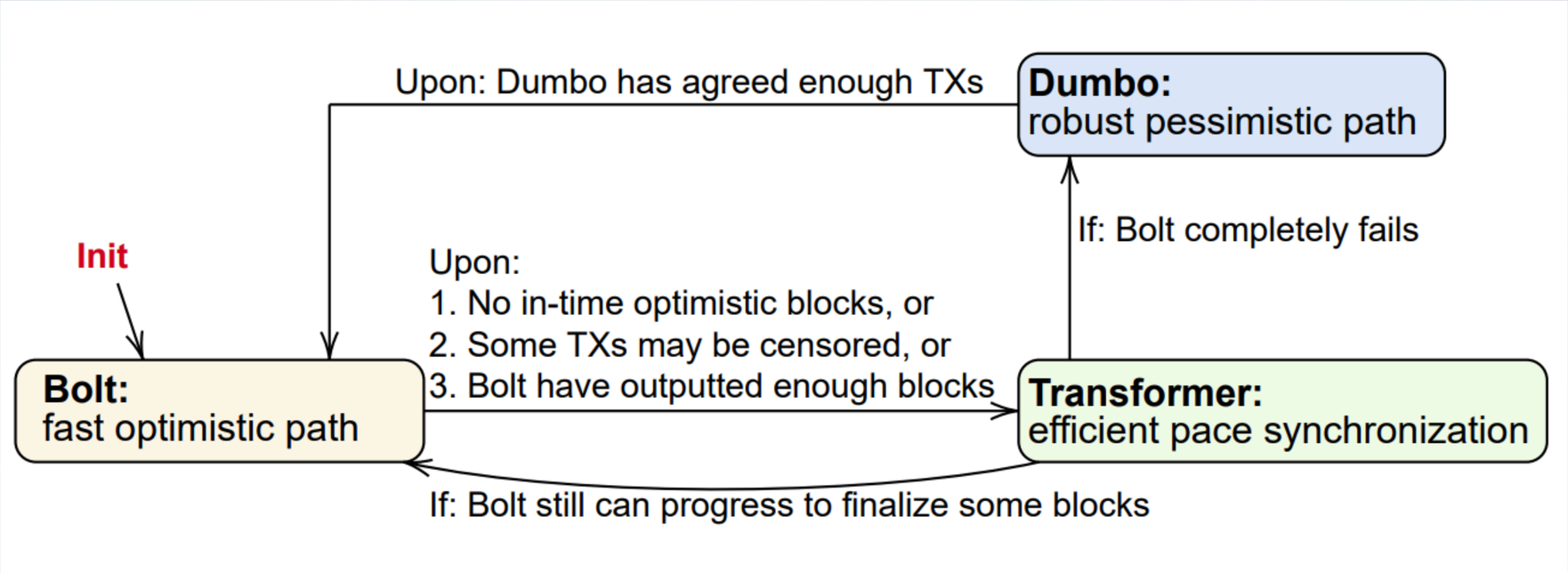


Overall still much slower than fastlane even if network is stable majority of the time

Need a super light pace-sync

And avoid pessimistic path as much as possible

Bolt-Dumbo-Transformer



First, replace pessimistic path with new ACS (e.g., sDumbo-ACS)

New abstraction for Bolt for a cheapest possible Transformer

New Abstraction of Fastlane: Bolt

Notorizable weak atomic broadcast (nwABC)

Instantiation 1:
"provable" multicast
(simplified stable-leader
HotStuff)

Instantiation2:
Provable reliable
broadcast

Easy to get, but enables cheapest possible Transformer

If an ficate,
f+1 honest guys output a valid previous one

When Bolt Got Stuck

Complain with the largest index of block (with certificate)

If receiving 2f+1 complaints — majority of honest guys complain
Pick the largest index as input to run Transformer

If fewer than f+1 honest guys making progress, no one will, thus will complain

If more than f+1 honest guys making progress, some isolated guys could simply ask for help

How Does Notarizability Prepare For Fallback?

Let us examine the input pattern for Transformer of honest nodes

Suppose the largest valid block index is v

Remember everyone receives a set C of 2f+1 complaints

Claim 1. Every honest input index at least v-1

At least f+1 honest nodes (set Good) already got v-1

C intersects with Good

How Does Notarizability Prepare For Fallback?

