Dumbo-NG: Fast Asynchronous BFT Consensus with Throughput-Oblivious Latency by Gao et.al

Presented by Shujuan Chen, Tianxiao Cheng, Junlong Tang, Hao-Nan Zhu

Dumbo-NG: Fast Asynchronous BFT Consensus with Throughput-Oblivious Latency

Why asynchronous BFT?

- Adversarial Resilience: Essential for networks with high risk of malicious attacks, especially in financial and banking services.
- Critical Application Support: Ideal for infrastructures hosting essential services, where disruptions have significant consequences.
- Network Instability Tolerance: Effective in environments with network fluctuations, misconfigurations, or targeted network attacks.
- Guaranteed Liveness and Safety: Ensures continuous operation (liveness) and accurate consensus (safety) despite message delays.
- Superior to Synchronous Protocols: Overcomes limitations of synchronous consensus mechanisms, which may fail in unstable network conditions.
- High Security Assurance: Provides robust protection against colluding nodes and unpredictable adversarial behaviors.

Throughput-Oblivious Latency

Dumbo-NG's latency, which is lowest among all tested protocols, can almost remain stable when throughput grows

Dumbo-NG realizes a peak throughput 4-8x over Dumbo, 2-4x over Speeding-Dumbo, and 2-3x over sDumbo-DL for varying scales

1. Introduction

1.1 Practical obstacles of adopting asynchronous BFT consensus

Trade off between high security assurance, low latency, and high throughput

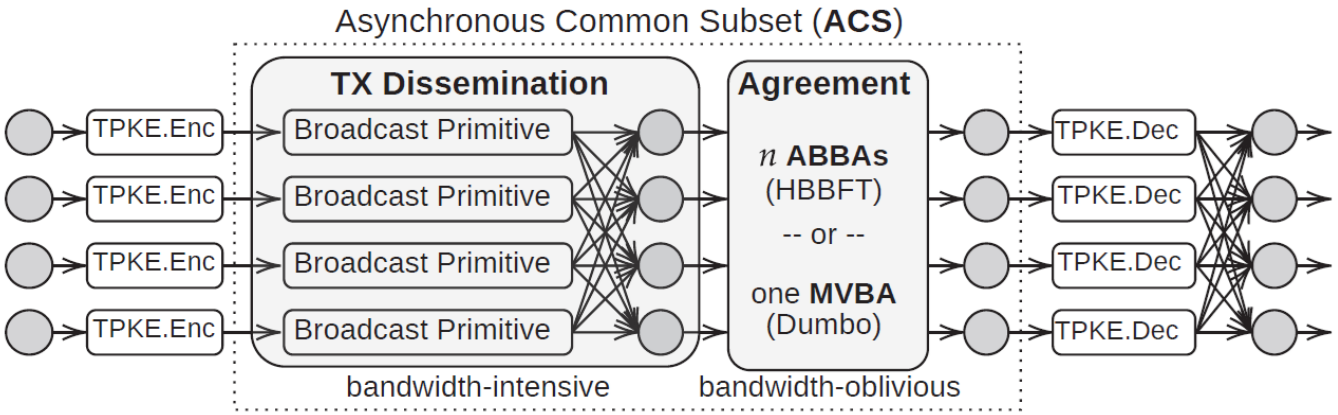


Figure 1: Execution flow of an epoch in HBBFT, Dumbo and their variants. The protocols proceed by consecutive epochs.

