Attested Append-Only Memory: Making Adversaries Stick to their Word

Presented by Chuan He

Talk Outline

- Introduction and Motivation
- Attested Append-Only Memory (A2M)
- A2M Protocols
- Evaluation
- Conclusion

Motivation

- You want to build a service
  - Easy on a single machine
  - Replicate service on multiple machines

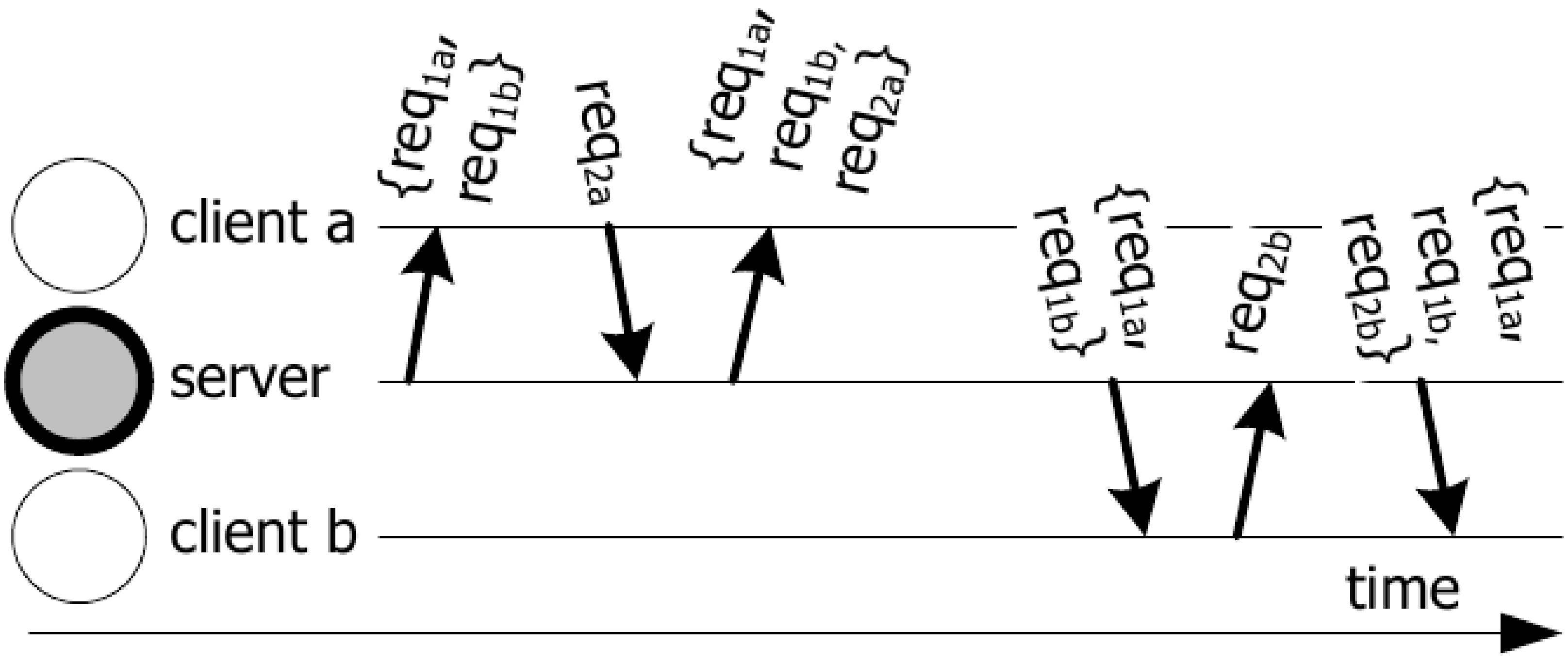
- Replicated services must appear as single server
  - Linearizability: Completed client requests appear to have been processed in a single, totally ordered, serial schedule consistent with the order they were submitted

