Classical Consensus

Possible Timing Models

Asynchronous

- A sent message will *eventually* be delivered

Synchronous

- Guarantees on the time of delivers
- e.g. message sent at time t, will be delivered at time t+x

Eventually Synchronous

- A mix between both, where there is a *known upper bound* on the delivered time which is *variable*.

Possible Fault Models

Up to **f out of N processes can fail**

- Typically f < N/2, f < N/3

Honest nodes

- Remain available and do not behave byzantine

Availability failure

- A node might suddenly crash/Internet connectivity drop

Byzantine failure

- Malicious failure of an adversary

Broadcast Models

Consistent Broadcast

- A corrupted sender implies that not every party might terminate/deliver a request.

Reliable Broadcast

- Sender emits value v
  - Termination: if sender honest, correct party outputs v
  - Reliability: every correct party outputs v
  - Consistency: two distinct parties output v1, v2 and v1=v2

https://lpd.epfl.ch/site/_media/education/sdc_byzconsensus.pdf

PBFT, Paxos, RAFT, et al.

Leader election

- Every node can become eventually a leader (e.g. round-robin)

Safety

- The output is guaranteed to be consistent, even under an unstable network and malicious leader
- Although asynchronous eventually synchronous

Liveness

- If no progress, new leader elected
- If network stable and honest leader then liveness

Known Impossibility Results

Fischer, Lynch, Patterson (1985)

- FLP result
- In a fully asynchronous system, there is no deterministic consensus solution that tolerates one or more failures
- No algorithm can always reach consensus in bounded time.

Implication

- Timing assumptions are required for every protocol
- Randomness is crucial

Given any protocol

Is the impossibility result respected?

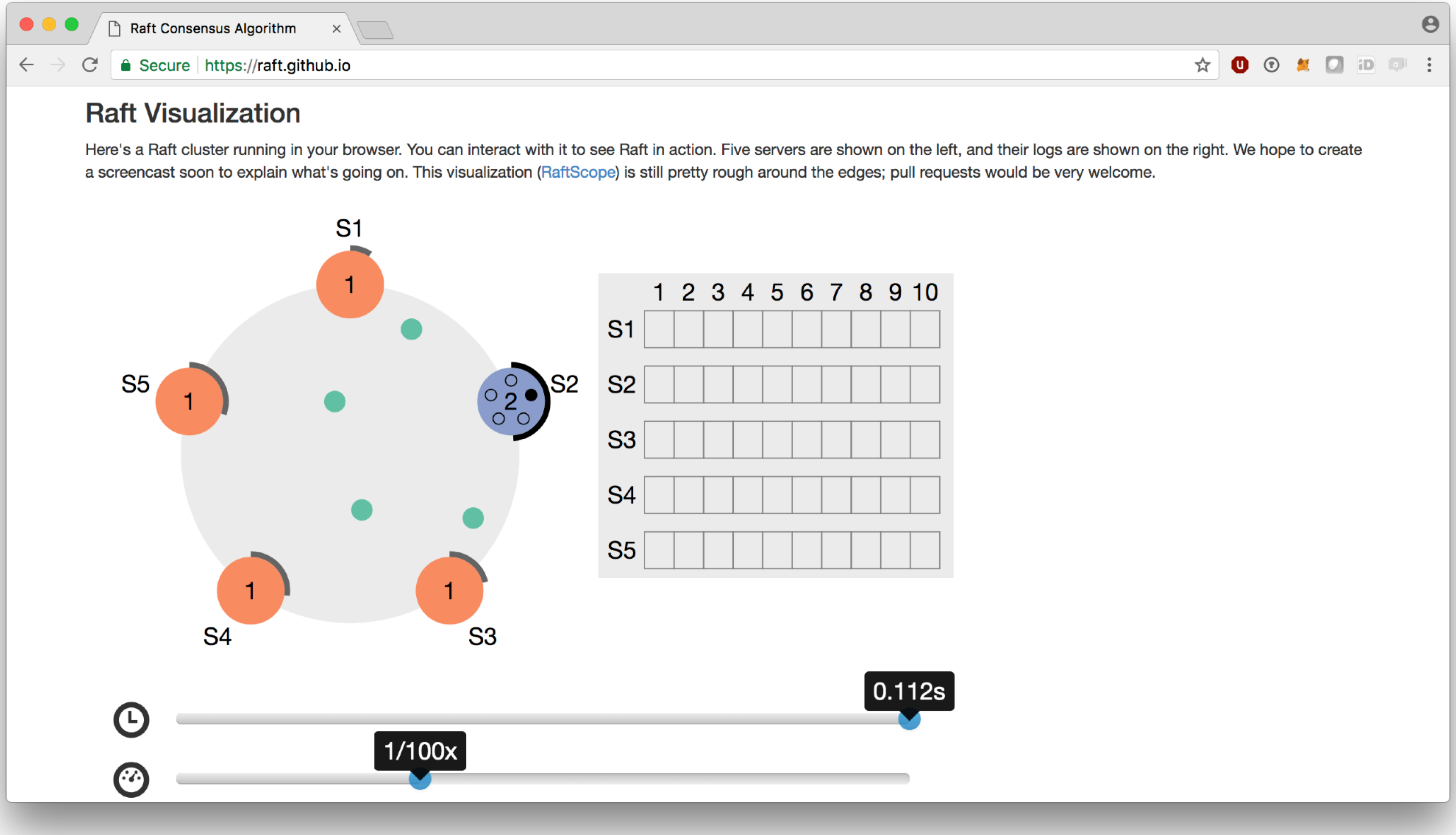
What's the message complexity/number of rounds?

How many nodes are allowed to fail?

Where does the randomness come from?

Can an adversary manipulate the randomness/become leader?

RAFT



Raft Consensus Algorithm

Secure https://raft.github.io

Raft Visualization

Here's a Raft cluster running in your browser. You can interact with it to see Raft in action. Five servers are shown on the left, and their logs are shown on the right. We hope to create a screencast soon to explain what's going on. This visualization (RaftScope) is still pretty rough around the edges; pull requests would be very welcome.

	1	2	3	4	5	6	7	8	9	10
S1										
S2										
S3										
S4										
S5										

http://thesecretlivesofdata.com/raft/ https://raft.github.io/

How does this relate to Bitcoin?

- No need for a final consensus output
- Block/transaction reward as incentive to participate
- **The participating nodes do not need to be known upfront!**

➡ Fundamentally different to the results of years of research