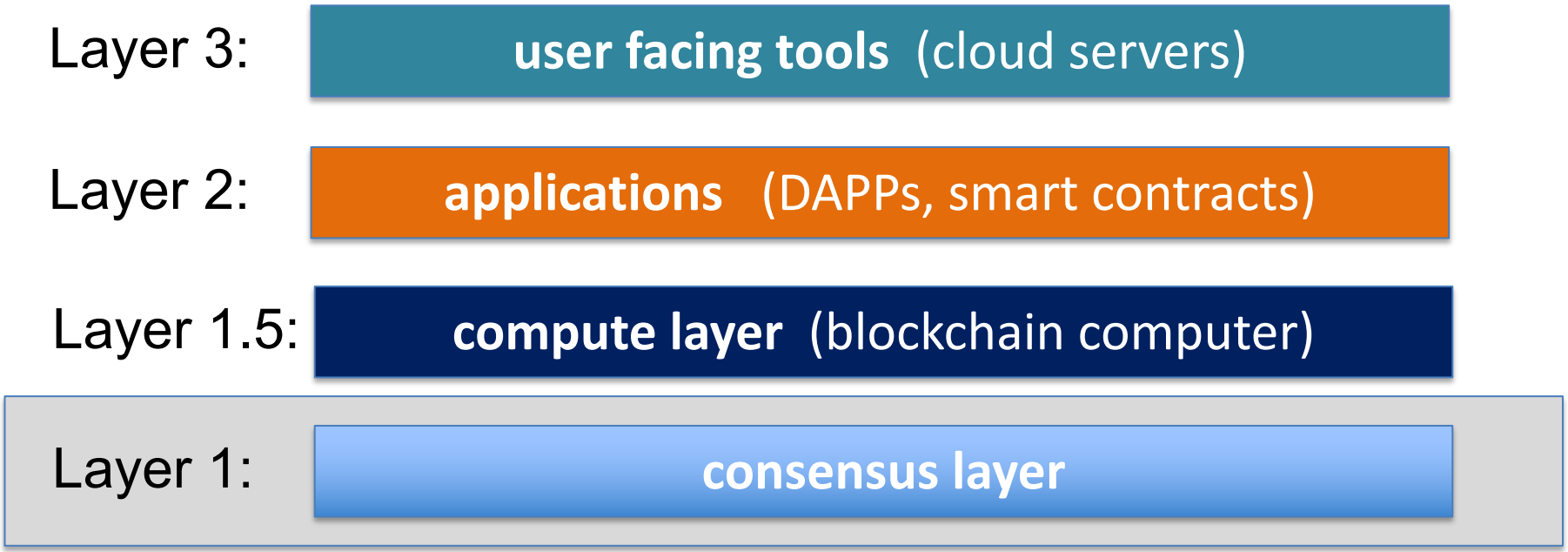
CS251 Fall 2020

(cs251.stanford.edu)