Motivation

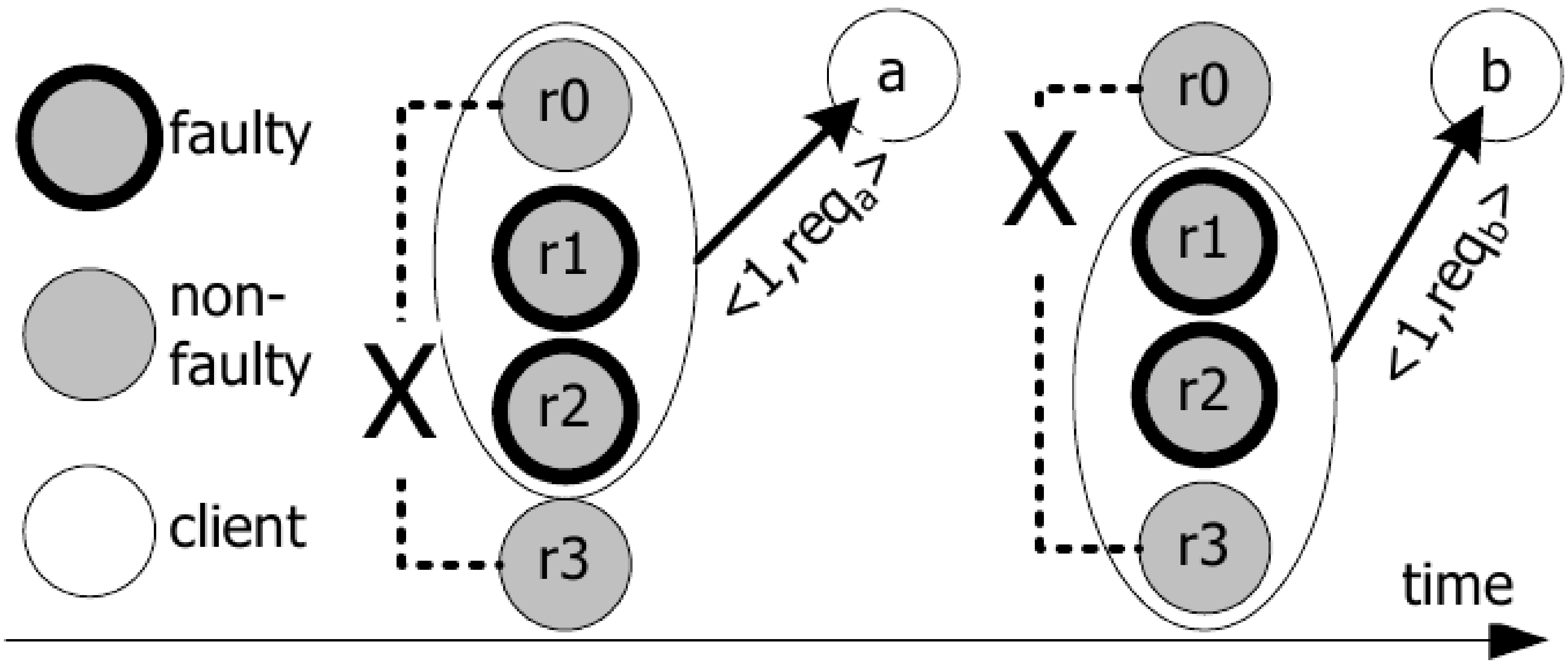
- You want to build a service
  - Easy on a single machine
  - Replicate service on multiple machines

- Replicated services must appear as single server
  - Equivocation: Different lies to different people

Servers Equivocating to Clients



Servers Equivocating to Servers



Questions

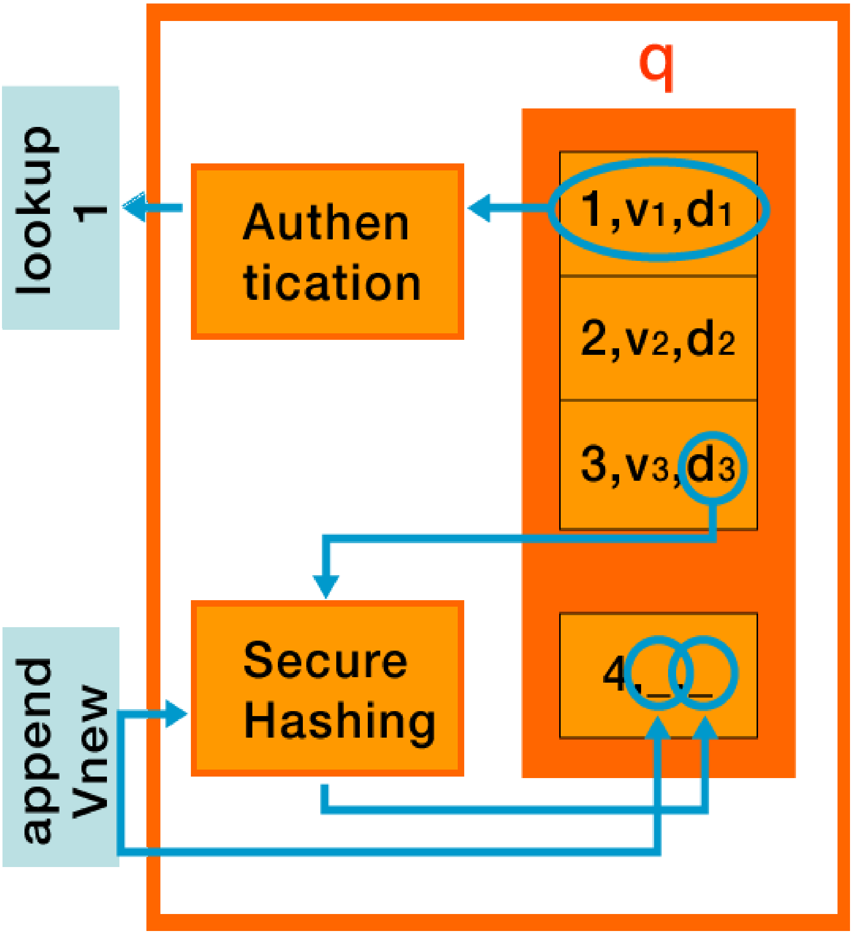
- Does preventing equivocation help at all?
  - Can we improve upon the 1/3 Byzantine fault bound?
- How do we prevent equivocation?
  - Is there any minimal system support?

