Bolt-Dumbo Transformer: Asynchronous Consensus As Fast As the Pipelined BFT

Authors: Yuan Lu, Zhenliang Lu, Qiang Tang

Presenters: Anubhav Mishra, Sriharshini D, Aakash Kotha, Thrisha Kopula

Key Terms

- **Synchronous protocols** - messages will be delivered within some known delay (Upper bound)
- **Asynchronous protocols** - There are no fixed bounds on message delivery time.
- **Partial Synchronous protocols** - Asynchronous before some unknown point in time (Global Standardization Time), and synchronous after that

Synchronous vs Asynchronous

- **Problem with Synchronous protocols:**
  - Synchronous protocols have threat from DOS attacks, fluctuating bandwidth, unreliable links, substantial delays that may compromise safety and liveness in an asynchronous network setting

- **Need of Asynchronous protocols:**
  - More robust in adversarial conditions
  - No Manual timeouts

Synchronous vs Asynchronous

Why are asynchronous consensus not practical for a long time?

Synchronous vs Asynchronous

Why are asynchronous consensus not practical for a long time?

- FLP impossibility!!
  - “ No deterministic protocol can ensure both safety and liveness in an asynchronous network.”
  - Safety, liveness, fault tolerance or asynchrony?

- Asynchronous consensus is complicated and slower
- Many attempts were just theoretical

First asynchronous in practice

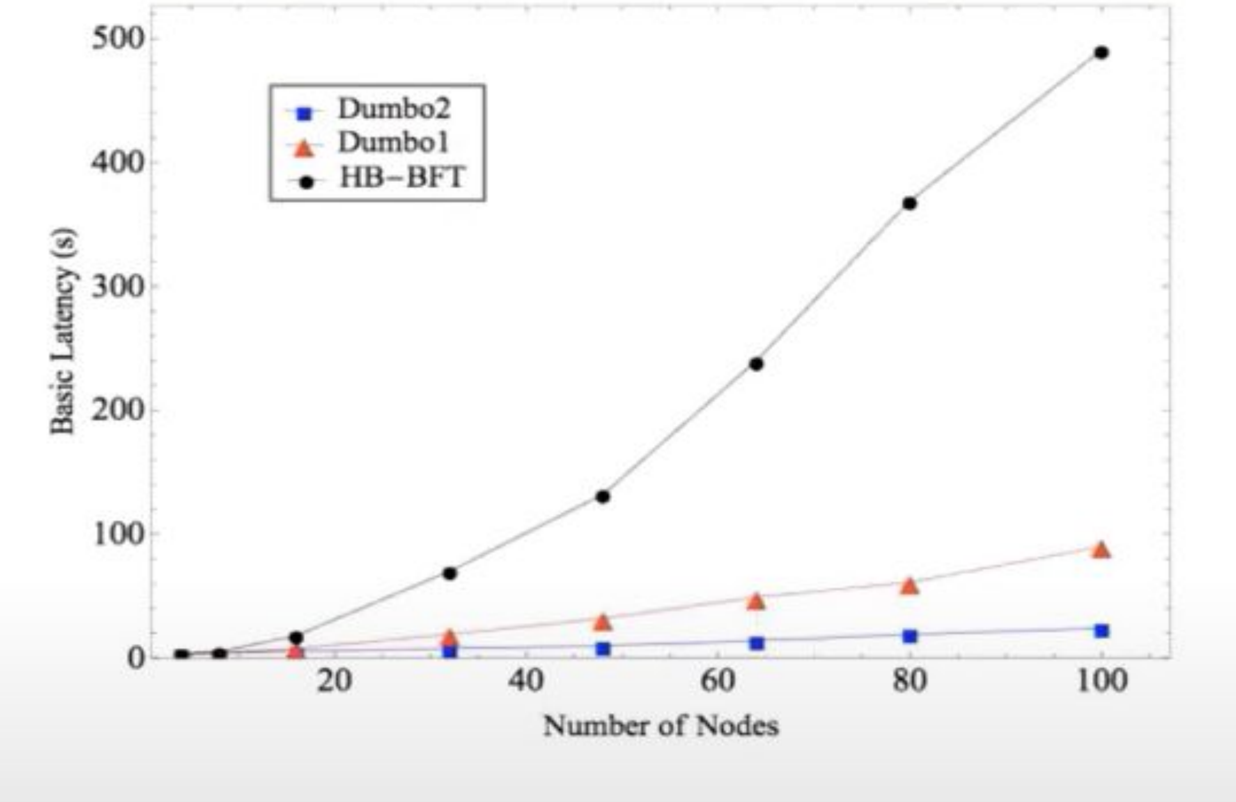
- HBBFT - First Practical Asynchronous Protocol
- 2 Phases (RBC & ABA)
- RBC: A special type of broadcast protocol
- ABA: Binary Agreement Phase
- In ABA, Each party has multiple joint instances running parallel to vote for each and every Transaction
- So, the complete 2nd phase depends on slowest instance
- So, **Dumbo!!!**

Dumbo

- Asynchronous Common Subset (ACS)- Every honest party input values and outputs “**set**” of values
- Instead of ABA in HBBFT, we use Multi-valued Validated Byzantine Agreement (MVBA)
- Predefined predicate to validate whether the output is from a honest node or not
- MVBA is heavy tool, if inputs are large
- So, we send indexes as inputs instead
- RBC + MVBA = “Dumbo”