Let us examine the input pattern for Transformer of honest nodes

Suppose the largest valid block index is v

Now honest indices are narrowed to
v-1, v, v+1

Claim 2. No one can complain with index >v+1

Otherwise f+1 honest nodes receive v+1

How Does Notarizability Prepare For Fallback?

Let us examine the input pattern for Transformer of honest nodes

Suppose the largest valid index of honest block is v

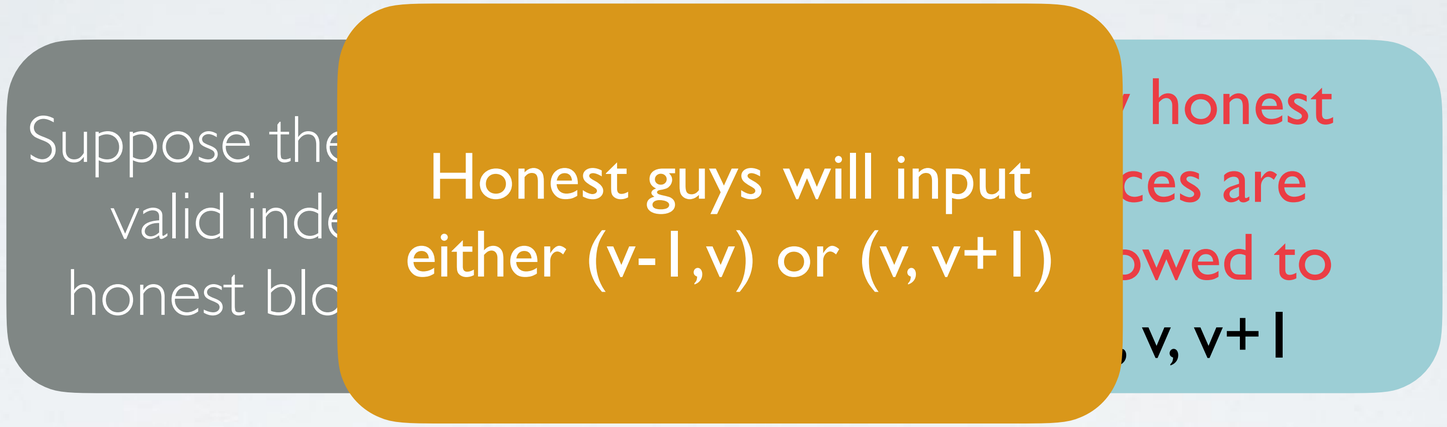
Now honest indices are narrowed to v-1, v, v+1

Claim 3. If no malicious nodes complain at indices v+1,

Then v+1 is out — it won't be any honest node's input

How Does Notarizability Prepare For Fallback?

Let us examine the input pattern for Transformer of honest nodes



Claim 4. If one malicious node complains at v+1,

Then v-1 is out — it won't be any honest node's input

At least f+1 honest nodes (set **Better**) already got v

C (the set of 2f+1 complaints) intersects with **Better**

Supercheap Pace-Sync: Transformer

Honest guys will input
either (v-1,v) or (v, v+1)

Just select one of two indices to make a decision for sync

Two-consecutive-value-BA (tcv-BA): fairly easy from **Async Binary Agreement**

Ensuring Data Retrievability

If tcv-BA output u

u could be v+1, thus no honest guy has this block

Transformer asks everyone to sync to u-1

But it is also possible u=v-1, or v; sync to u-1 may revoke committed blocks

Fine-tune Bolt: run nwABC to output one block in every two blocks,
with one pending, complain at latest pending

