Decentralized Consensus

Prof. David (Duc) Tran, PhD
University of Massachusetts, Boston (USA)

1

Muddy Children Puzzle

- **10** children played in the playground, **2** having **mud** on forehead
- Everyone can see each other's forehead, but not themselves
- After playing, teacher tell them "***some have mud on forehead***"

Problem

How do the children know that they have mud on forehead, **without communicating** with one another?

2

Muddy Children Puzzle: Solution

- **10** children played in the playground, **2** having mud on forehead
- Everyone can see each other's forehead, but not themselves
- After playing, teacher tell them "***some have mud on forehead***"

Solution

Teacher says to the kids: "***step up if you know have mud on your forehead***". As no one moves up, teacher repeats the request.

Question: in **which round** will some (all) muddy children step forward?

3

k children have mud → need k rounds

- k=1 (Muddy child: Alice)
  - Alice will step up because she sees that no other child has mud
- k=2 (Muddy children: Alice, Bob)
  - Round 1: Alice does not step up. She sees mud on Bob's forehead but is not sure if she has mud too (same thinking for Bob)
  - Round 2: Bob did not step in Round 1, so Alice knows that Bob sees somebody else having mud, and she could be one. But since she sees nobody else having mud other than Bob, that person must be herself. Thus, she steps up (same thinking for Bob; he steps up too)

- **In general, k<n:** after **k rounds, all** the k muddy children will step up

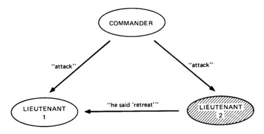
4

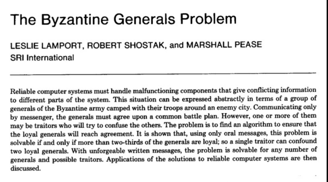
Some history

- Back in the 1970s: Aircraft control
  - Computers were being used in aircraft control. As a mission-critical system, it was important to replicate it on multiple machines to tolerate faults
  - NASA sponsored the *Software Implemented Fault Tolerance (SIFT)* project to build a resilient aircraft control system
  - In this project, Lamport et al. (1982) introduced the well-known ***"Byzantine Generals Problem"*** and **laid the foundation** of distributed consensus
- Since 2000: Industry adoption
  - Companies like Google and Facebook have adopted distributed consensus for mission-critical services such as Google Wallet and Facebook Credit.
- 2009: Bitcoin
  - A **new breakthrough** in distributed consensus, showing that consensus is viable in a decentralized, permissionless environment where anyone is allowed to participate

5

The Byzantine Generals Problem (1982)




The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, Pages 382–401.

6

Leslie Lamport

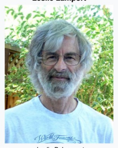
American **computer scientist**

B.S. degree (1960)
in **mathematics** from MIT

M.A. (1963) and Ph.D. (1972) degrees in **mathematics** from Brandeis University

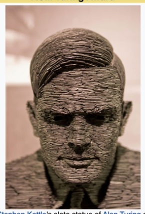
Winner of Turing Award

Leslie Lamport



Born	Leslie B. Lamport February 7, 1941 (age 80) New York City, New York
Alma mater	Massachusetts Institute of Technology (BSc) Brandeis University (PhD)
Known for	LaTeX Sequential consistency Atomic Register Hierarchy Lamport's bakery algorithm Byzantine fault tolerance Paxos algorithm Lamport signature
Awards	Dijkstra Prize (2000, 2005, and 2014) IEEE Emanuel R. Piore Award (2004) IEEE John von Neumann Medal (2008) Turing Award (2013)

ACM Turing Award



Stephen Kettle's slate statue of Alan Turing at Bletchley Park

Awarded for	Outstanding contributions in computer science
Country	United States
Presented by	Association for Computing Machinery (ACM)
Reward(s)	US $1,000,000[1]
First awarded	1966; 55 years ago