- Throughput is severely hurting latency

Transaction dissemination phase

Agreement phase: incur large

latency that blocks next epoch's

TD phase

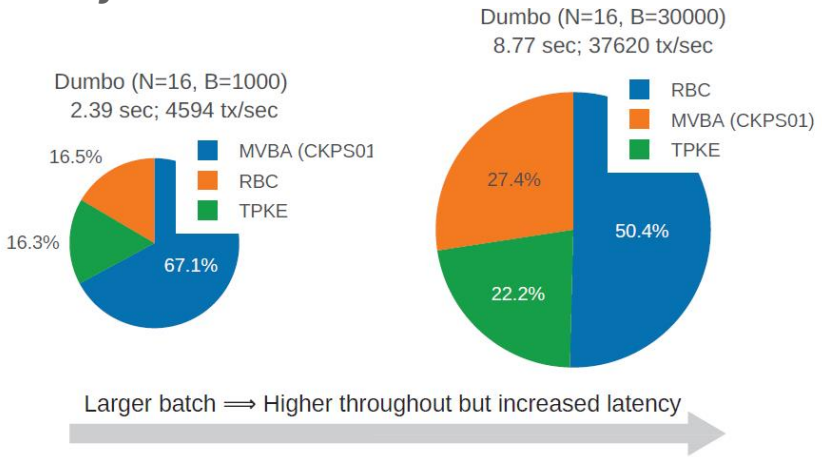
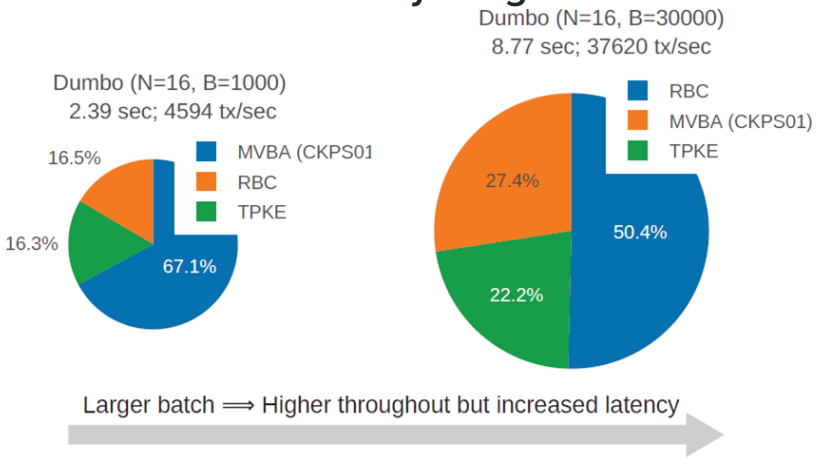


Figure 2: Latency breakdown of Dumbo **(on 16 Amazon EC2 c5.large instances across different regions).** $|B|$ **is batch size, i.e., the number of tx to broadcast by each node (where each tx is 250-byte to approximate the size of Bitcoin's basic tx). The use of TPKE is from** HBBFT **to prevent censorship.**

- Small Batch Performance: With a batch of 1,000 transactions, latency is lower (2.39 sec), but throughput is just 4,594 tx/sec.
- Large Batch Impact: Increasing the batch size to 30,000 transactions results in a higher throughput of 37,620 tx/sec but latency rises to 8.77 sec.

Liveness relies on heavy cryptography or degraded efficiency

- Censorship Threat: Adversarial networks can delay transactions, threatening the liveness and fairness of the protocol.
- Current Mitigations:
  - Strategies like transaction diffusion and de-duplication are used but increase communication complexity.
  - TPKE Solution: HBBFT employs encryption to hide transactions, preventing targeted delays but raising decryption costs.
- DAG-Rider's Approach: Continuous listening for delayed broadcasts offers censorship resistance at the risk of unbounded memory usage.



1. Introduction

1.2 Our(the paper's) contribution

The paper answers the above challenges with **Dumbo-NG**—a direct, concise and efficient reduction from asynchronous BFT atomic broadcast to multi-valued validated Byzantine agreement (MVBA).

- Resolve the latency-throughput tension
  - By running the TX Dissemination and agreement stage together

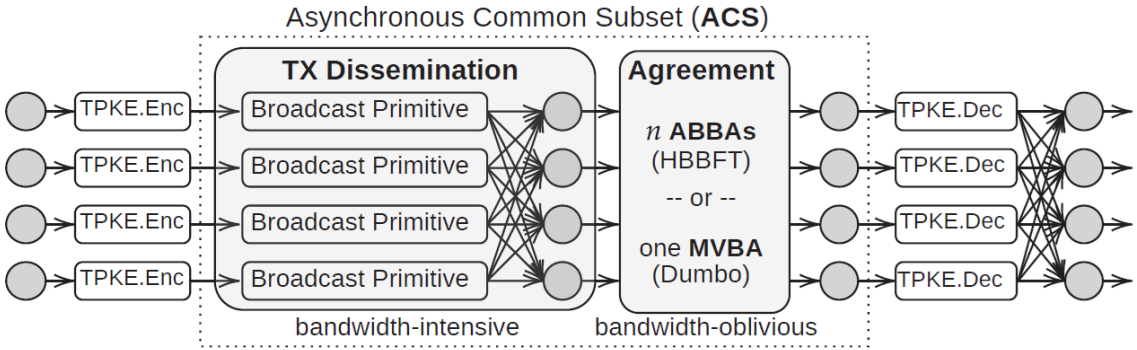


Figure 1: Execution flow of an epoch in HBBFT, Dumbo **and their variants. The protocols proceed by consecutive epochs.**