Bolt-Dumbo Transformer

Bolt: "handicapped": Easy to get

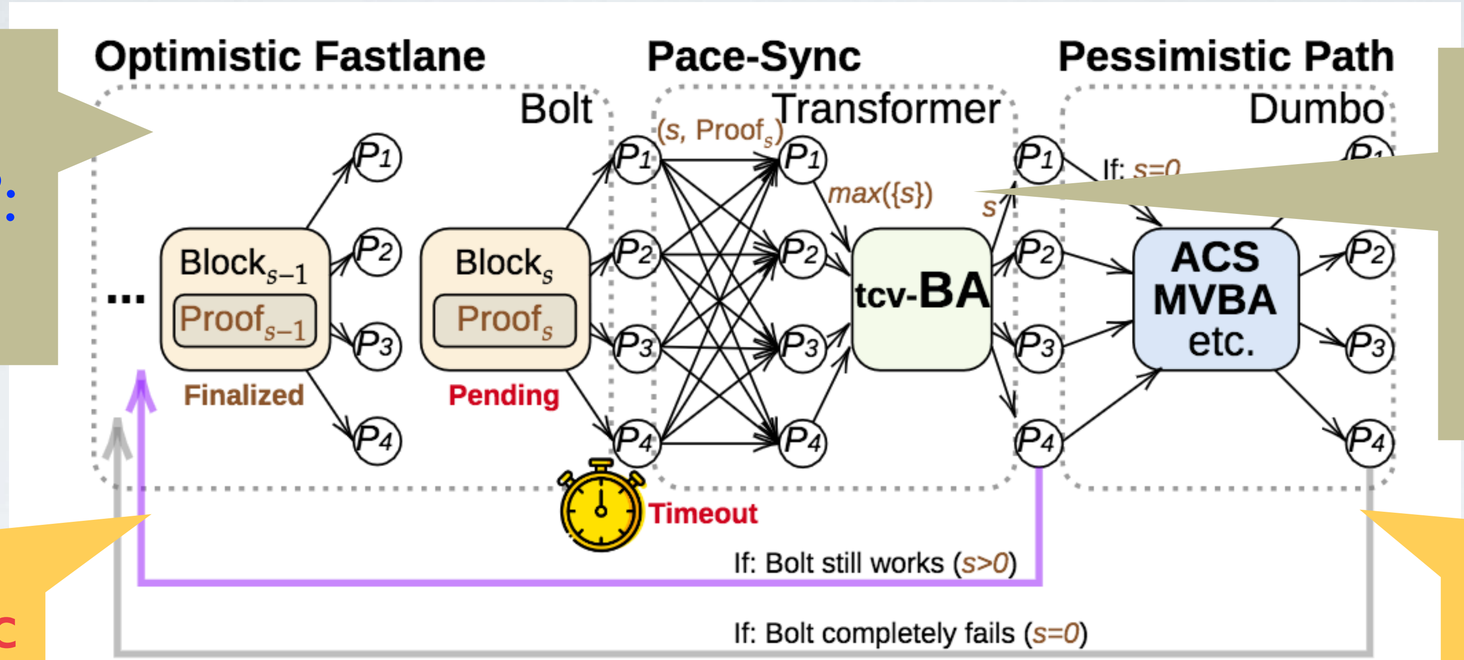


Fig. 11: The execution flow of Bolt-Dumbo Transformer

Enabled Simplest possible Transformer: AB(inary)A

Skip pessimistic path if there is some progress

Every component could use the best instantiation

Basic Latency and Overhead