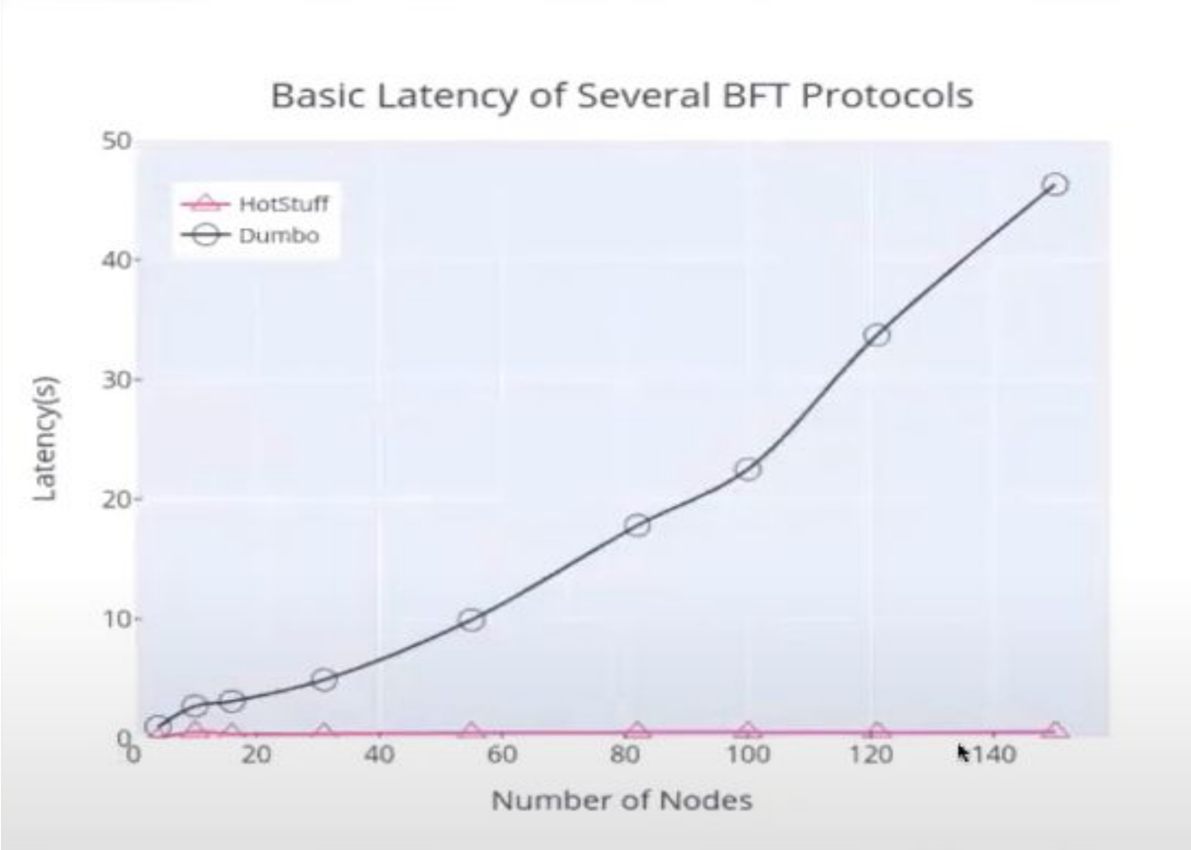
Latency Comparison

- Dumbo >> HB-BFT (Performance)



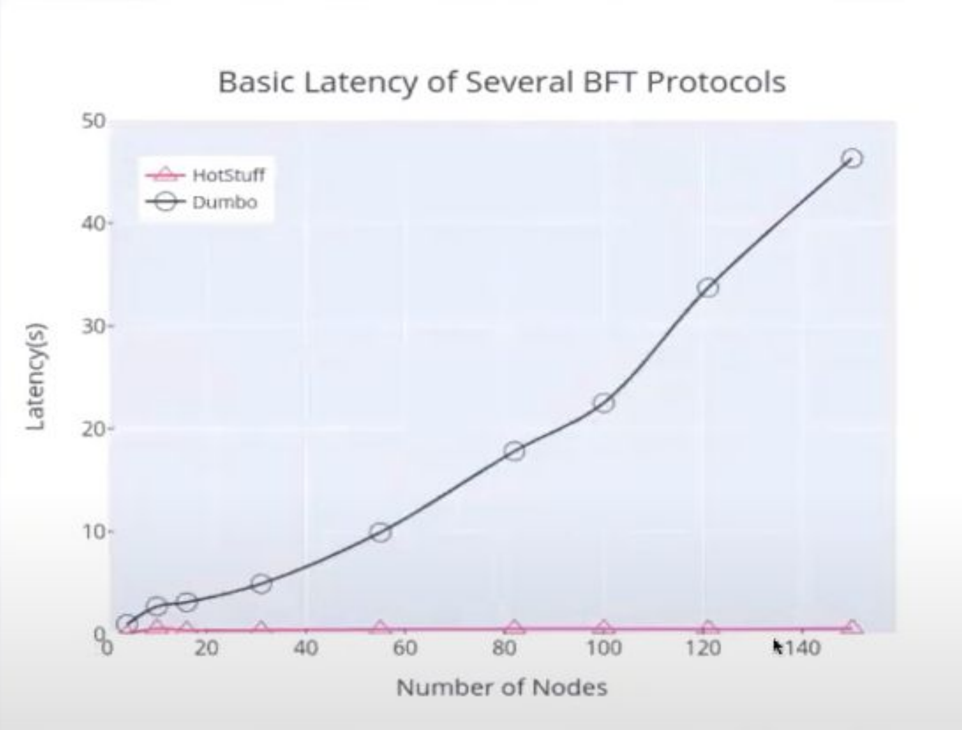
Latency Comparison (Cont.)

- However, Hotstuff >> Dumbo (Performance)



Latency Comparison (Cont.)

- But, Hotstuff >> Dumbo (Performance)



- So, there arises a need to design something even better!

Security vs Latency Dilemma

Synchronous:
Fast, but may not have Safety

Asynchronous:
Robust, but still quite slow

- Dilemma - we choose safety or fastness??

Can we get the best of both?

Security vs Latency Dilemma

Synchronous:
Fast, but may not have Safety

Asynchronous:
Robust, but still quite slow

- Dilemma - we choose safety or fastness??

Can we get the best of both?

" Optimistic Asynchronous Atomic Broadcast!! "

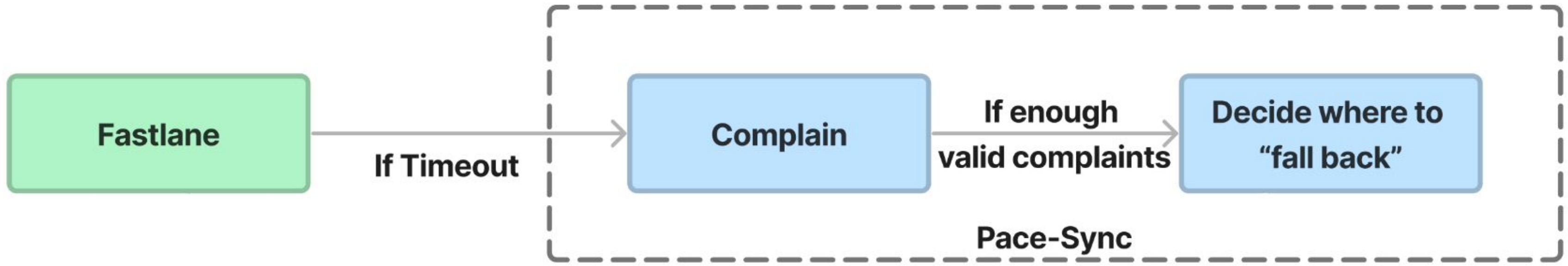
Optimistic Asynchronous Atomic Broadcast

- Framework that was proposed to improve slow, asynchronous consensus
- Consists of:
  - Deterministic Fastlane
    - Runs a deterministic protocol
  - Pessimistic Path
    - Runs an asynchronous protocol
  - Pace-Synchronization Mechanism
    - Uses a heavy Multi-Validated Byzantine Agreement (MVBA)