Prevent censorship with minimal cost

DAG-Rider and DispersedLedger, Dumbo-NG ensures that any transaction input by an honest node can eventually output (**strong validity)**

Table 1: Validity (liveness) of asynchronous atomic broadcast if stressing on nearly *linear* amortized communication

	Strong validity (Definition 4.1) ?	Memory-bounded implementation?
DAG-Rider [52], DispersedLedger [75], and Aleph [39]	✓ *	○ †
Tusk [33]	✗ suboptimal comm.; or after GST ‡	✓
HBBFT [62], Dumbo [49] and variants [36, 48]	✓ diffuse TX + TPKE for de-duplication	✓
Dumbo-NG (this paper)	✓ *	✓

By-product of adapting DispersedLedger to the state-of-the-art

From DispersedLedger to sDumbo-DL

+ From sDumbo: fast termination of the ACS design.
+ From Dumbo-MVBA: provable dispersal is adopted to replace AVID

Result in hurting latency less while realizing maximum throughput

Table 2: In comparison with existing performant asynchronous consensuses in the WAN setting at n=4, 16, 64 nodes.

Protocol	Peak throughput (tps)†			Latency@peak-tps(sec)		
	n=4	n=16	n=64	n=4	n=16	n=64
Dumbo [49]	22,038	40,943	28,747	11.85	13.43	29.91
Speeding Dumbo [48]	38,545	74,601	43,284	4.86	7.37	15.45
DispersedLedger‡[75]	54,868	92,402	33,049	3.09	5.05	7.03
Dumbo-NG	166,907	165,081	97,173	1.89	1.97	6.99

Implementation and extensive experiments over the Internet

The author implement Dumbo-NG and extensively test it among n = 4,16 or 64 nodes over the global Internet

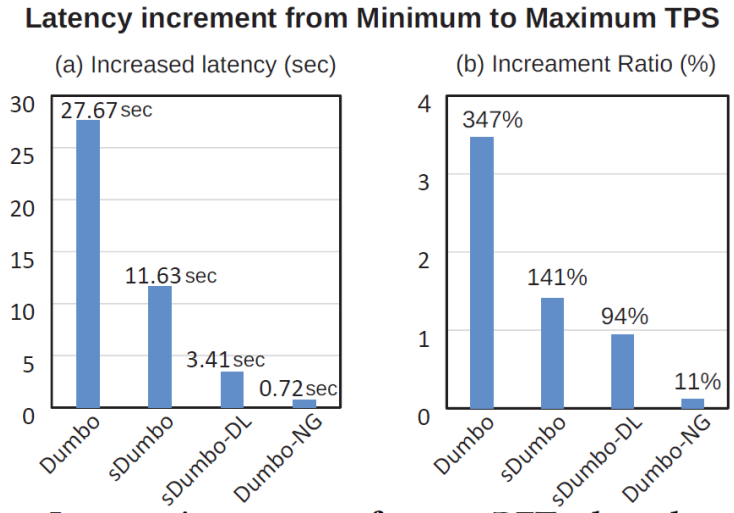


Figure 3: Latency increment of async. BFT when throughput increases from minimum to maximum in the WAN setting for n=64 nodes (cf. Section 7 for detailed experiment setup).