100 AWS EC2 c5.large instances (2vCPU, 4G memory, "humble" configuration) uniformly spread in 16 regions world-wide

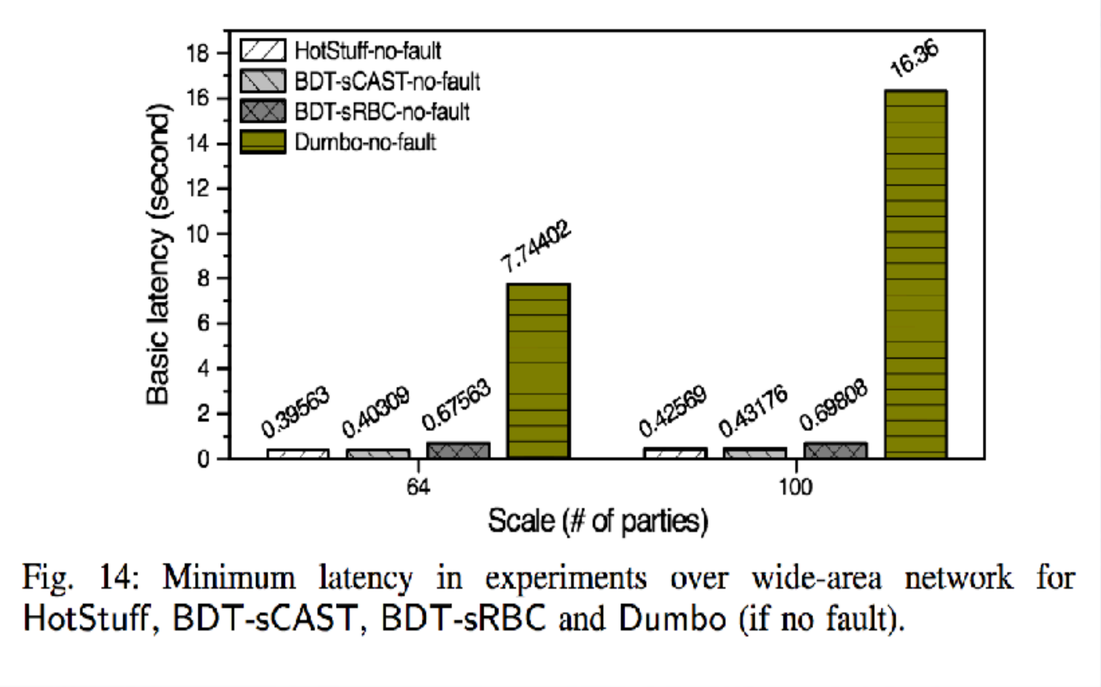
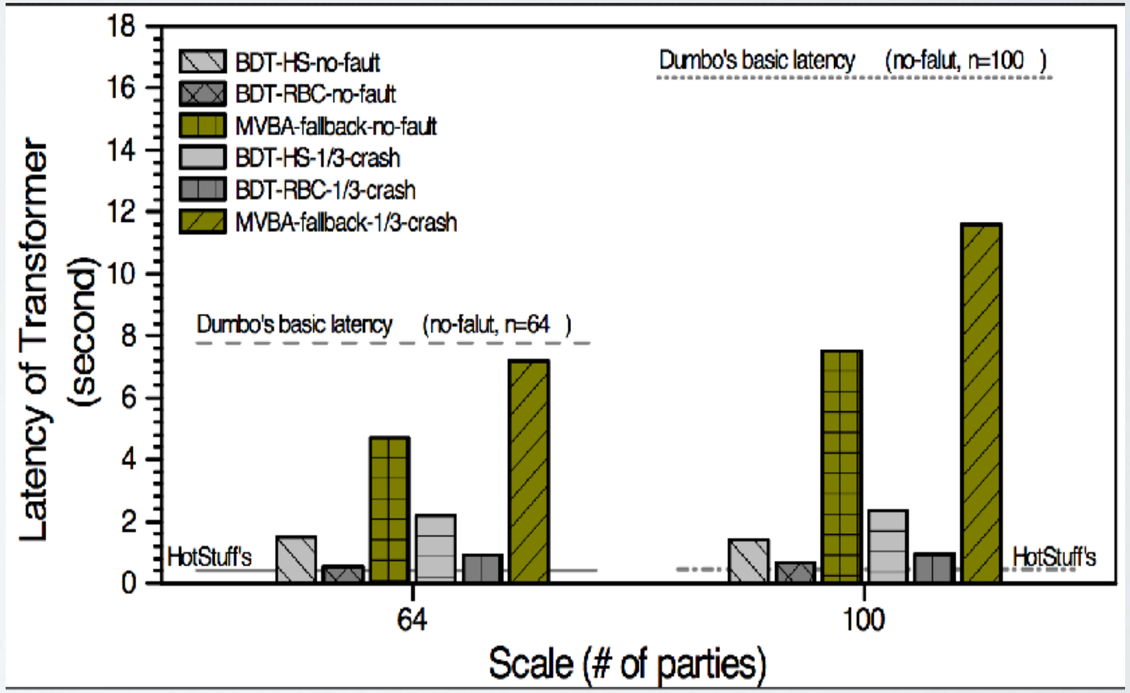


Fig. 14: Minimum latency in experiments over wide-area network for HotStuff, BDT-sCAST, BDT-sRBC and Dumbo (if no fault).



We intentionally run Transformer once every 50 blocks

The RBC based fastlane makes Transformer to terminate faster….