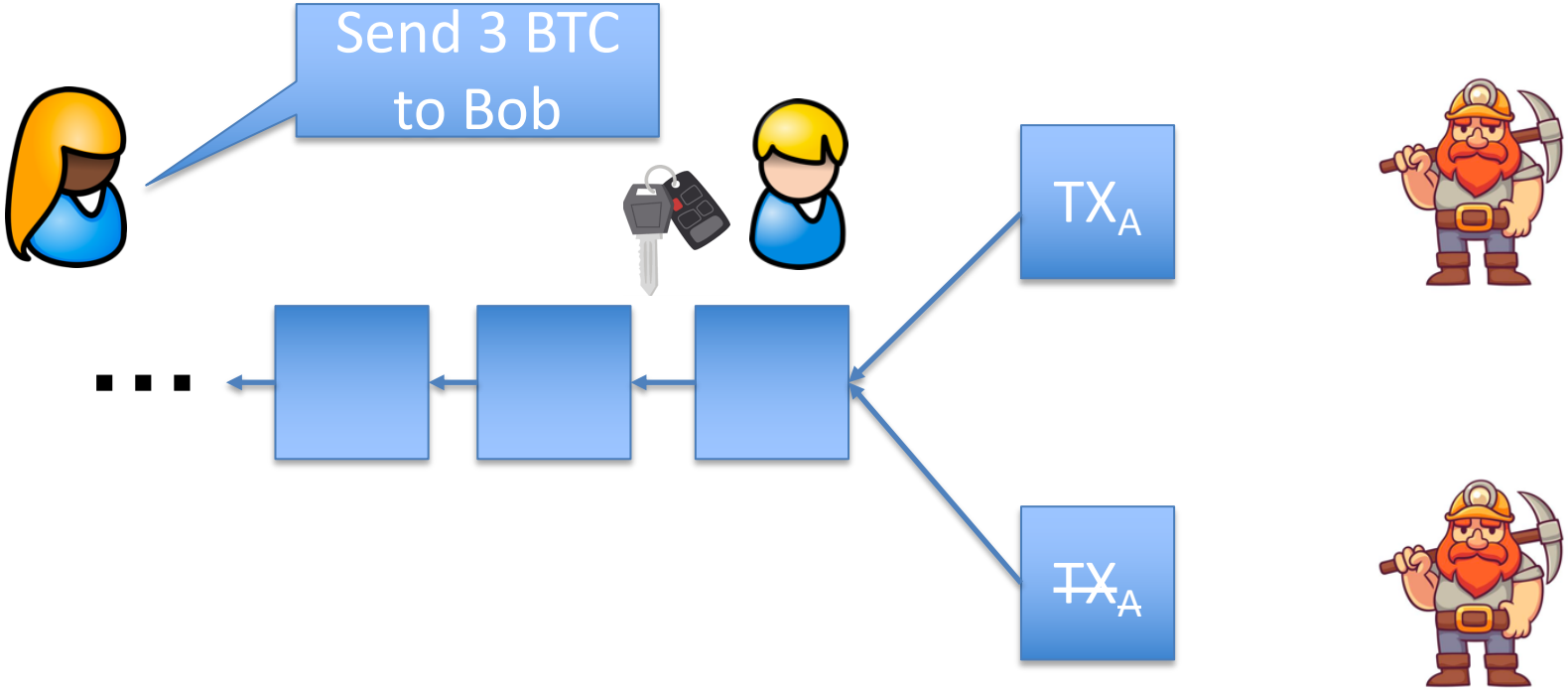
Classical Consensus

Benedikt Bünz

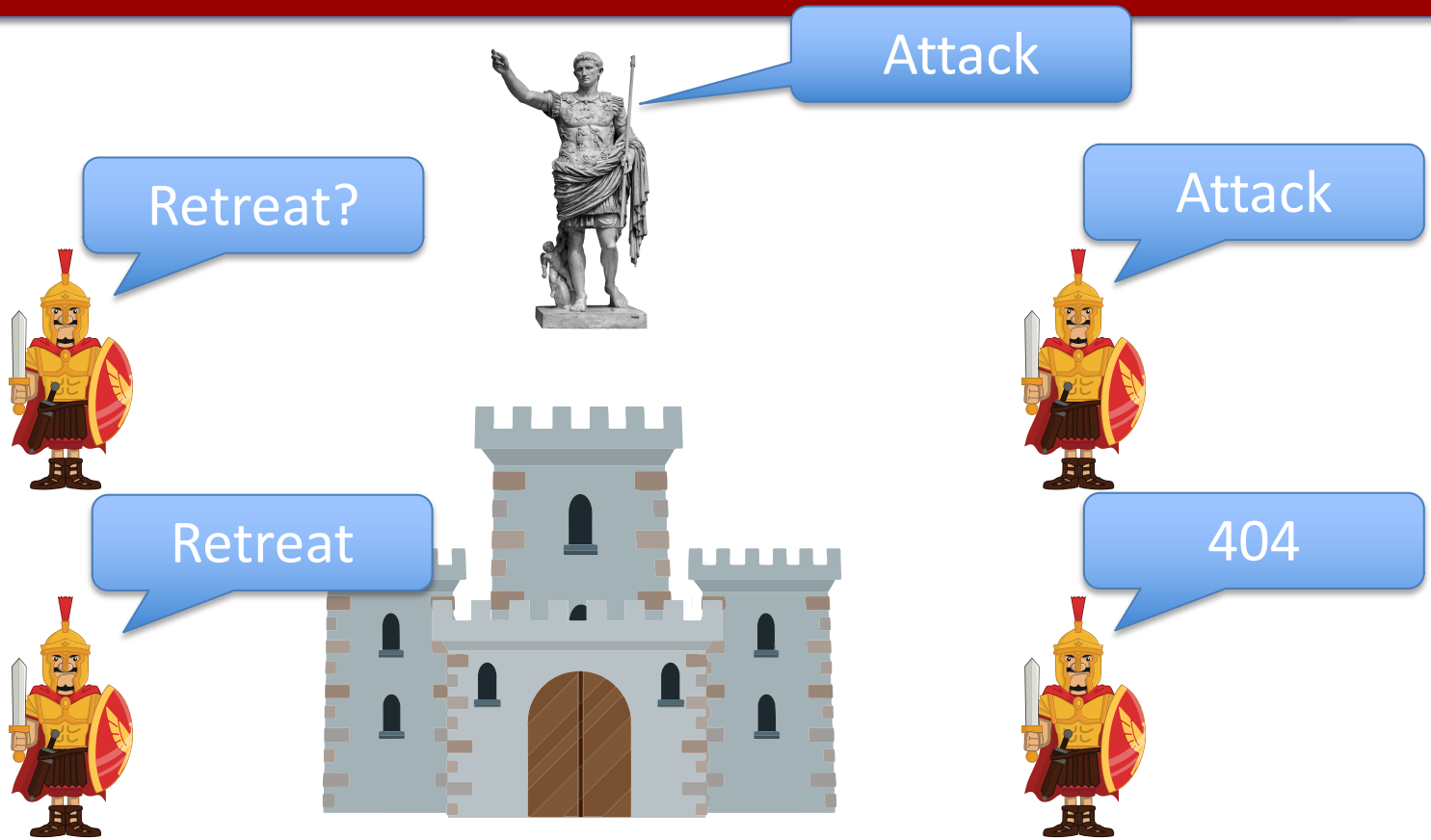
Blockchain Layers



Blockchain Forks



Byzantine Generals Problem