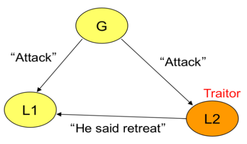
7

Byzantine Generals Problem

- Generals communicate to decide upon a common **action**: ATTACK or RETREAT
- The **commander** calls the action and sends it to all generals
- Some unknown generals are **traitors**, including the commander

Goals

1. **Consistency**: All loyal generals reach the same decision
2. **Validity**: If commander is loyal, all loyal generals will obey the commander



8

Byzantine Broadcast Model

Byzantine fault: Nodes can behave **arbitrarily maliciously** and can **collude** with each other

Byzantine Broadcast protocol
At the beginning, the **sender** receives an **input** bit ∈ {0, 1}. At the end, every node **outputs** a bit. Must satisfy 2 requirements:

1. **Consistency**: all **honest** nodes must output the same bit: **b = b'**
2. **Validity**: if the sender is honest and receives input bit **b**, all honest nodes must output bit **b**

9

Synchronous Network

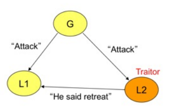
- When honest nodes send messages, the honest recipients will receive them within a bounded delay (called one "**round**")
- Synchrony assumption: If an honest node sends a message in **round r** to an honest recipient, then the recipient will receive the message at the beginning of **round (r + 1)**

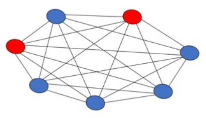
10

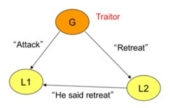
Main Result: **Solvable** iff >= **2/3rd honest**

Impossibility Result

- **f**: number of traitors
- **n**: number of generals
- No solution if **n < 3f+1**





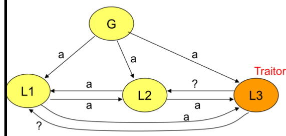


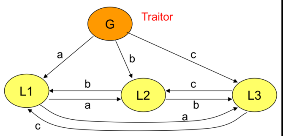
11

Solution for **n>3f**: Majority-Vote Idea

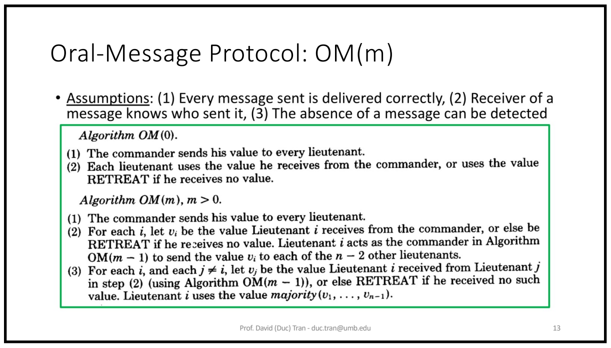
Sender: sends its value to all nodes
Each non-sender node: iterate over **f rounds**

1. Choose *value* = **most popular** value received in last round
2. Send *value* to all other non-sender nodes





12



Oral-Message Protocol: OM(m)

- Assumptions: (1) Every message sent is delivered correctly, (2) Receiver of a message knows who sent it, (3) The absence of a message can be detected

Algorithm OM(0).

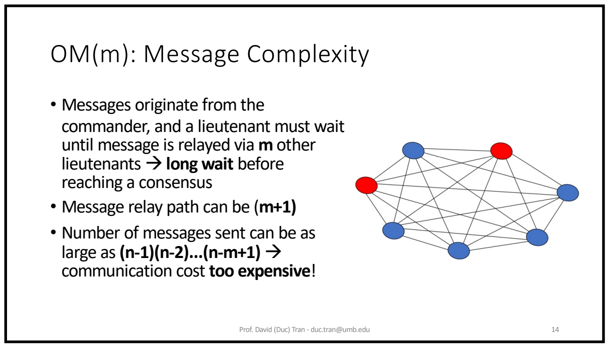
(1) The commander sends his value to every lieutenant.
(2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value.

Algorithm OM(m), $m > 0$.