Throughput

100 AWS EC2 c5.large instances (2vCPU, 4G memory, "humble" configuration) uniformly spread in 16 regions world-wide

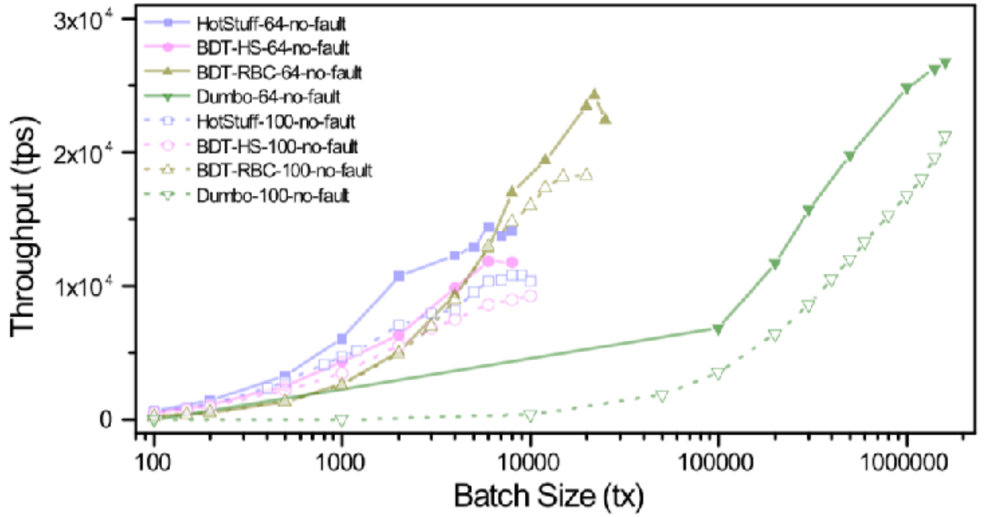


Fig. 14. Throughput v.s. batch size for experiments over wide-area network when $n = 64$ and $n = 100$, respectively (in case of no fault).

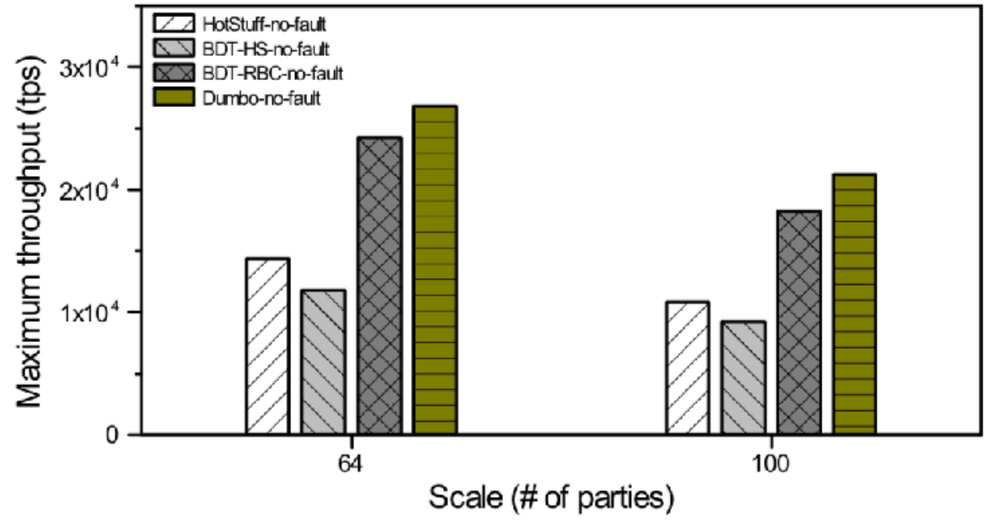


Fig. 10. Highest throughput in experiments over wide-area network for HotStuff, BDT-HS, BDT-RBC and Dumbo (if no fault).