2. PATH TO OUR SOLUTION

DispersedLedger(DL) -> HBBFT

Problem -> Bandwidth

DL's idea -> separate the bandwidth-intensive transaction dissemination and the bandwidth-oblivious agreement

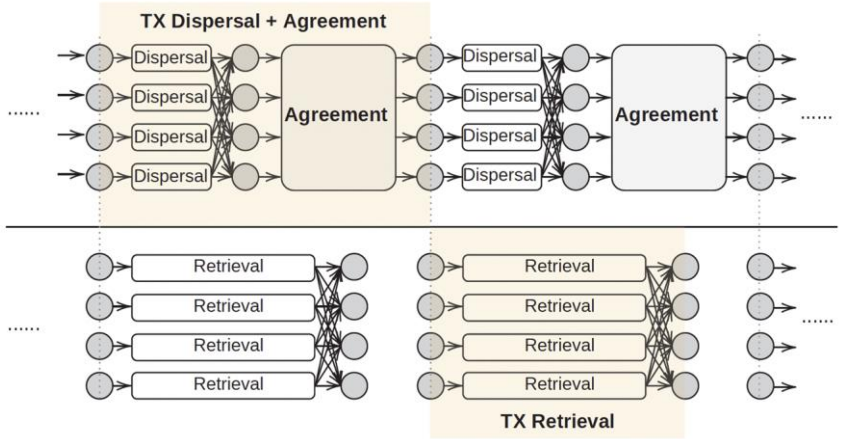


Figure 4: Execution flow of DispersedLedger. **Modules in the light-orange region represents one** ACS.

DL:

☑ realize considerable improvement in throughput

✗ cannot preserve a stable latency while approaching the maximum throughput

2. PATH TO OUR SOLUTION

Aim: to make the broadcast and the agreement to execute fully concurrently.

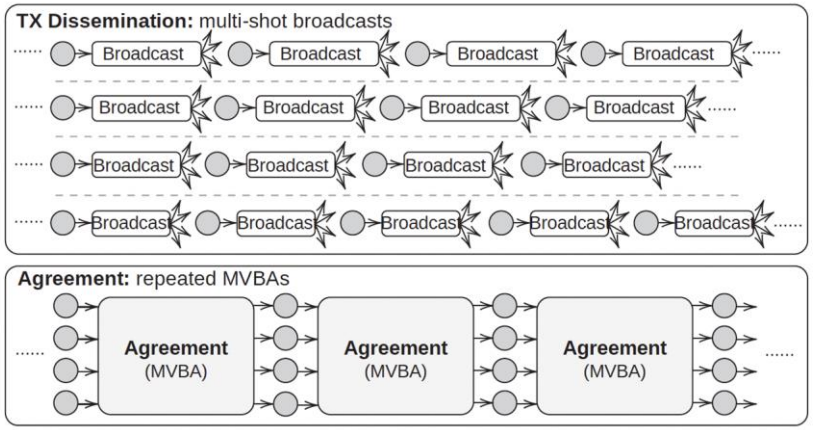


Figure 5: High-level of Dumbo-NG. **Each node leads an ever-running multi-shot broadcast. Concurrently, a sequence of** MVBAs **are executed to finalize the broadcasted transactions.**

Challenge I: allow BA to pack broadcasts, concurrently & validly

- Some fast, some slow
- Solution: add quorum certificates

Challenge II: output all completed broadcasts to prevent censorship

- MVBA cannot ensure liveness
- Solution: choose an MVBA protocol with quality

Tusk's transaction diffuse: does not generate quorum certificates
Our techniques: every node(even the slowest) can generate certificates

2. PATH TO OUR SOLUTION

An improved version of DL: sDumbo-DL->apply DL to sDumbo

Improvement:

- Replace the agreement phase of n concurrent ABBA instances by one single MVBA (instantiated by the state-of the-art GLL+22-MVBA [48]);
- Replace AVID-M by a weakened information dispersal primitive without totality (provable dispersal, PD)

Table 3: Complexities per ACS in DL and sDumbo-DL; $|B|$ is the size of each node' input, and λ is security parameter.

Protocol	Complexities of each ACS		
	Round	Communication †	Message
DispersedLedger	$O(\log n)$	$O(\|B\|n^2 + \lambda n^3 \log n)$	$O(n^3)$
sDumbo-DL	$O(1)$	$O(\|B\|n^2 + \lambda n^3 \log n)$	$O(n^2)$ ‡