(1) The commander sends his value to every lieutenant.
(2) For each i, let v_i be the value Lieutenant i receives from the commander, or else be RETREAT if he receives no value. Lieutenant i acts as the commander in Algorithm OM($m - 1$) to send the value v_i to each of the $n - 2$ other lieutenants.
(3) For each i, and each $j \neq i$, let v_j be the value Lieutenant i received from Lieutenant j in step (2) (using Algorithm OM($m - 1$)), or else RETREAT if he received no such value. Lieutenant i uses the value *majority*$(v_1, \ldots, v_{n-1})$.

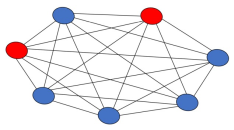
Prof. David (Duc) Tran - duc.tran@umb.edu 13

13



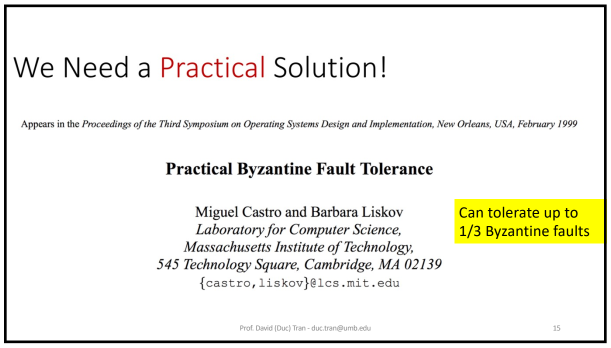
OM(m): Message Complexity

- Messages originate from the commander, and a lieutenant must wait until message is relayed via **m** other lieutenants → **long wait** before reaching a consensus
- Message relay path can be **(m+1)**
- Number of messages sent can be as large as **(n-1)(n-2)…(n-m+1)** → communication cost **too expensive**!



Prof. David (Duc) Tran - duc.tran@umb.edu 14

14



We Need a Practical Solution!

Appears in the *Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, February 1999*

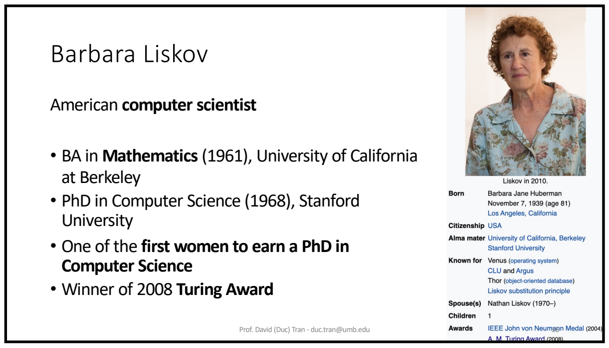
Practical Byzantine Fault Tolerance

Miguel Castro and Barbara Liskov
Laboratory for Computer Science,
Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139
{castro,liskov}@lcs.mit.edu

Can tolerate up to 1/3 Byzantine faults

Prof. David (Duc) Tran - duc.tran@umb.edu 15

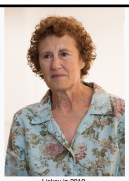
15



Barbara Liskov

American **computer scientist**

- BA in **Mathematics** (1961), University of California at Berkeley
- PhD in Computer Science (1968), Stanford University
- One of the **first women to earn a PhD in Computer Science**
- Winner of 2008 **Turing Award**

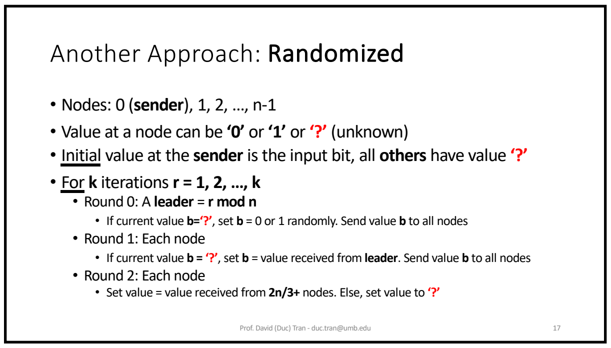


Liskov in 2010.