The RBC based fastlane has a much larger throughput with large batch size

More experimental and numerical analysis under different settings

Throughput/Latency Tradeoff

100 AWS EC2 c5.large instances (2vCPU, 4G memory, "humble" configuration)
uniformly spread in 16 regions world-wide

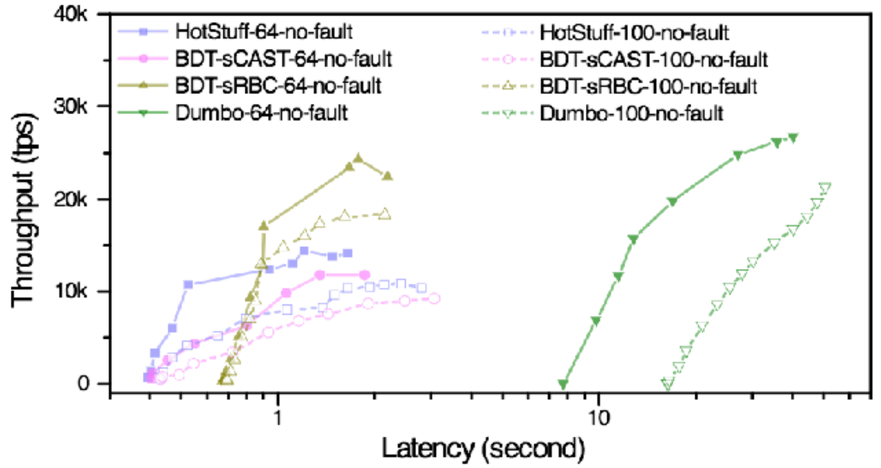


Fig. 18: Throughput v.s. latency for experiments over wide-area network when $n = 64$ and $n = 100$, respectively (in case of no fault).

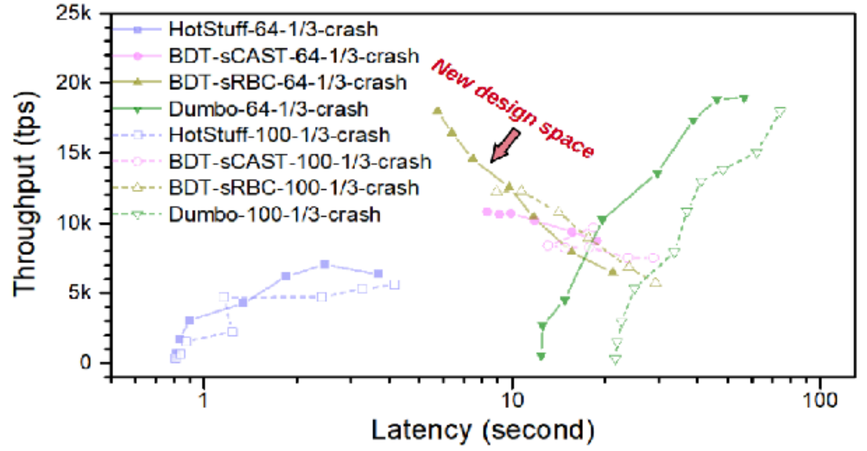
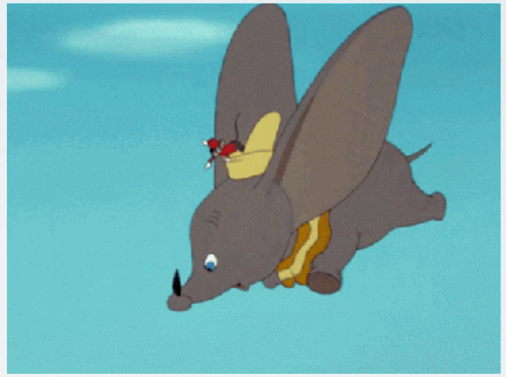


Fig. 19: Throughput v.s. latency for experiments over wide-area network when $n = 64$ and $n = 100$, respectively (in case of 1/3 crash fault). We fix the fallback batch size of BDT instances to 10^6 transactions in all tests.

We intentionally run Transformer
once every 50 blocks

HotStuff is with a stable leader
Intentionally trigger pace-sync for all muted
Timeout = 2.5s, could be shorter

Towards Making Asynchronous Consensus Real



The seemingly simple Broadcast-then-Agree structure is simplified dramatically

Best trade-offs among comm, comp, rounds, setup, etc?

Is there inherent gap between async protocols and deterministic protocols?

Scale to thousands of nodes?