2. PATH TO OUR SOLUTION

An improved version of DL: sDumbo-DL->apply DL to sDumbo

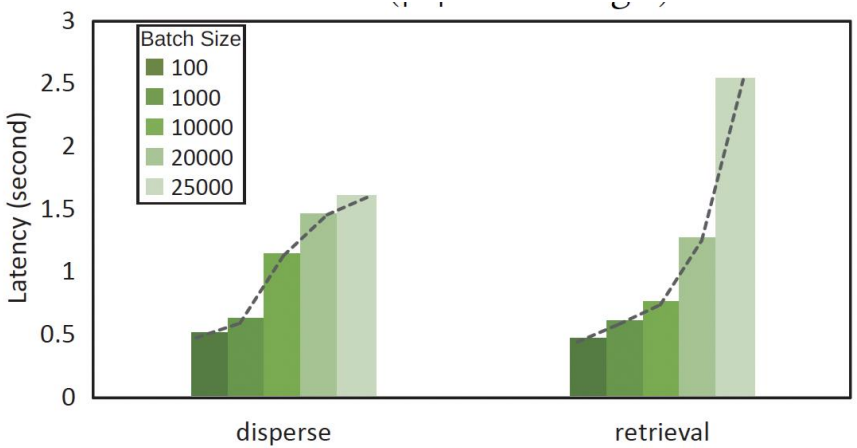


Figure 6: Latency of dispersal/retrieval as batch size grows.

When batch size |B| increases,

- The communication cost of provable dispersal PD quickly grows up
- The cost of retrieval might soar even more dynamically

-> its latency remains to be dramatically sacrificed while approaching larger throughput

3. OTHER RELATED WORK

- A large number of “robust” (partially) synchronous protocols such as Prime[5], Spinning [74], RBFT [8] and many others [30, 31] are subject to robustness-latency trade-off, and none of them can have guaranteed liveness in a pure asynchronous network, inherently.

- A few recent results [3, 66, 72] make synchronous protocols to attain fast (responsive) confirmation in certain good cases, but still suffer from slow confirmation in more general cases.

4. PROBLEM DEFINITION

System Model

- Known identities & setup for threshold signature
- n/3 Byzantine corruptions
- Asynchronous fully meshed P2P network

Security Goal: Atomic Broadcast (ABC) Abstraction

- Agreement: if one honest node outputs a transaction, so should every honest node
- Total-order: if any two honest nodes outputs two sequence of transactions, then the i-th transaction should be the same
- Liveness: if a transaction is input by any honest node, it will eventually output.

Performance metrics

- Message complexity
- Communication complexity
- Round complexity

6. Dumbo-NG

The Issue

the **agreement modules block the succeeding transaction** dissemination, so **huge batches** are necessary there to utilize most bandwidth **for maximum throughput**, which **unavoidably hurts the latency**

The Solution

With the **support concurrent processes** for bandwidth-intensive transaction dissemination and bandwidth-oblivious BA modules, Dumbo-NG can **use much smaller batches** to seize most bandwidth for realizing peak throughput

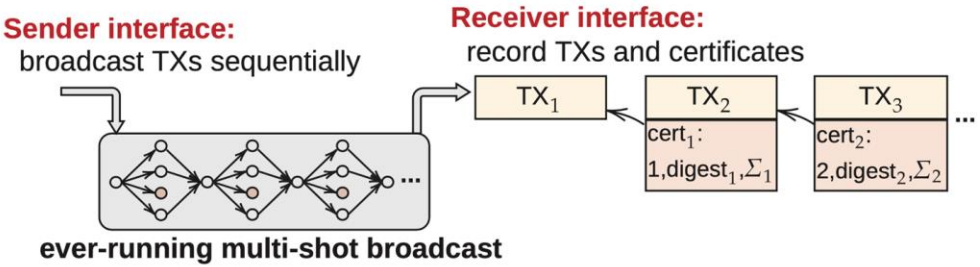


Figure 7: Ever-growing multi-shot broadcast.

6. Dumbo-NG - An Overview

N Ever-Running Broadcasts

- The broadcast is never blocked to wait for any agreement modules or other nodes' broadcasts.
- It just proceeds by consecutive slots at its own speed
- In each slot,
  - The broadcast delivers a batch of transactions along with a quorum certificate
  - A valid certificate delivered in some slot can prove:
    - At least f+1 honest nodes have received the same transactions in all previous slots of the broadcast
- The multi shot broadcast is implementable, since
  1. Each node only maintains several local variables
  2. All earlier delivered transactions can be thrown into persistent storage to wait for the final output