Talk Outline

- Introduction and Motivation
- Attested Append-Only Memory (A2M)
- A2M Protocols
- Evaluation
- Conclusion

Attested Append-Only Memory (A2M)

- A set of numbered logs
- Each log entry contains
  - Sequence number
  - Stored value
  - Crypto digest
- `lookup / end`
  - Get a log entry
  - Attest (sequence number, value, history digest)
  - Attest freshness
  - Attest the end of log
- `append / advance`
  - Cannot overwrite

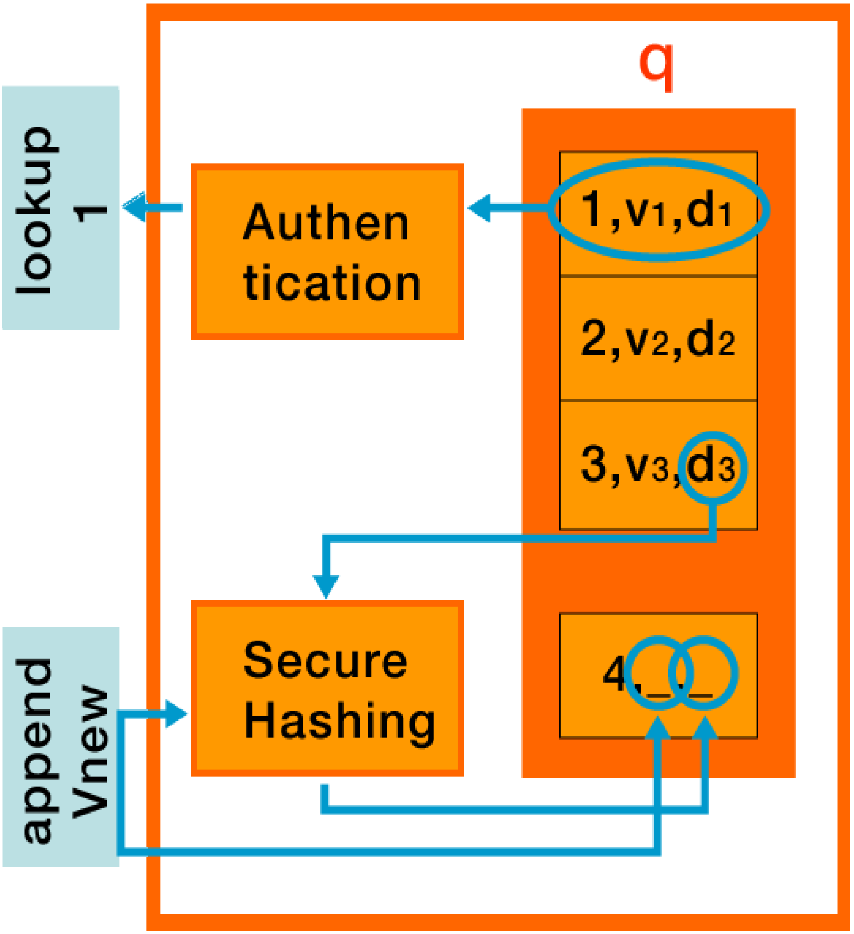


Attested Append-Only Memory (A2M)