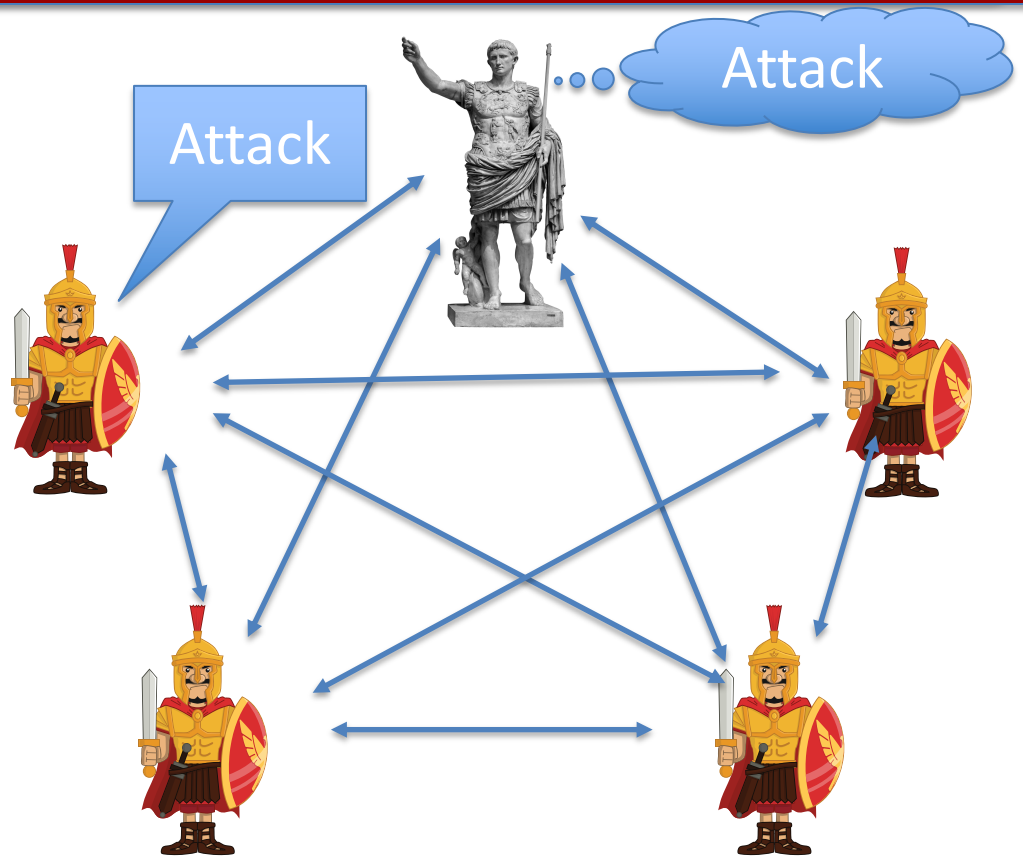


Byzantine Generals Problem

Leader gets an input bit 0/1

Every round each *node* sends messages to every other general. Messages are received in the next round