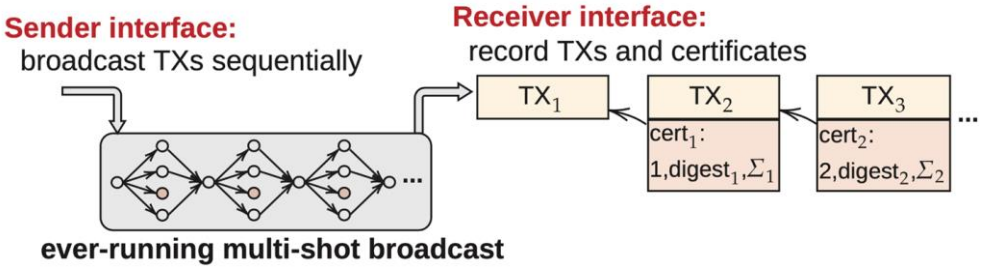


Figure 7: Ever-growing multi-shot broadcast.

6. Dumbo-NG - An Overview

A Sequence of BAs

- Concurrently execute MVBA with fine-tuned external validity condition
  - with certificates added to ever-running broadcasts
- A node will invoke an MVBA protocol if
  - n-f distinct broadcasts deliver new transactions to it (and new certificates as well)
  - It can take the n-f certificates as MVBA input
- The MVBA's external validity will check
  - 1. All certificates' validity
  - 2. These n-f indeed correspond to some newly delivered transactions that were not agreed to output before
- Once MVBA returns, all honest nodes receive a list of n-f valid certificates, and pack the transactions certified by these certificates as a block of consensus output.

6. Dumbo-NG - Workflow

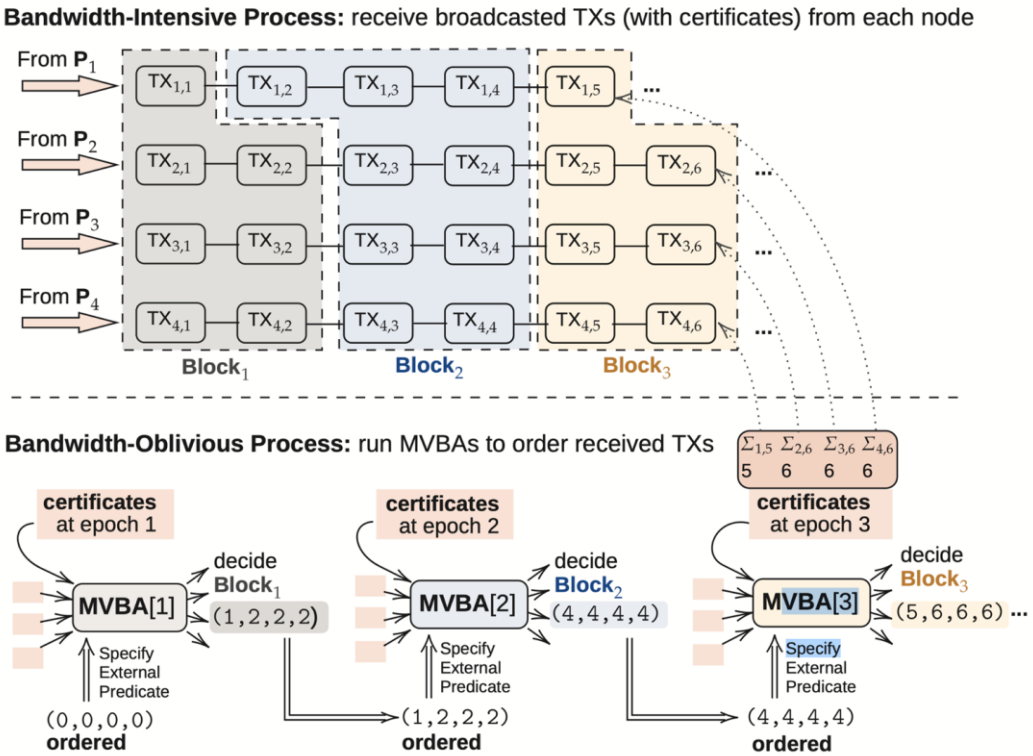


Figure 8: Illustration on how to totally order the received broadcasts through executing a sequence of MVBAs.

Broadcast

- As senders
- As Receivers

Agreement

- Concurrently executing and totally ordering the transaction batches
- Maintaining a vector at each epoch to denote ordered transaction batches.