Born	Barbara Jane Huberman November 7, 1939 (age 81) Los Angeles, California
Citizenship	USA
Alma mater	University of California, Berkeley Stanford University
Known for	Venus (operating system) CLU and Argus Thor (object-oriented database) Liskov substitution principle
Spouse(s)	Nathan Liskov (1970–)
Children	1
Awards	IEEE John von Neumann Medal (2004) A. M. Turing Award (2008)

Prof. David (Duc) Tran - duc.tran@umb.edu 16

16

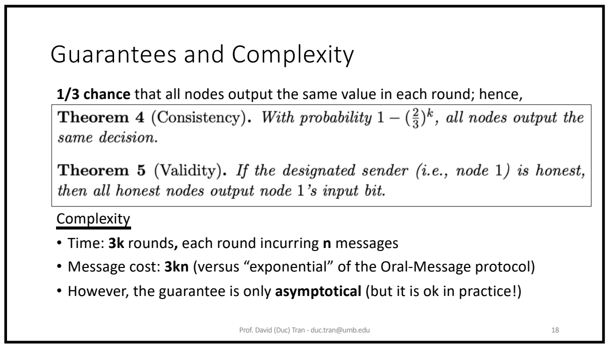


Another Approach: **Randomized**

- Nodes: 0 (**sender**), 1, 2, …, n-1
- Value at a node can be **'0'** or **'1'** or **'?'** (unknown)
- Initial value at the **sender** is the input bit, all **others** have value **'?'**
- For **k** iterations **r = 1, 2, …, k**
  - Round 0: A **leader** = **r mod n**
    - If current value **b**='?', set **b** = 0 or 1 randomly. Send value **b** to all nodes
  - Round 1: Each node
    - If current value **b** = '?', set **b** = value received from **leader**. Send value **b** to all nodes
  - Round 2: Each node
    - Set value = value received from **2n/3+** nodes. Else, set value to **'?'**

Prof. David (Duc) Tran - duc.tran@umb.edu 17

17



Guarantees and Complexity

1/3 chance that all nodes output the same value in each round; hence,

Theorem 4 (Consistency). *With probability* $1 - (\frac{2}{3})^k$, *all nodes output the same decision.*

Theorem 5 (Validity). *If the designated sender (i.e., node 1) is honest, then all honest nodes output node 1's input bit.*

Complexity

- Time: **3k** rounds**,** each round incurring **n** messages
- Message cost: **3kn** (versus "exponential" of the Oral-Message protocol)
- However, the guarantee is only **asymptotical** (but it is ok in practice!)

Prof. David (Duc) Tran - duc.tran@umb.edu 18

18

Solution for **n<3f+1**?

- Yes, as long as **f <= n-2**, with **public-key cryptography**

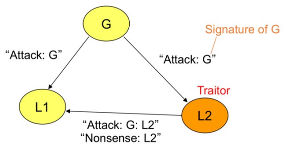
Assumption: an existing **digital signature** scheme

- Each node **i** has a public-secret key pair (**Pk_i**, **Sk_i**)
- The public key **PK_i** is known to everybody
- A node **signs** every message (with its secret key) before sending it
- Thus, guarantee the authentication of the received message

19

Digital Signatures help

- A traitor **cannot** change the original message or fake an identity

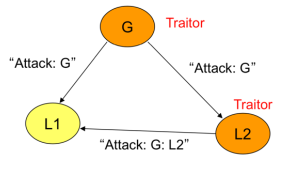


L2 may take two actions:

- Forward the signed message. This leads to correct outcome
- Send a different message: will be recognized by L1 as fake (loyal generals ignore messages coming from known traitors)

20

Why n >= f+2?

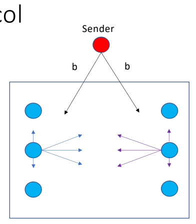


L1 cannot detect G, L2 are traitors attack!