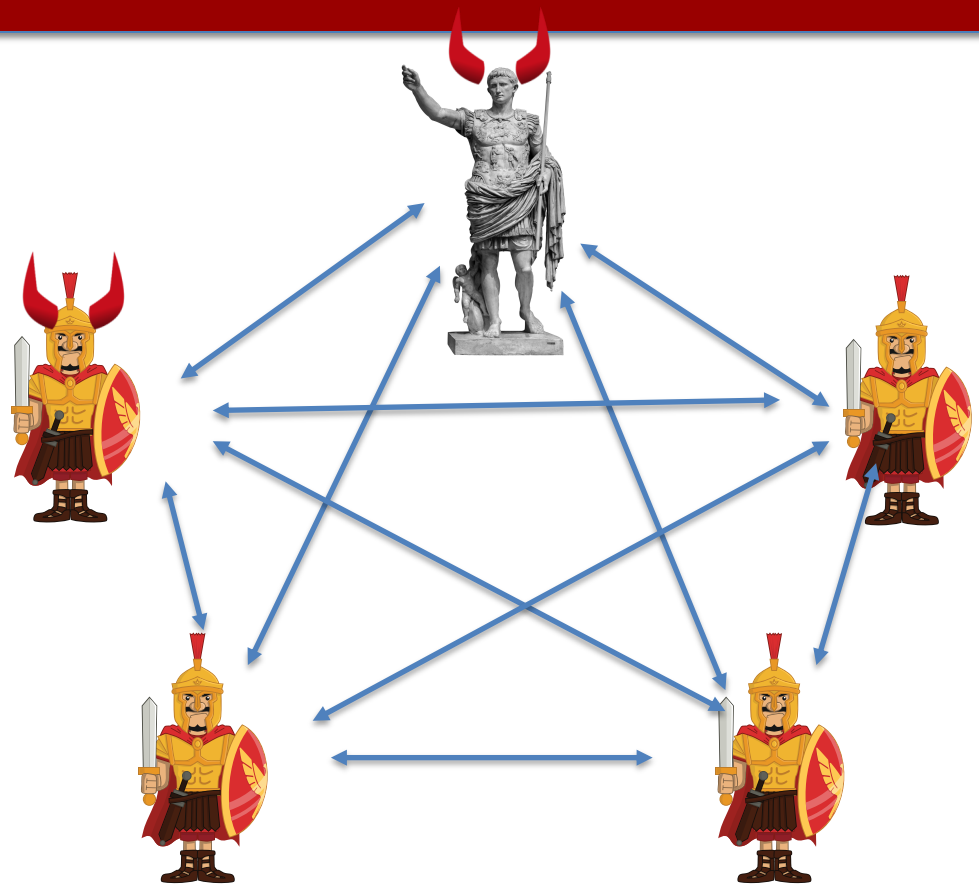
At the end of the protocol honest nodes output a bit or abort