7. IMPLEMENTATION AND EVALUATIONS

7.1 Implementation &WAN experiment setup

- Implement Dumbo-NG, sDumbo, Dumbo, and DispersedLedger in Python3
- The same cryptographic libraries and security parameters are used throughout all implementations.
- Both Dumbo-NG and sDumbo-DL are implemented as two-process Python programs.
- Use gevent library -> concurrent tasks in one process.
- Coin flipping -> Boldyreva's pairing-based threshold signature
- Quorum certificates -> concatenate ECDSA signatures

Implementation of asynchronous network: a (persistent) unauthenticated TCP connection between every two nodes

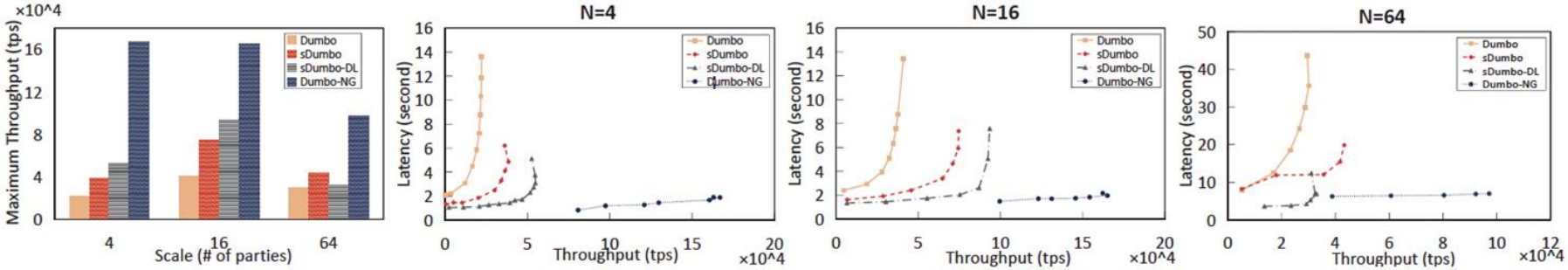
Setup on Amazon EC2: n = 4, 16, and 64 nodes

7. IMPLEMENTATION AND EVALUATIONS

7.2 Evaluation results in the WAN setting

Dumbo-NG v.s. prior art.

- Peak throughput
- Latency-throughput trade-off



(a) Peak throughput (b) Latency-TPS trade-off ($n = 4$) (c) Latency-TPS trade-off ($n = 16$) (d) Latency-TPS trade-off ($n = 64$)

Figure 9: Performance of Dumbo-NG in comparison with the state-of-the-art asynchronous protocols (in the WAN setting).

7. IMPLEMENTATION AND EVALUATIONS

7.2 Evaluation results in the WAN setting

Dumbo-NG v.s. prior art.