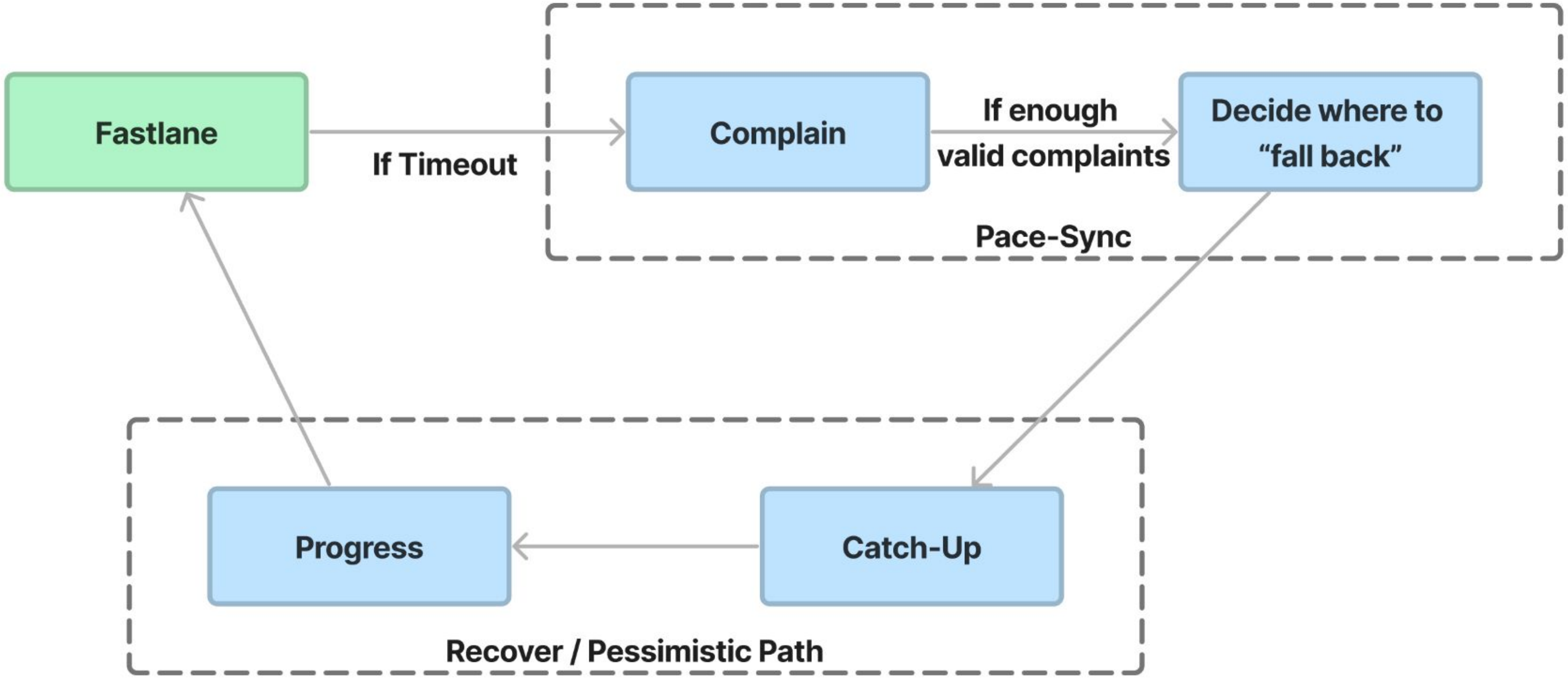
Fastlane

Fastlane

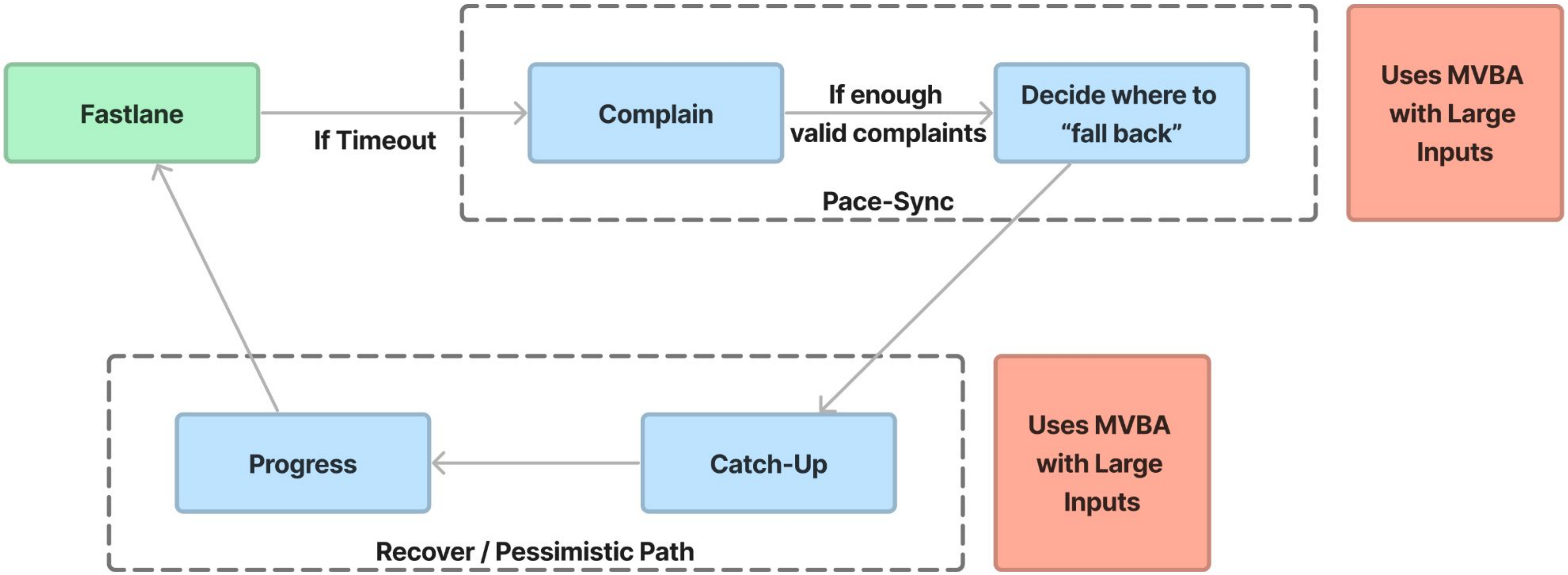
Pace-Synchronization



Pessimistic Path

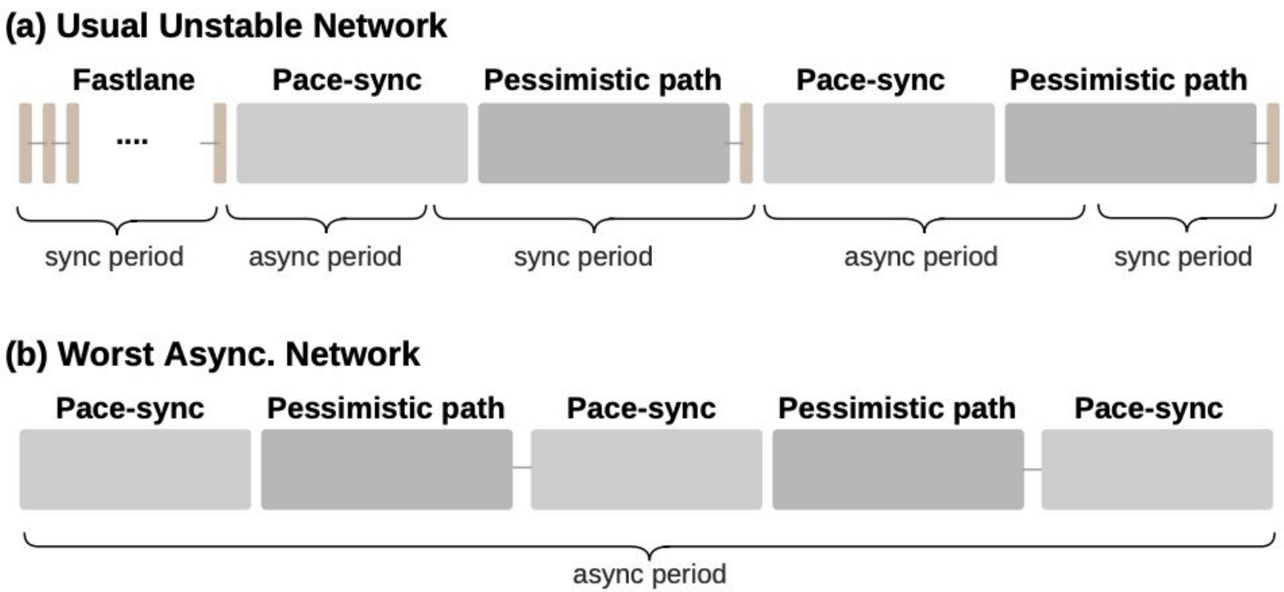


Pessimistic Path



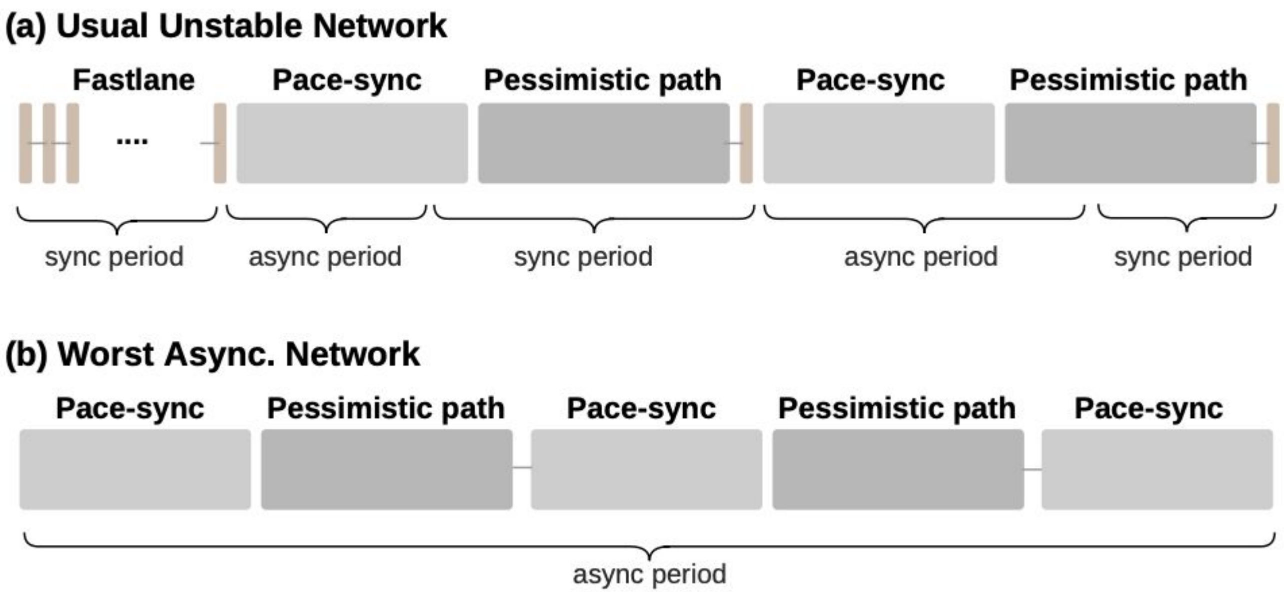
UCDAVIS

Optimistic Asynchronous Consensus (Cont.)