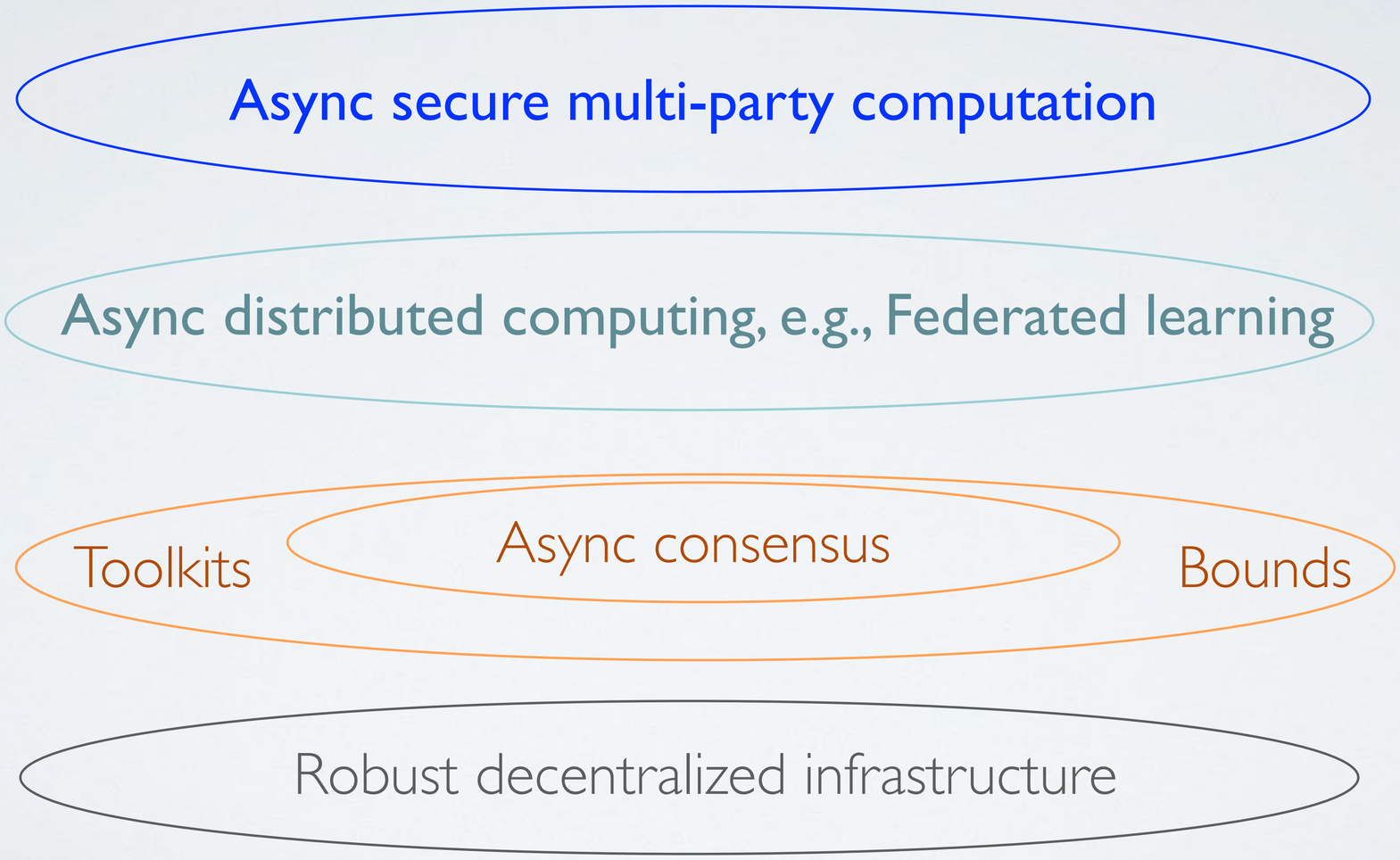
ethereum foundation

Stellar Development Foundation

Protocol Labs Research

Algorand™ FOUNDATION

A Slightly Bigger Picture



Welcome to U Sydney

Dumbo Protocol Family
Making Asynchronous Consensus Real

Qiang Tang
The University of Sydney
qiang.tang@sydney.edu.au
http://alkistang.github.io/