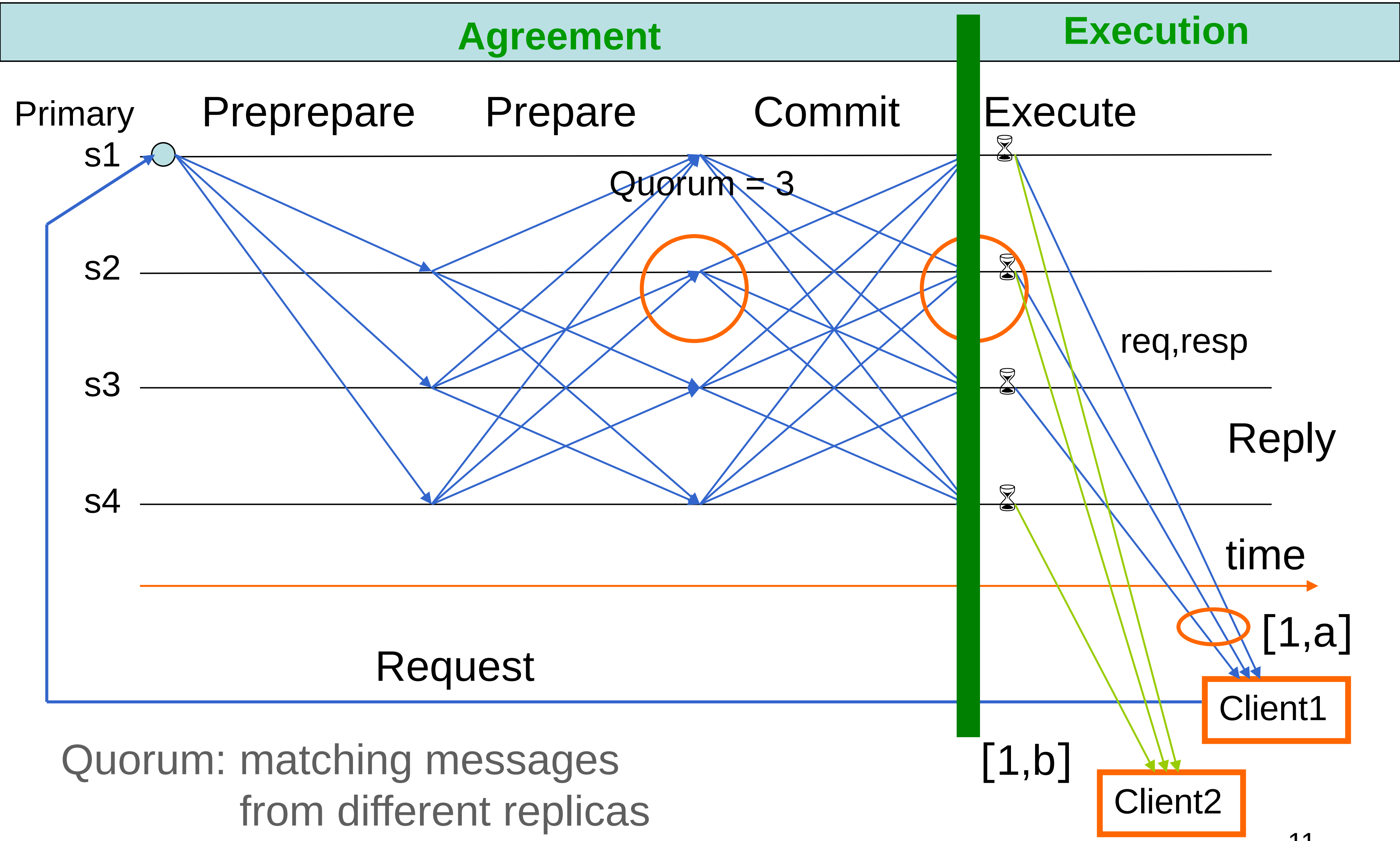
- `append / advance`

$$d_{\mathcal{H}} = h(\mathcal{H} \| x \| d_{\mathcal{H}-1})$$

- Important feature
  - Cannot equivocate

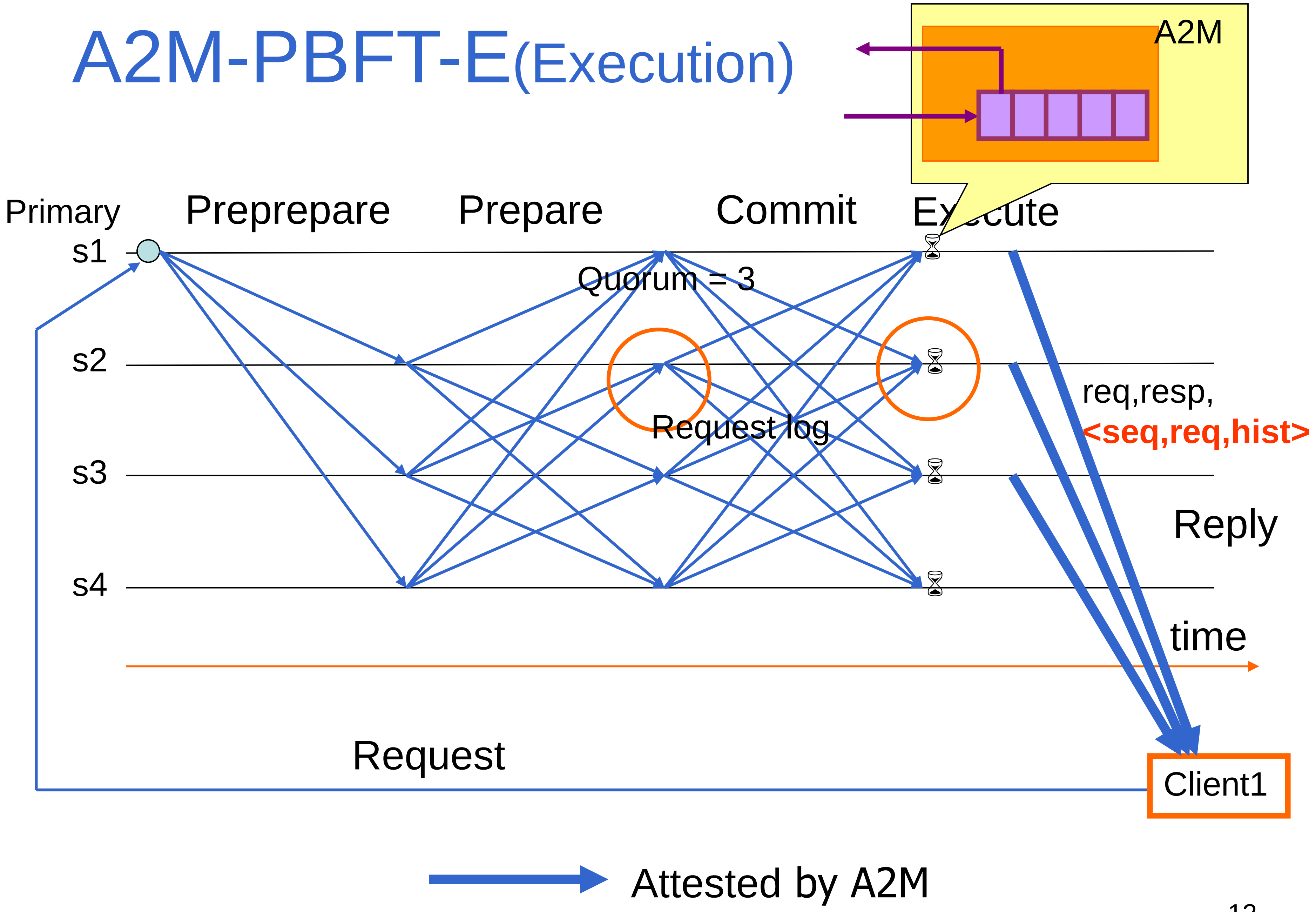