Byzantine Generals Problem

Honest generals follow the protocol. Malicious generals behave arbitrarily

Assuming signatures



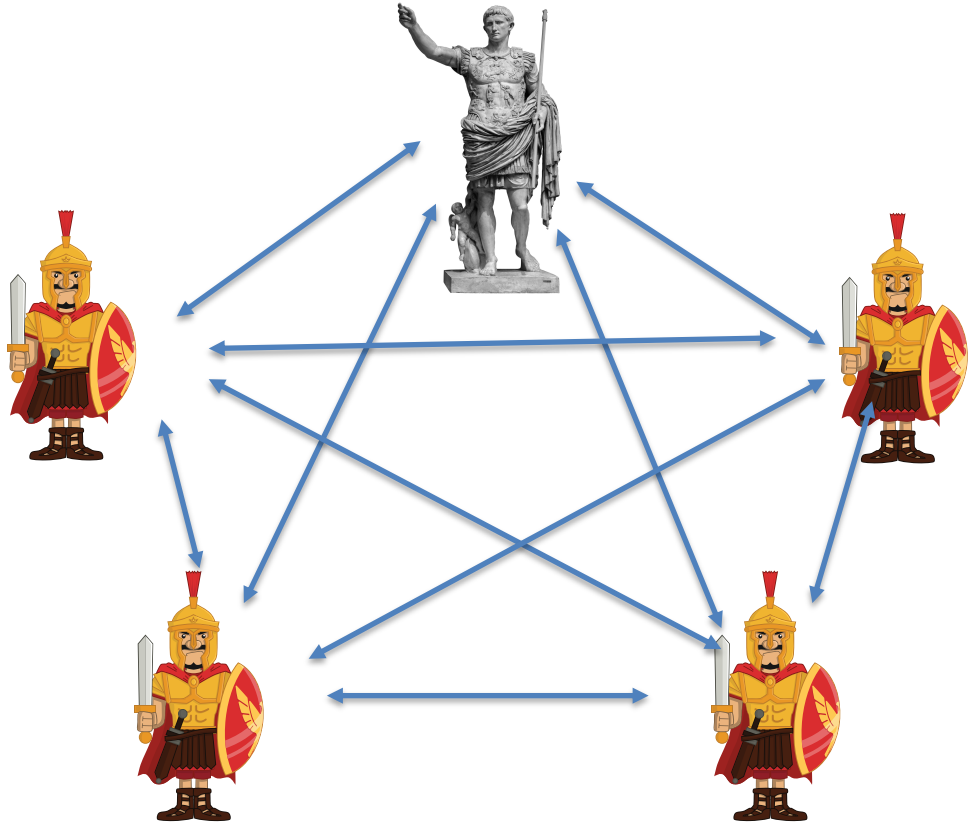
Byzantine Generals Problem

Consistency

If two honest nodes output b and b', respectively, then b=b'.

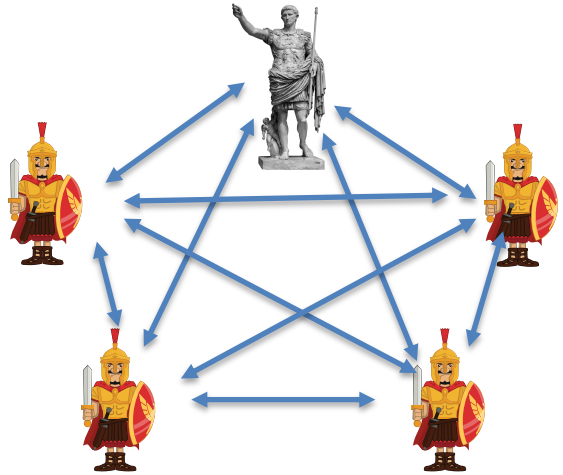
Validity

If the leader is honest and receives input b then all honest nodes output b



Voting Protocol

1. Leader sends b to all nodes
2. All nodes forward received bit to all other nodes (Voting)
3. Each node tallies votes (including its own vote) and outputs majority bit

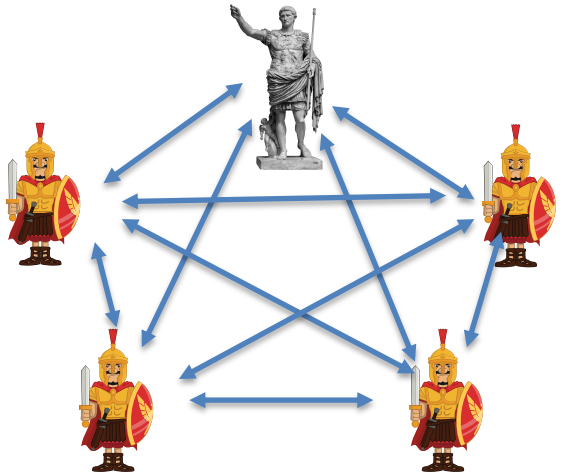


Broken by corrupt leader

Dolev Strong Protocol

Maximum $\boldsymbol{f}$ corrupt nodes, input message $\boldsymbol{m}$

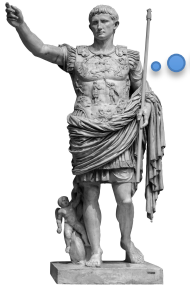
1. Leader sends $\boldsymbol{m}$ to all nodes
2. For $r = 1$ to $\boldsymbol{f} + 1$
   1. If you received an unseen message $\boldsymbol{m}$ signed by r signatures (including leader) sign $\boldsymbol{m}$ and send to all. Set $S \leftarrow S \cup \{\boldsymbol{m}\}$
   2. Otherwise remain silent
3. If $|S| = 1$ output $\mathrm{m} \in S$ otherwise output "Confused" (or default message)