- Problems with this:
  - Pace-sync mechanism too heavy
  - With frequent fallbacks, eliminates the benefits of adding a Fastlane
- We need a super light pace-sync and be able to utilize the fastlane more

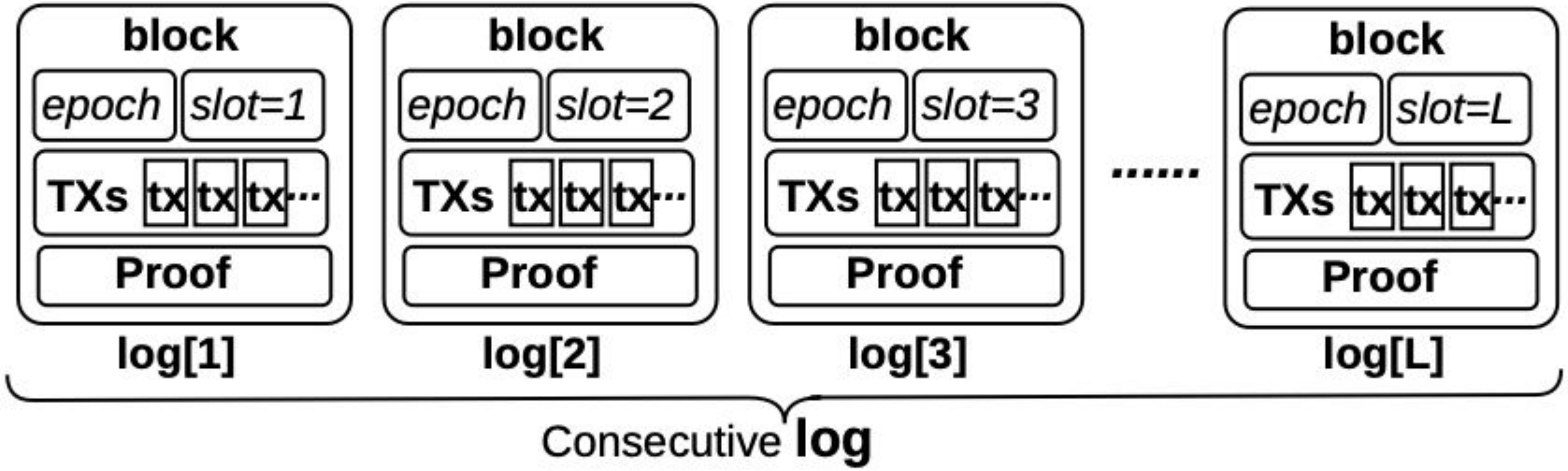
Optimistic Asynchronous Consensus (Cont.)