Background: PBFT

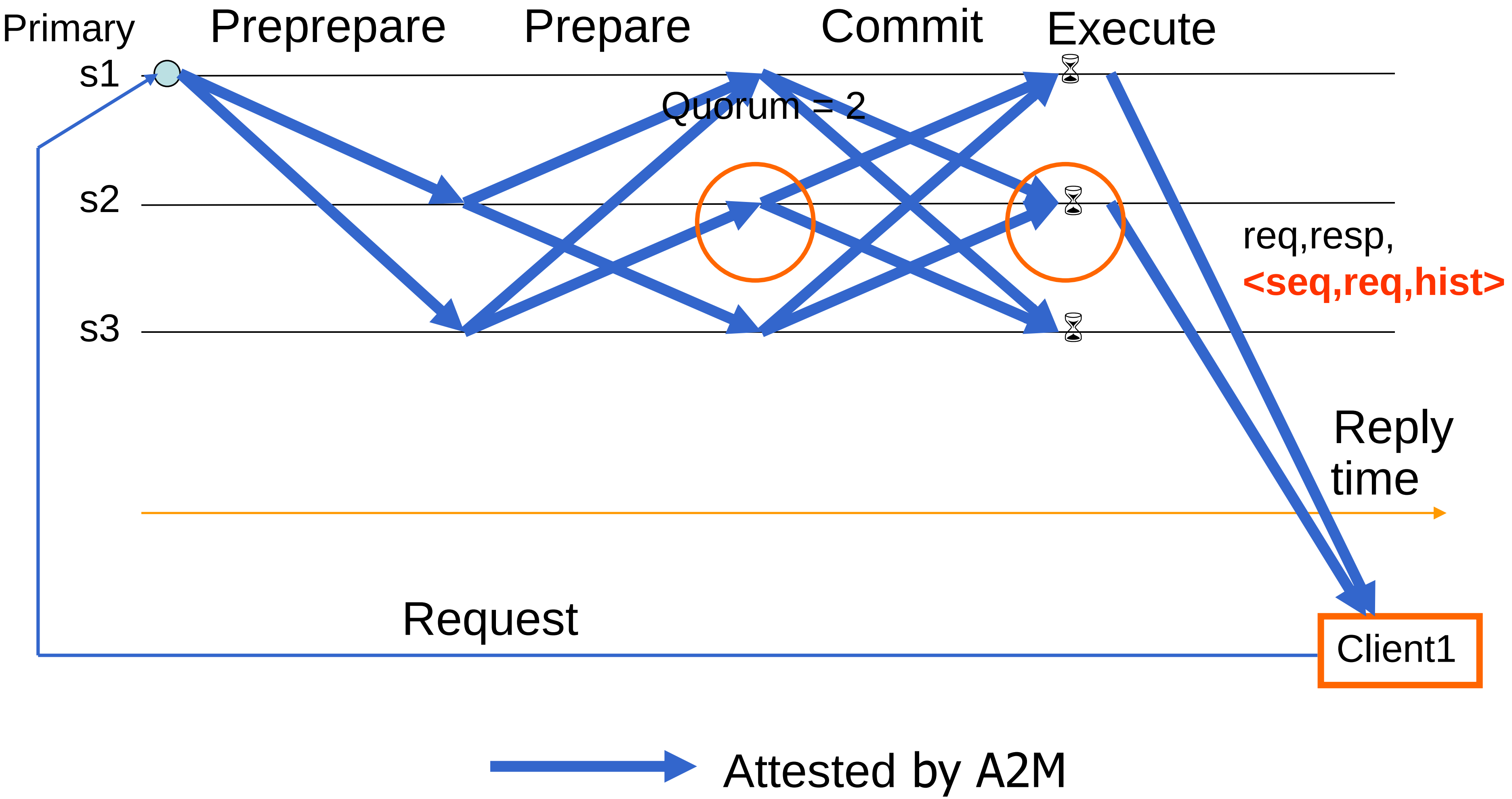


Quorum: matching messages
from different replicas