f+1 rounds too slow for practice

Dolev Strong Example

f=2



Attack=1

Brutus

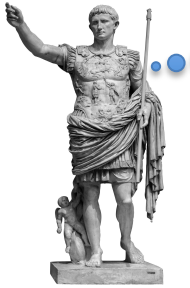
Marc Anthony

Pompeius

Augustus

Dolev Strong Example

f=2
r=1



Attack=1

Brutus

$1_{caesar,MA}$

Marc Anthony

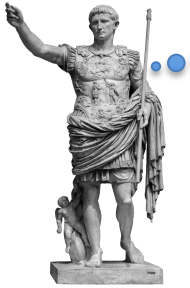
Pompeius

$1_{caesar,Aug}$

Augustus

Dolev Strong Example

f=2
r=2



Attack=1

REJECTED

Brutus

$0_{Brutus,Pompeius}$

Pompeius

Marc Anthony

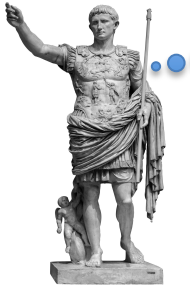
$1_{caesar,MA}$

Augustus

$1_{caesar,Aug}$

Dolev Strong Example

f=2
r=3



Attack=1

Brutus

$1_{caesar,MA}$

Marc Anthony

Pompeius

$1_{caesar, Aug}$

Augustus

Dolev Strong Example

f=2
r=3

Attack=1

Attack

Brutus

$1_{caesar,MA}$

Marc Anthony

Pompeius

Attack

$1_{caesar,Aug}$

Augustus

More than f corruptions

f=2
r=3

Brutus

$0_{Caesar,Brutus,Pompeius}$

$1_{Caesar,MA}$

Marc Anthony

Pompeius

$1_{Caesar, Aug}$

Augustus

More than f corruptions

f=2
r=3

Confused

0 Caesar,Brutus,Pompeius

1 Caesar,MA

Marc Anthony

Brutus

Attack

Pompeius

1 Caesar, Aug

Augustus

Dolev Strong Analysis

Why f+1 rounds?

f corrupt nodes can confuse honest node

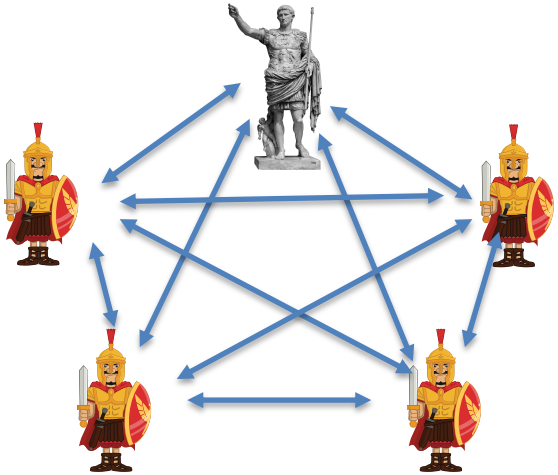
Validity?

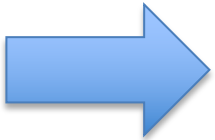
Honest nodes only update set S if signed by leader

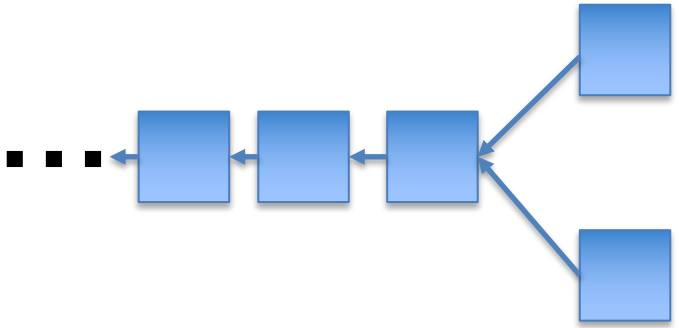
Consistency?

1. If honest node has $\mathrm{m} \in S$ at round $r \leq f$ then all other nodes will have $\mathrm{m} \in S$ at $r + 1$
2. If honest node receives new m at round $\mathrm{f} + 1$ then it must have received it from an honest node
3. -> All honest nodes have identical S

From Byzantine Consensus to Blockchains







Sybil Resistance

In BC participants are fixed but we want an open consensus





Approach 1:
Anyone can join

Permissioned Consensus

Proof of Stake

Weighted Byzantine Consensus

Super large consensus

Assumption 2/3rd of stake with honest nodes

How to initialize? Incentives?

Proof of Work

Recall: $H(x, y) < \frac{2^n}{D}$

Truly permissionless

3 TH/s

5 TH/s

2 TH/s

7 TH/s

5 TH/s

Terrible for the environment

More next lecture

Network Model

- Dolev Strong assumes messages gets delivered by next round
  - Not realistic (honest nodes can have network outages)
  - Protocol broken if messages aren’t delivered in time

Network Model

- **Synchronous:** There is known maximum delay Δ such that any message sent from one node to another is delivered within Δ time.
  - Protocol *can* use Δ as parameter

Any f (Dolev-Strong)

- **Partially Synchronous:** Δ exists but is unknown
  - Same protocol must work for any Δ
  - Equivalent definition: There exists periods of synchrony in which delay is Δ. Protocol does not know when these begin

$f < n/3$

- **Asynchronous:** Network experiences arbitrary failures
  - Consensus problem unsolvable