- Problems with this:
    - Pace-sync mechanism too heavy
    - With frequent fallbacks, eliminates the benefits of adding a Fastlane
- We need a super light pace-sync and be able to utilize the fastlane more

“Bolt-Dumbo Transformer (BDT)!!”

Terminologies of Block



- **log** - list of blocks
- **epoch** - number that represents the round of operation
- **slot** - index number of blocks in epoch
- **TXs** - sequence of transactions (payload)
- **Proof** - quorum proof that attests that at least f + 1 honest parties contain the previous block

Bolt-Dumbo Transformer (BDT)

- Bolt (fastlane)
  - uses notarizable-weak atomic broadcast (nw-ABC) to allow for a simple pace-sync mechanism
  - runs a deterministic protocol to quickly progress through synchronous network conditions

- Transformer (pace-synchronization mechanism)
  - uses a much simpler two-consecutive-valued Byzantine agreement (tcv-BA)

- Dumbo (pessimistic path)

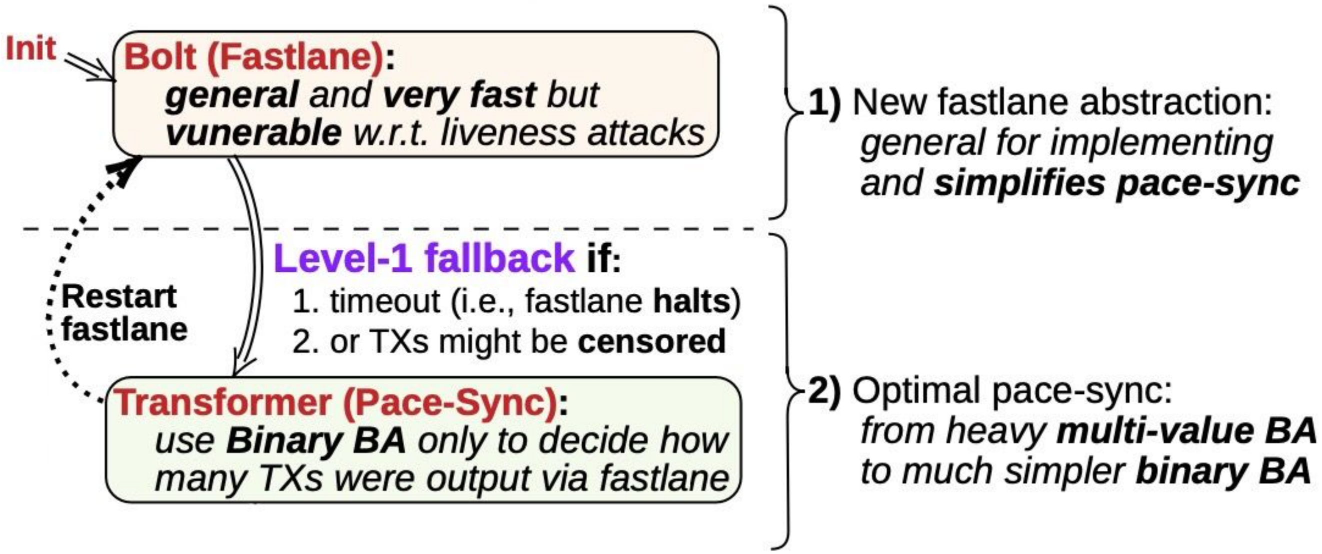
  runs an asynchronous protocol to ensure liveness

Overview of BDT Framework

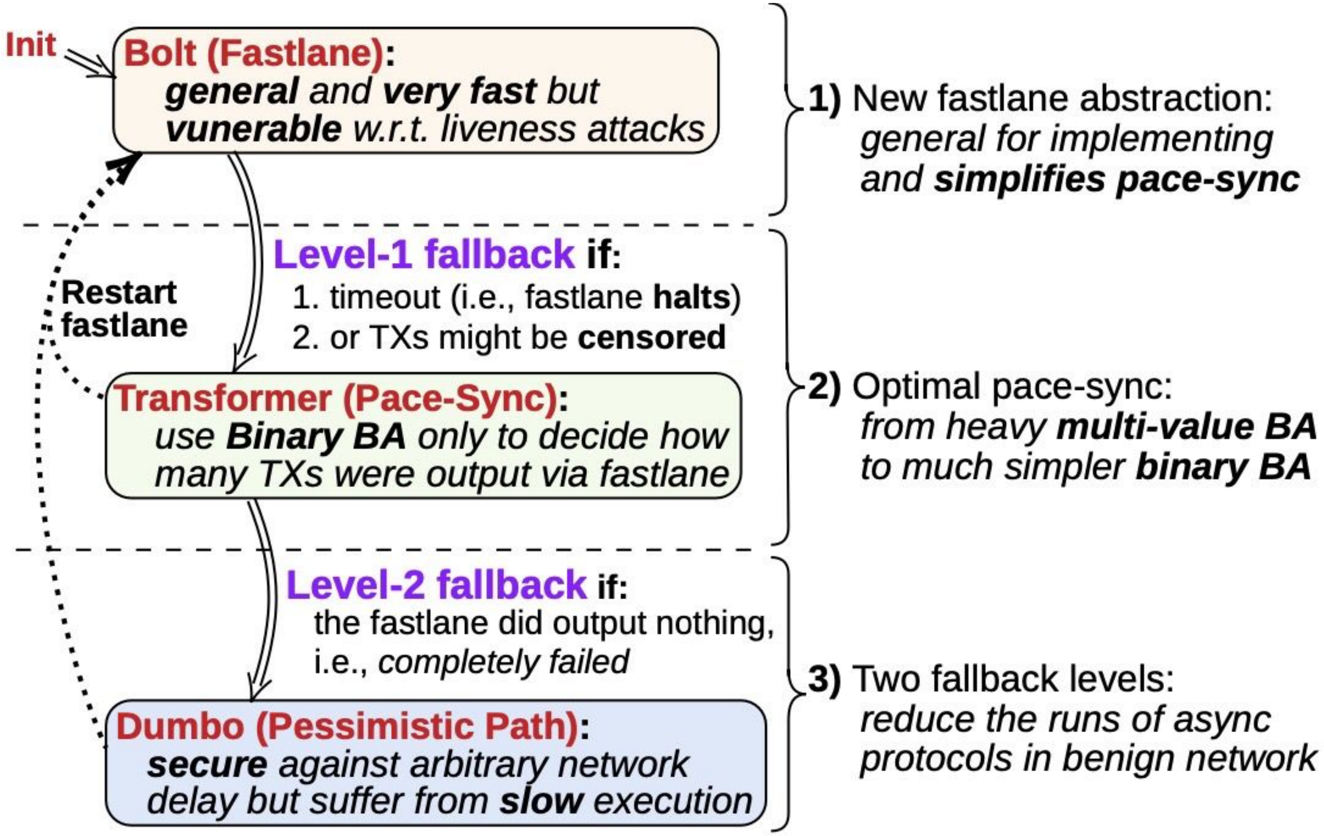
Init → **Bolt (Fastlane)**:
general *and* ***very fast*** *but*
vunerable *w.r.t. liveness attacks*