21

A Naïve Majority Voting Protocol

- Sender (i.e. node 1) receives the bit b as input.
- **Round** 1: Node 1 sends $\langle b \rangle_1$ to every node (including itself).
- **Round** 2: Every node $i \in [n]$ does the following: if a single bit $\langle b' \rangle_1$ is received, send the vote $\langle b' \rangle_i$. Else send the vote $\langle 0 \rangle_i$.
- **Round** 3: If no bit or both bits received more than $n/2$ votes from distinct nodes, then output 0. Else output the bit that received more than $n/2$ votes from distinct nodes.

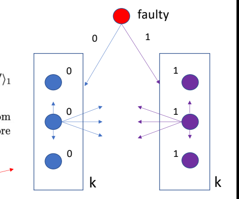


22

But it is **flawed**

- Sender (i.e. node 1) receives the bit b as input.
- **Round** 1: Node 1 sends $\langle b \rangle_1$ to every node (including itself).
- **Round** 2: Every node $i \in [n]$ does the following: if a single bit $\langle b' \rangle_1$ is received, send the vote $\langle b' \rangle_i$. Else send the vote $\langle 0 \rangle_i$.
- **Round** 3: If no bit or both bits received more than $n/2$ votes from distinct nodes, then output 0. Else output the bit that received more than $n/2$ votes from distinct nodes.

DOES NOT WORK: for example, when n=2k+1 nodes and the faulty Sender can attack as follows

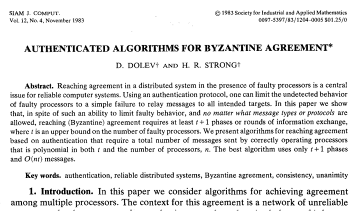


These nodes output 0 because 0 gets (k+1) votes

These nodes output 1 because 1 gets (k+1) votes

23

The Dolev-Strong Protocol


SIAM J. COMPUT.
Vol. 12, No. 4, November 1983

© 1983 Society for Industrial and Applied Mathematics
0097-5397/83/1204-0005 $01.25/0

AUTHENTICATED ALGORITHMS FOR BYZANTINE AGREEMENT*

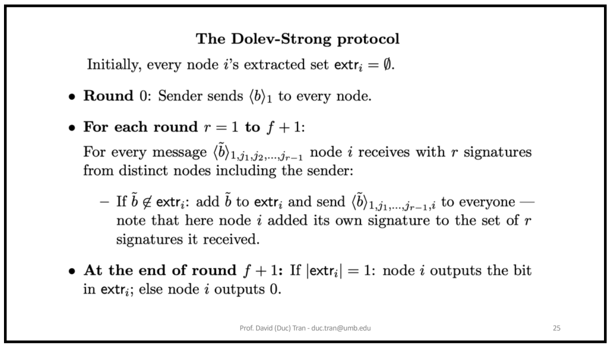
D. DOLEV† AND H. R. STRONG†

Abstract. Reaching agreement in a distributed system in the presence of faulty processors is a central issue for reliable computer systems. Using an authentication protocol, one can limit the undetected behavior of faulty processors to a simple failure to relay messages to all intended targets. In this paper we show that, in spite of such an ability to limit faulty behavior, and *no matter what message types or protocols* are allowed, reaching (Byzantine) agreement requires at least $t+1$ phases or rounds of information exchange, where t is an upper bound on the number of faulty processors. We present algorithms for reaching agreement based on authentication that require a total number of messages sent by correctly operating processors that is polynomial in both t and the number of processors, n. The best algorithm uses only $t+1$ phases and $O(nt)$ messages.

Key words. authentication, reliable distributed systems, Byzantine agreement, consistency, unanimity

1. Introduction. In this paper we consider algorithms for achieving agreement among multiple processors. The context for this agreement is a network of unreliable

24



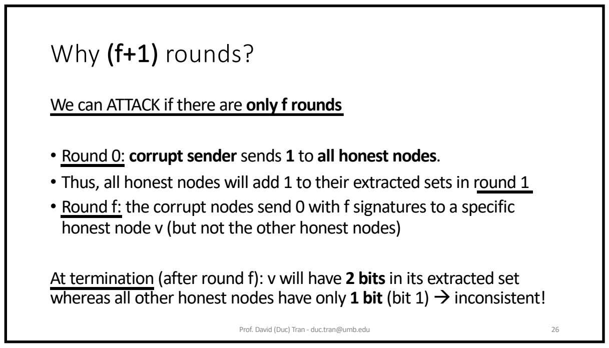
The Dolev-Strong protocol

Initially, every node i's extracted set $\mathsf{extr}_i = \emptyset$.