Blockchain Consensus

- ”State Machine Replication” on n nodes (or servers)
- Stream of transactions $tx_1, tx_2, \ldots$
- For $i = 1, \ldots, n$: $L_i(t)$ is a list of confirmed Tx by node i at time t
- Goal: Protocol that satisfies two properties:
  - ✓Nodes confirmed transactions are consistent with each other
  - ✓Transactions will eventually get confirmed

Blockchain Consensus

Consistency

For all honest nodes $i, j \in [n]$ and times t, t':

Either list $L_i(t)$ is a prefix of $L_j(t)$ or vice versa

Δ −Liveness

There exists function T such that:

If any honest node receives tx at time t then $\forall i \;\; tx \in L_i(t + \mathrm{T}(\Delta, \mathrm{n}))$

$\Delta = maximum\ network\ delay$

Blockchain from Byzantine Consensus

Epoch t

$L_1(t)$

$L_2(t)$

$S = \{tx_k, \dots, tx_l\}$

s.t. $tx_k, \dots, tx_l \notin L_1(t)$

$L_5(t)$

BC using S

$L_3(t)$

$L_4(t)$

Blockchain from Byzantine Consensus

Epoch t+1

$L_1(t+1) = L_1(t)||S$

$L_2(t+1) = L_2(t) \cup S$

BC using S

$L_5(t+1) = L_5(t) \cup S$

$L_3(t+1) = L_3(t) \cup S$

$L_4(t+1) = L_4(t) \cup S$

Blockchain from Byzantine Consensus

Epoch t+1

$L_1(t+1)$

$L_2(t+1)$

$L_3(t+1)$

$L_4(t+1)$

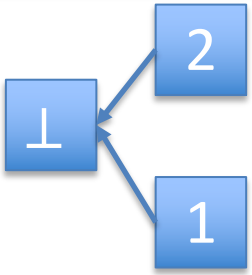
Malicious leader?

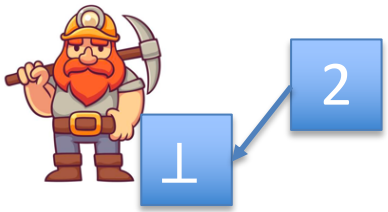
Confirmation time?

Streamlet: A simple Blockchain protocol

Assumptions:
n nodes (permissioned)
Less than 1/3 corrupt
Partially synchronous network
Proceed in epochs

Random rotating leader:
Leader id= H(epoch) mod n





Each node stores locally notarized chain

Streamlet [Chan,Shi20]

Propose Vote In every epoch:

1. Leader creates block of TXs extending *longest* local *notarized* chain
2. Nodes sign off on first block from leader iff it extends one of their longest local *notarized* chain
3. If Block has signatures from 2n/3 nodes it becomes *notarized*

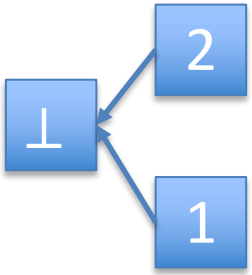
Finalize

1. If a chain has 3 notarized blocks from consecutive epochs, chop off the final block and *finalize* the chain

Streamlet: A simple Blockchain protocol

Assumptions:
Less than 1/3 corrupt
Partially synchronous network
Proceed in epochs

Random rotating leader:
Leader id= H(epoch) mod n

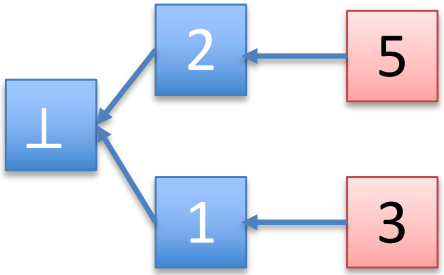


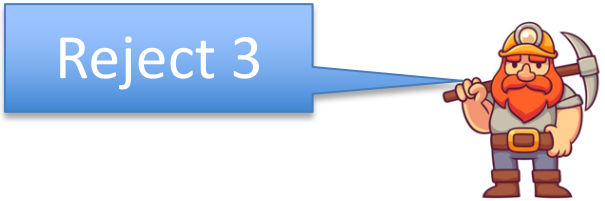
Each node stores locally notarized chain

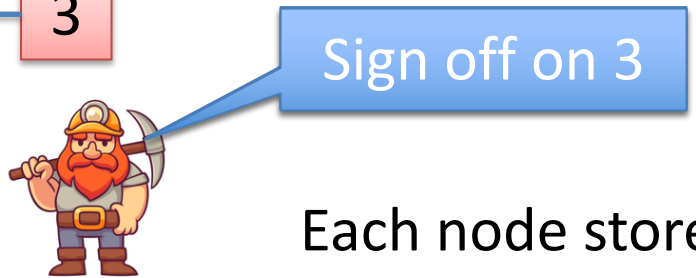
Streamlet: A simple Blockchain protocol

Assumptions:
Less than 1/3 corrupt
Partially synchronous network
Proceed in epochs