1) New fastlane abstraction:
general for implementing
and ***simplifies pace-sync***

Overview of BDT Framework



Overview of BDT Framework



Bolt - Notarizable-Weak Atomic Broadcast (nw-ABC)

- “Handicapped consensus”
  - fastlane that might keep on progressing under optimistic conditions: Leader is honest and Network is synchronous. (similar to Hotstuff and PBFT)

- Notarizability property:
  - Whenever any party outputs a block at position j with a valid quorum proof, at least $f + 1$ honest parties already output at the position $j - 1$

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is ‘s’

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is ‘s’

“**Claim-1:** There is honest node which input index at least s-1”

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is 's'

"**Claim-1:** There is honest node which input index at least s-1"

At least f+1 honest nodes (Set Good) already got s-1

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is ‘s’

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

“**Claim-1:** There is honest node which input index at least s-1”

At least f+1 honest nodes (Set Good) already got s-1

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is 's'

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

"**Claim-1:** There is honest node which input index at least s-1"

At least f+1 honest nodes (Set Good) already got s-1

At least one common between these 2 sets (C & Good)

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is ‘s’

“**Claim-2:** No one can complain with index > s+1”

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is ‘s’

“**Claim-2:** No one can complain with index > s+1”

If anyone can complain with index greater than s+1 then, according to notarizability, there will be f+1 honest nodes already got s+1!

How does notarizability enable cheaper pace-sync?

Suppose the largest valid index of a honest node is ‘s’

“**Claim-2:** No one can complain with index > s+1”

If anyone can complain with index greater than s+1 then, according to notarizability, there will be f+1 honest nodes already got s+1!

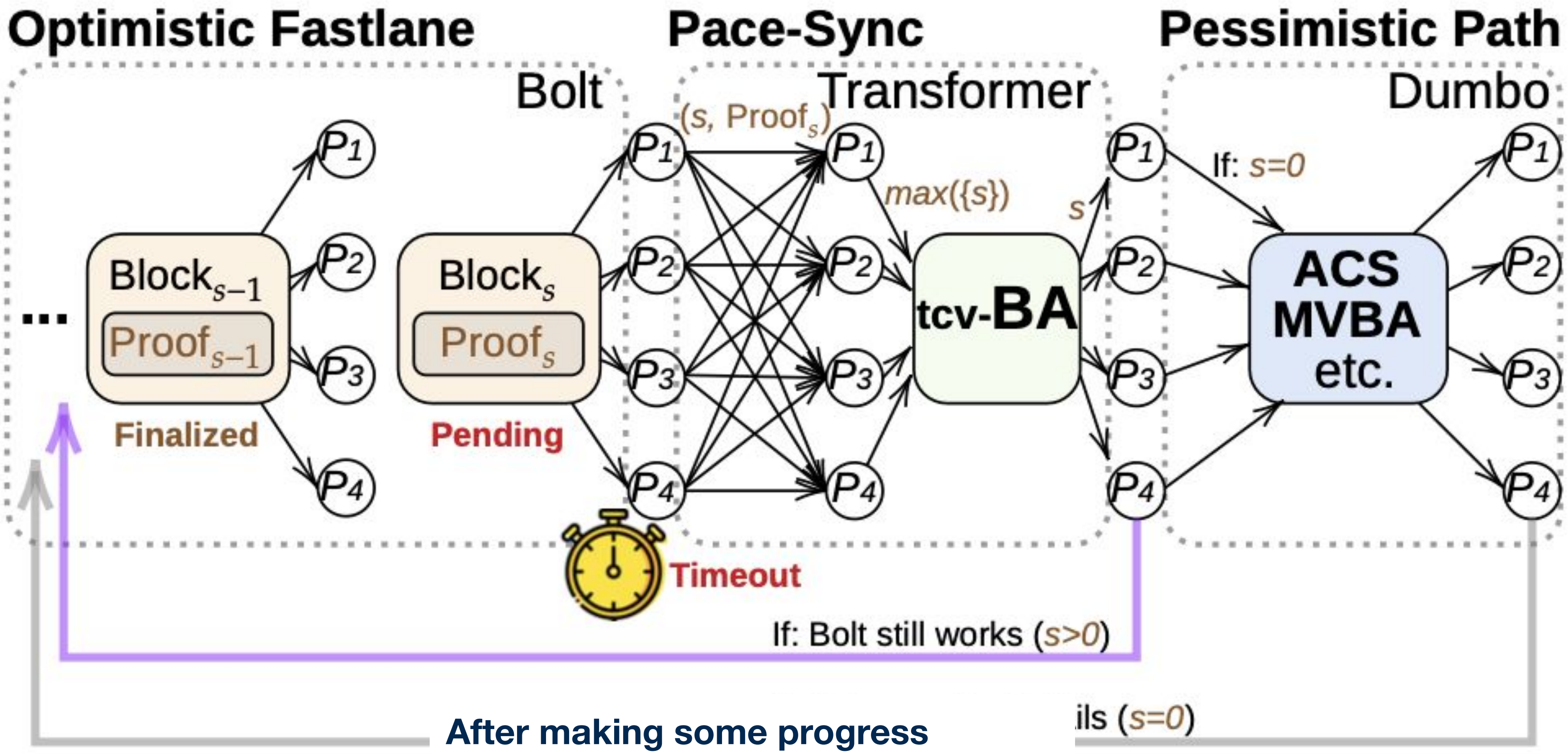
How does notarizability enable cheaper pace-sync?

- We can make two claims:
  - No honest party can see a valid fallback request with an index >= s + 1
  - All honest parties must see some fallback request with an index >= s - 1
- These two claims narrow the range of fallback positions to {s-1, s}

Two-Consecutive-Valued Byzantine Agreement (tcv-BA)

- Asynchronous agreement for consecutive values
- Only has to choose a value s between {s-1, s}
- After s is chosen, we check:
  - If s > 0, progress was made in the fastlane, so we go back to the fastlane
  - If s = 0, no progress was made in the fastlane, so we switch to the pessimistic path
- Utilizing the fastlane more and avoiding the use of pessimistic path as much as possible

Execution Flow



How safety is ensured?

- Transformer returns a common index which all parties have to sync up to
- The parties will then continue onto the pessimistic path
- Transformer will choose an index that is not too large:
  - Will contradict the notarizability property - cannot guarantee that f + 1 parties have all block up to that index
- Transformer will choose an index that is not too small:
  - No honest party can revoke any fastlane block that was already committed

How liveness is ensured?