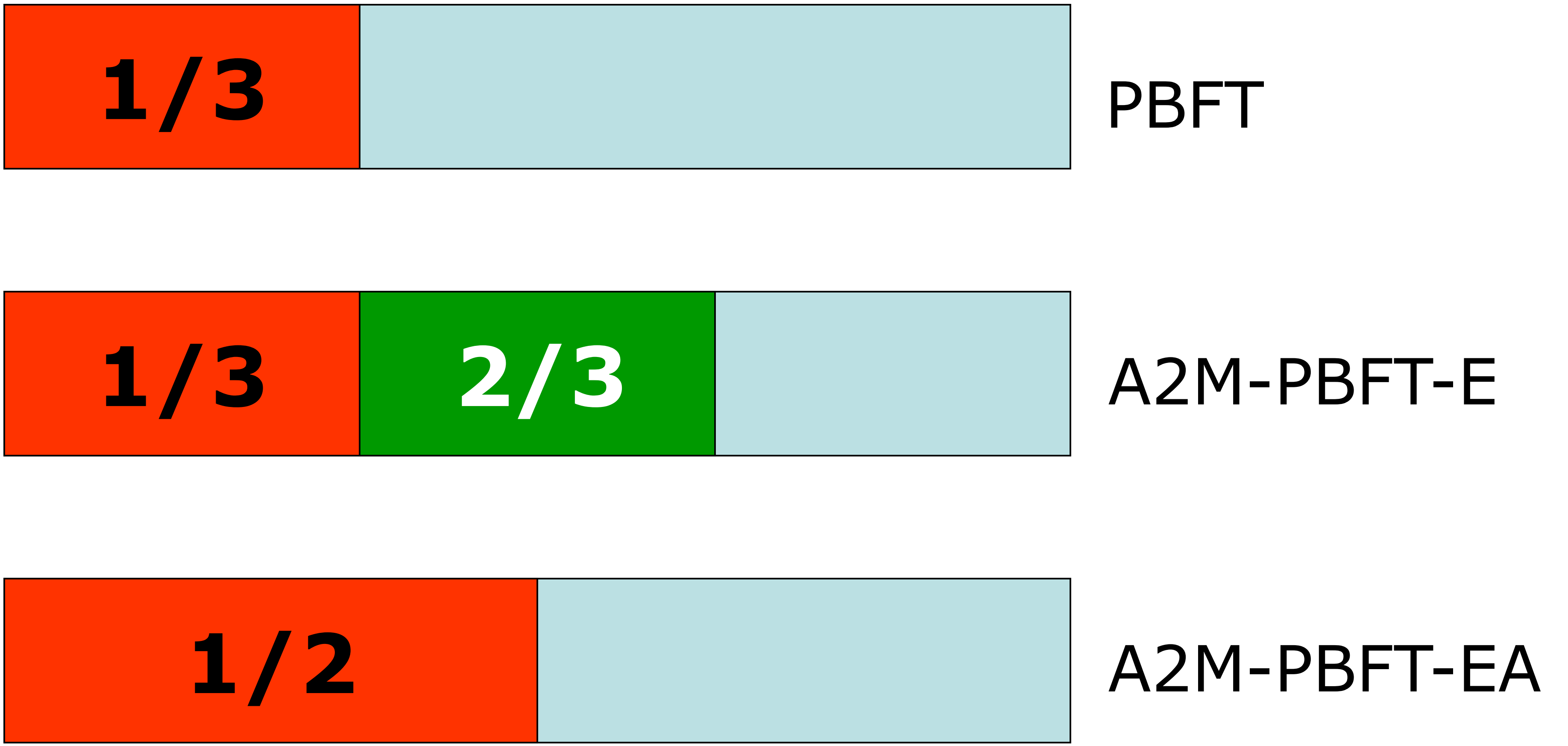
A2M-PBFT-E(Execution)



A2M-PBFT-EA (2f + 1 replicas)