- Latency/Throughput while varying batch sizes
- Throughput-oblivious metric

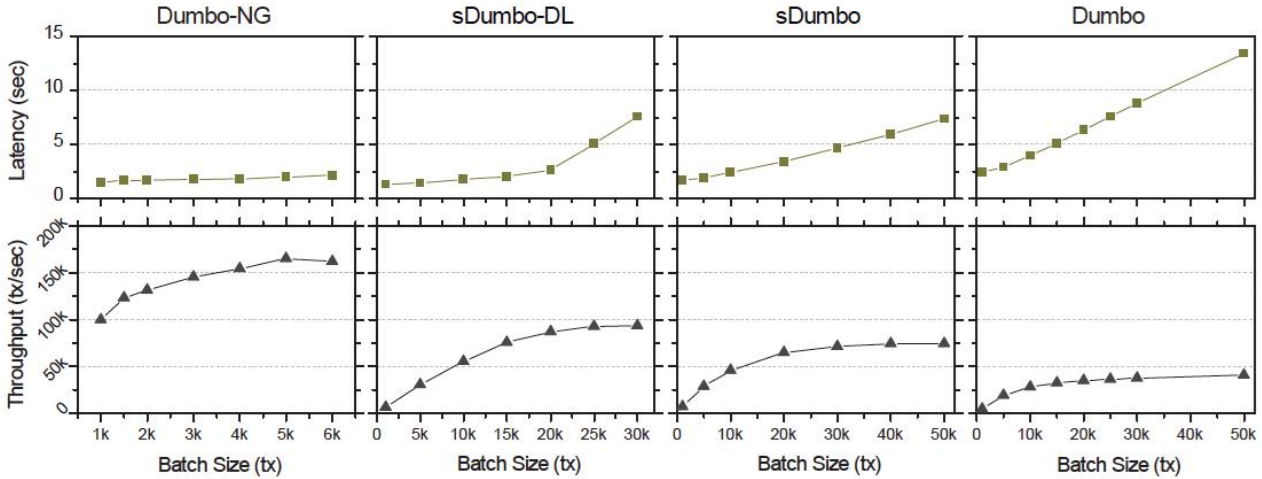


Figure 10: Throughput/latency of Dumbo-NG, sDumbo-DL, sDumbo **and** Dumbo **in varying batch size for WAN setting** ($n = 16$).

7. IMPLEMENTATION AND EVALUATIONS

7.3 More tests with controlled delay/bandwidth

Evaluate MVBA with controlled network bandwidth/delay.

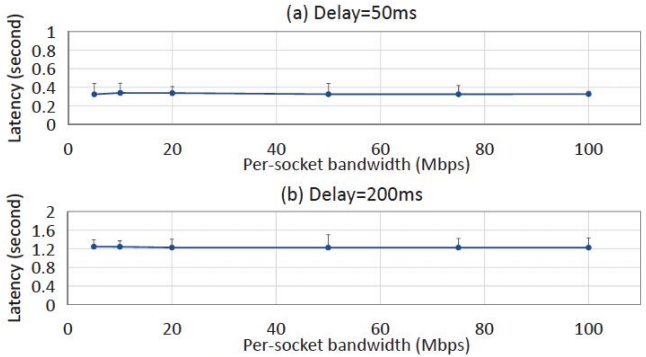


Figure 11: Latency of GLL+22-MVBA (n=16) in varying bandwidth for (a) 50ms and (b) 200ms network delay, respectively.

Test Dumbo-NG with controlled network bandwidth/delay

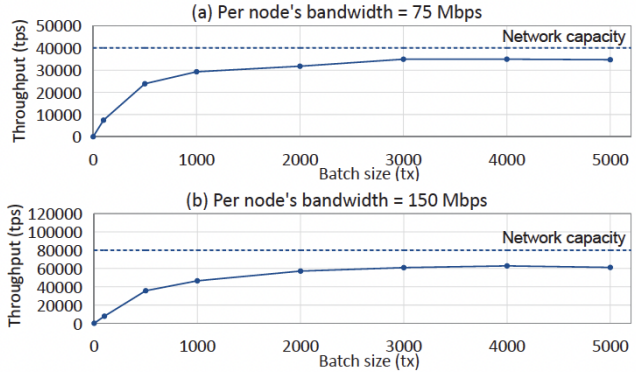


Figure 12: The dependency of throughput on varying batch size in controlled deployment environment with 50 ms one-way delay and (a) 75Mbps and (b) 150Mbps bandwidth.

7. IMPLEMENTATION AND EVALUATIONS

7.4 Behind the throughput-oblivious latency

*Why the latency of Dumbo-NGis almost **independent** to its throughput?*

Reasons:

- Small batch sizes already fully utilize network bandwidth -> therefore the latency of broadcasting a transaction batch is much smaller than that of MVBA
- When the broadcasts seize most bandwidth, the latency of MVBA is nearly not impacted as it is bandwidth-oblivious.

8. DISCUSSIONS + 9. CONCLUSION

- Production-level implementation
- From validity to censorship resilience
  - Strong validity
  - Validity
  - Weak validity

Conclusions

➔ Present **Dumbo-NG** an efficient asynchronous BFT atomic broadcast protocol favoring censorship resilience

- Realizes **high** throughput, **low** latency and **guaranteed liveness** at the same time.

➔ Despite the recent progress of practical asynchronous BFT, problems remain open:

- It would be interesting to have versatile benchmarks to experimentally evaluate the actual robustness of asynchronous consensus protocols in the adversarial network environment
- Practical asynchronous BFT consensus with enhanced fairness guarantees is **largely unexplored**
- Most existing asynchronous BFT protocols **cannot smoothly scale up**