- Fastlane has timeouts which ensure parties are not stuck
- If any party has missing blocks, f+1 honest parties will help fetch them and so no honest party will be stuck at pace synch phase
- Pessimistic path ensures that any transactions can output with a constant probability, thus ensuring liveness even if in the worst case

Performance & Evaluation-Latency

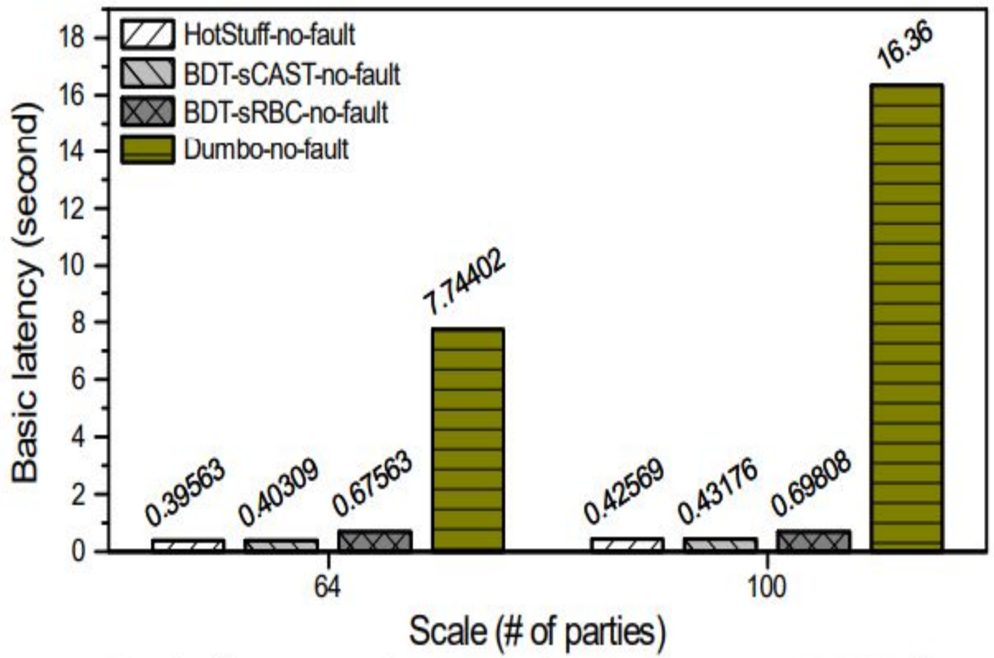


Figure 15: Basic latency in experiments over WAN for two-chain HotStuff, BDT-sCAST, BDT-sRBC and Dumbo.

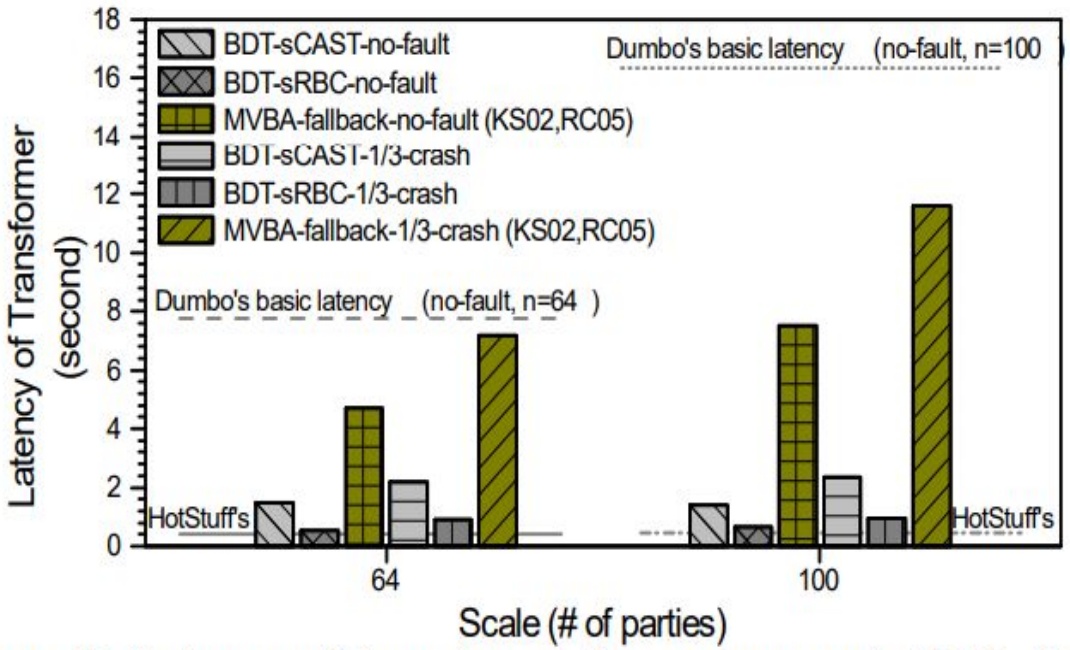


Figure 17: Latency of Transformer for pace-sync in BDT-sCAST and BDT-sRBC (when no fault and 1/3 crash, respectively).

UCDAVIS

Performance & Evaluation-Throughput

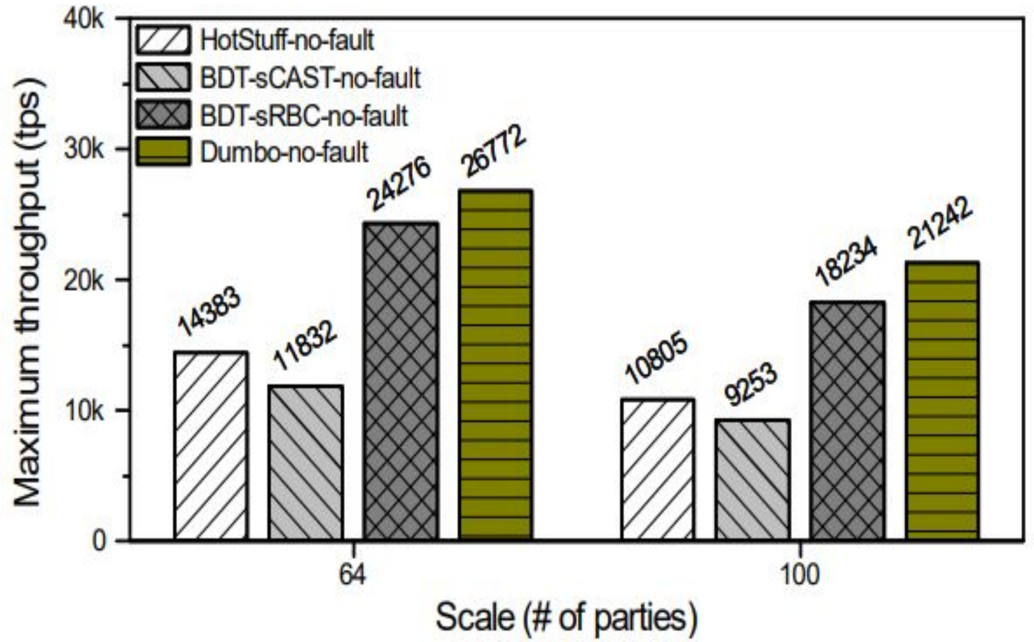


Figure 16: Peak throughput in experiments over WAN for two-chain HotStuff, BDT-sCAST, BDT-sRBC and Dumbo.