Protocol Trade-offs



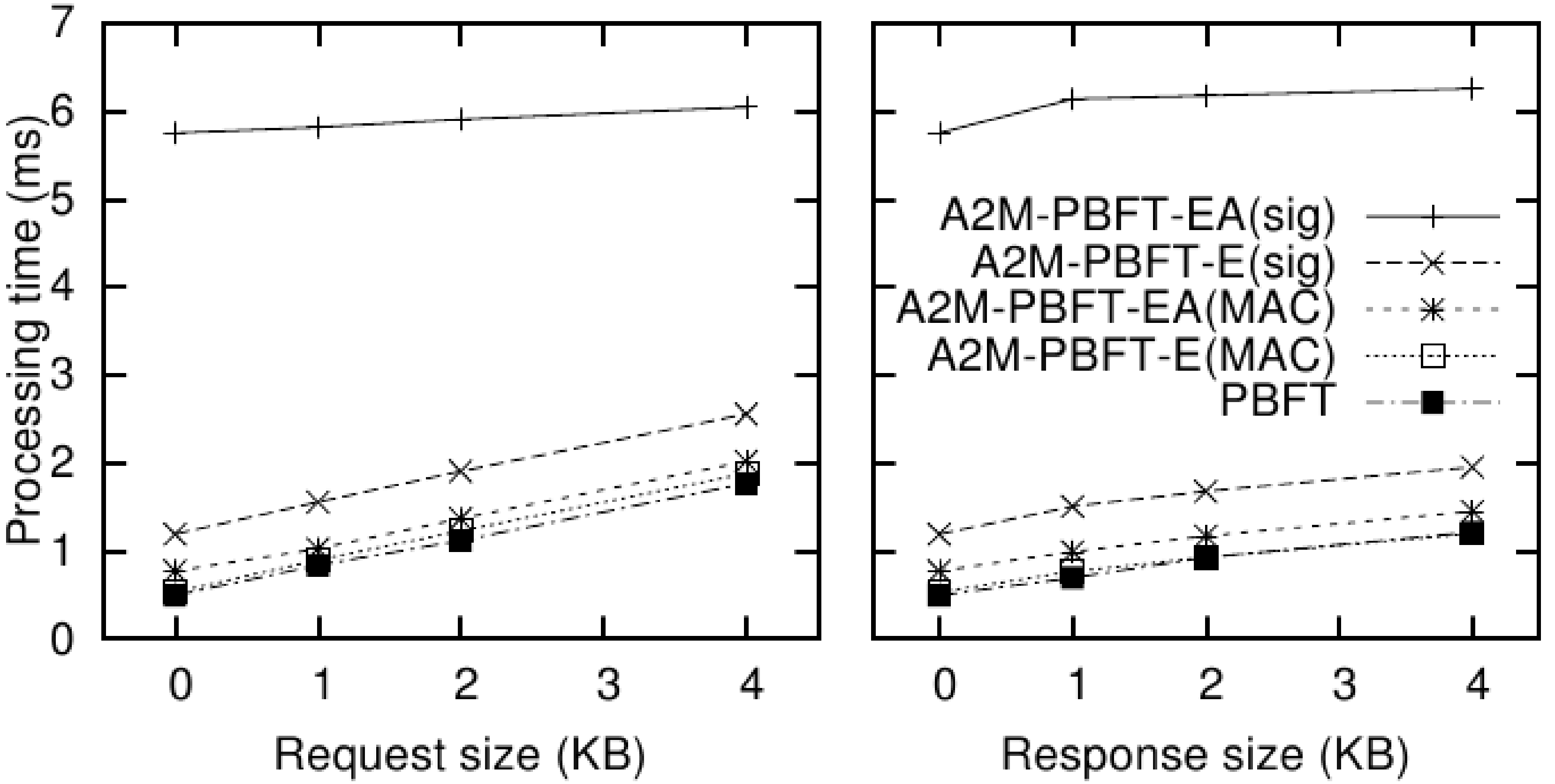
Evaluation Setup

- Implemented A2M-PBFT-E and A2M-PBFT-EA
- A2M protocols use signatures or MACs for authentication

- Four replicas in a LAN. Each replica has its own A2M.

- Microbenchmarks
  - Null operation with various request or response sizes
- Macrobenchmarks: NFS
  - Software package compilation

Evaluation - Microbenchmarks



Evaluation - Macrobenchmarks

NFS Phase	-S	-PBFT	-A2M-PBFT-E (sig)	-A2M-PBFT-E (MAC)	-A2M-PBFT-EA (sig)	-A2M-PBFT-EA (MAC)
Copy	0.219	0.709	1.026	0.728	2.141	0.763
Uncompress	1.015	3.027	4.378	3.103	8.601	3.236
Untar	2.322	4.448	6.826	4.553	12.896	4.669
Configure	12.748	12.412	19.173	12.659	26.181	13.040
Make	7.241	7.461	9.778	7.500	11.379	7.510
Clean	0.180	0.298	0.640	0.312	0.742	0.311
Total	23.725	28.355	41.821	28.854	61.940	29.528

Table 1: Mean time to complete the six macrobenchmark phases in seconds.

Varying delay time

Additional NFS-latency (μs)	A2M-PBFT-E (MAC)	A2M-PBFT-E (MAC) with batching	A2M-PBFT-EA (MAC)	A2M-PBFT-EA (MAC) with batching
1	28.854	28.763	29.528	29.505
10	29.598	29.025	31.299	30.188
50	32.735	30.232	36.242	32.214
250	48.784	37.237	66.441	45.199
1000	117.59	65.813	192.53	101.62

Table 2: Mean time to complete the six macrobenchmark phases in seconds for different A2M additional latency costs.

Conclusions

- Present A2M, a small trusted, log-based memory
  - Simple and easily implementable
  - Prevent equivocation
- Improve fault tolerance by forcing servers to commit to a single history of operations
  - Improve fault bounds of BFT state machine replication
  - Achieve linearizability in an untrusted single-server system
  - The benefits are achieved with small performance overhead

Thank you!

Related Work

- Weaken the guarantee
  - fork* consistency [NSDI07]
  - fork consistency [OSDI04]
- Standard trusted hardware like TPM
  - does not improve the fault bound
- Auditing
  - PeerReview [SOSP07], CATS [FAST07]
- Shared file servers
  - SUNDR[OSDI04], Ivy [OSDI02], Plutus[FAST03]
- Separating agreement from execution
- Symmetric faults – hybrid fault model
- Group communication