Random rotating leader:
Leader id= H(epoch) mod n



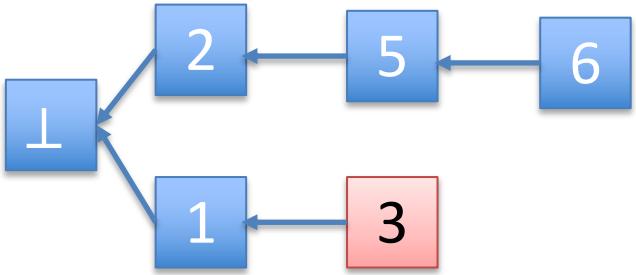




Each node stores locally notarized chain

Streamlet: A simple Blockchain protocol

Assumptions:
Less than 1/3 corrupt
Partially synchronous network
Proceed in epochs

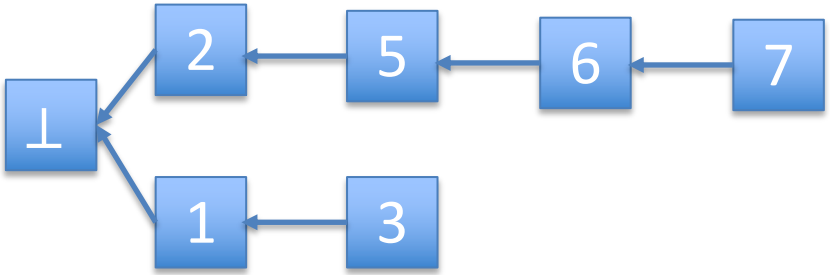


Random rotating leader:
Leader id= H(epoch) mod n

Each node stores locally notarized chain

Streamlet: A simple Blockchain protocol

Assumptions:
Less than 1/3 corrupt
Partially synchronous network
Proceed in epochs

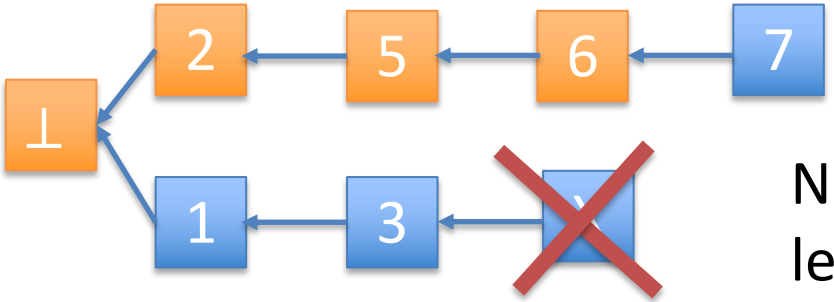


Random rotating leader:
Leader id= H(epoch) mod n

Each node stores locally notarized chain

Streamlet: A simple Blockchain protocol

Assumptions:
Less than 1/3 corrupt
Partially synchronous network
Proceed in epochs



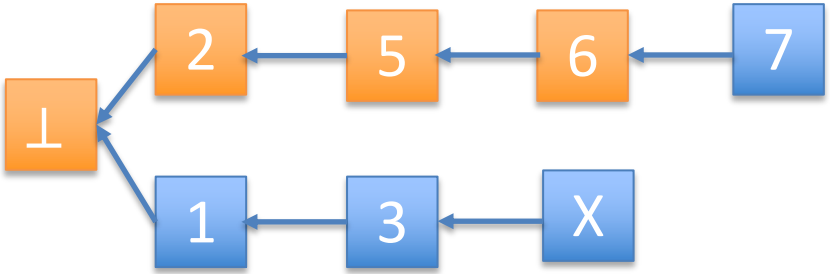
No other block on level 6 can be notarized

Random rotating leader:
Leader id= H(epoch) mod n

Each node stores locally notarized chain

Streamlet: Consistency Analysis

1. No two blocks with same epoch can be notarized (2/3 majority)
2. If X<5 then more than 1/3 honest nodes voted on 3. These nodes would never notarize 5 (because 5 doesn't extend 3). Without these 1/3+1 nodes 5 can't get notarized (Contradiction)
3. If X>7 more than 1/3 honest nodes have notarized 6. They won't notarize X because it doesn't extend 6



No other block on level 6 can be notarized.

Consistency holds irrespective of network

END OF LECTURE

Next lecture: Nakamoto Consensus, Incentives, Large Scale Consensus