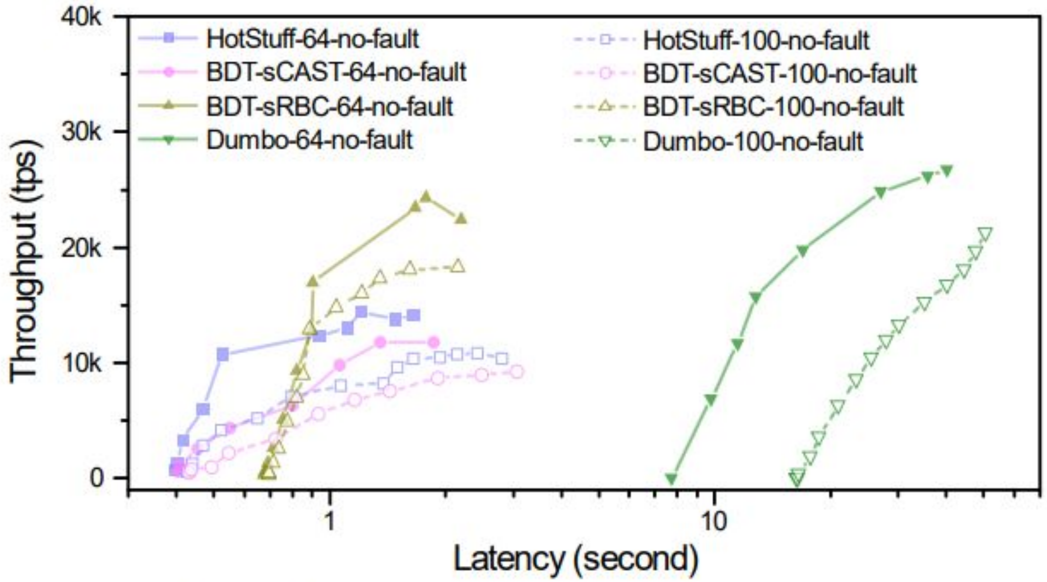


Figure 19: Throughput v.s. latency for experiments over WAN when $n = 64$ and 100, respectively (in case of periodically running pace-sync in BDT per only 50 fastlane blocks).

Performance & Evaluation-In Bad Networks

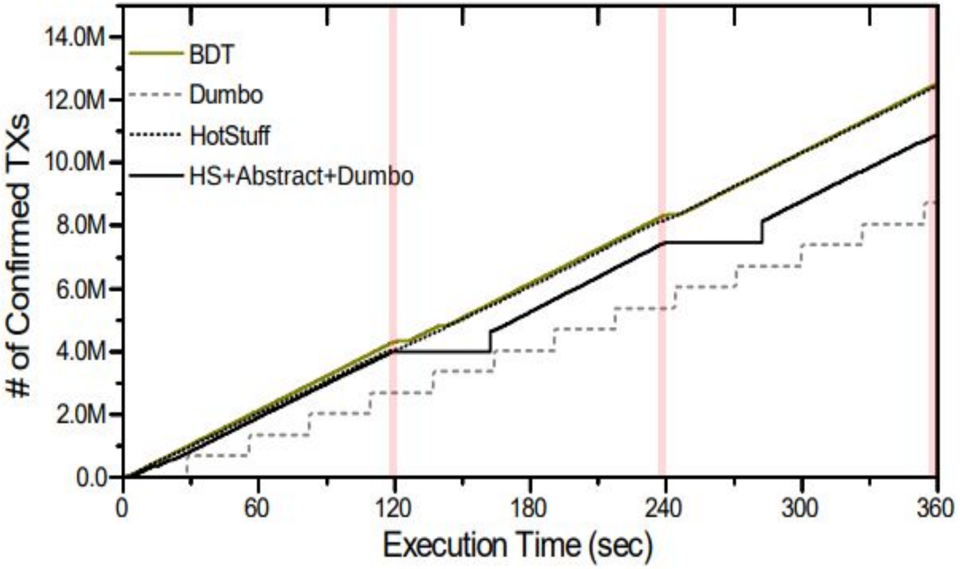


Figure 20: Sample executions of BDT, **2-chain** HotStuff, Dumbo, **and the composition of** HotStuff+Abstract+Dumbo **for** ***n*=64, when facing a few 2-second bad periods. The red region represents the 2-second period of bad network.**

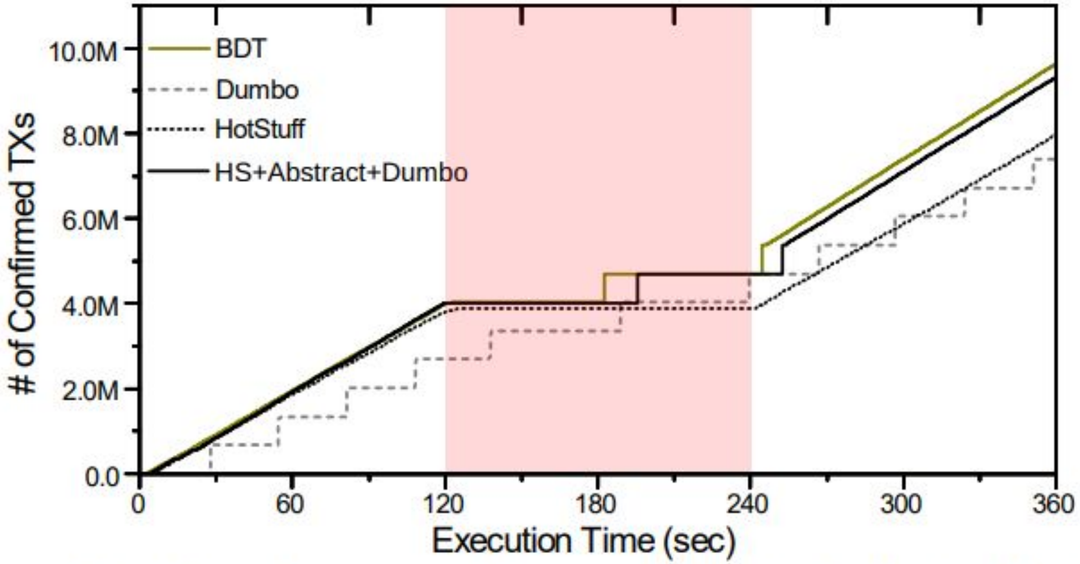


Figure 21: Sample executions of BDT, **2-chain** HotStuff, Dumbo, **and the composition of** HotStuff+Abstract+Dumbo **for** ***n*=64, when suffering from 120-second bad network. The red region represents the 120-second period of bad network.**

References

1. https://www.youtube.com/watch?v=mOe1_8Q6DjI
2. https://arxiv.org/pdf/2103.09425.pdf
3. https://dl.acm.org/doi/abs/10.1145/3548606.3559346
4. https://dl.acm.org/doi/10.1145/3382734.3405707

Thank You

(Any Questions?)