- **Round** 0: Sender sends $\langle b \rangle_1$ to every node.
- **For each round** $r = 1$ **to** $f + 1$:
  For every message $\langle \tilde{b} \rangle_{1,j_1,j_2,\ldots,j_{r-1}}$ node i receives with r signatures from distinct nodes including the sender:
  - If $\tilde{b} \notin \mathsf{extr}_i$: add $\tilde{b}$ to extr_i and send $\langle \tilde{b} \rangle_{1,j_1,\ldots,j_{r-1},i}$ to everyone — note that here node i added its own signature to the set of r signatures it received.
- **At the end of round** $f + 1$**:** If $|\mathsf{extr}_i| = 1$: node i outputs the bit in extr_i; else node i outputs 0.

Prof. David (Duc) Tran - duc.tran@umb.edu 25

25



Why **(f+1)** rounds?

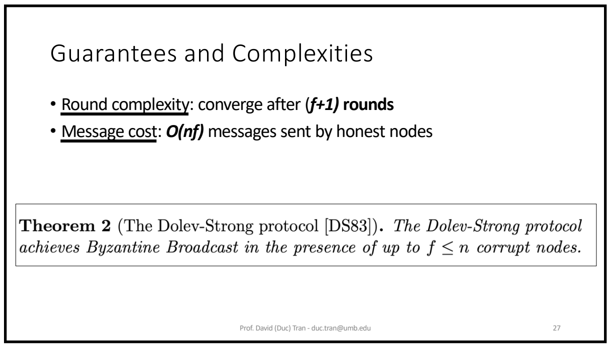
We can ATTACK if there are **only f rounds**

- Round 0: **corrupt sender** sends **1** to **all honest nodes**.
- Thus, all honest nodes will add 1 to their extracted sets in round 1
- Round f: the corrupt nodes send 0 with f signatures to a specific honest node v (but not the other honest nodes)

At termination (after round f): v will have **2 bits** in its extracted set whereas all other honest nodes have only **1 bit** (bit 1) → inconsistent!

Prof. David (Duc) Tran - duc.tran@umb.edu 26

26



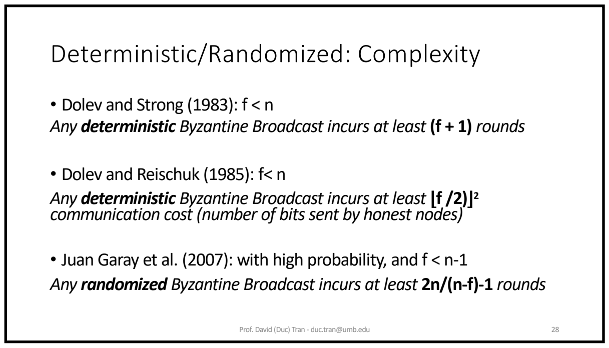
Guarantees and Complexities

- Round complexity: converge after ***(f+1)*** **rounds**
- Message cost: ***O(nf)*** messages sent by honest nodes

Theorem 2 (The Dolev-Strong protocol [DS83]). *The Dolev-Strong protocol achieves Byzantine Broadcast in the presence of up to $f \leq n$ corrupt nodes.*

Prof. David (Duc) Tran - duc.tran@umb.edu 27

27



Deterministic/Randomized: Complexity

- Dolev and Strong (1983): f < n

*Any **deterministic** Byzantine Broadcast incurs at least* **(f + 1)** *rounds*

- Dolev and Reischuk (1985): f< n

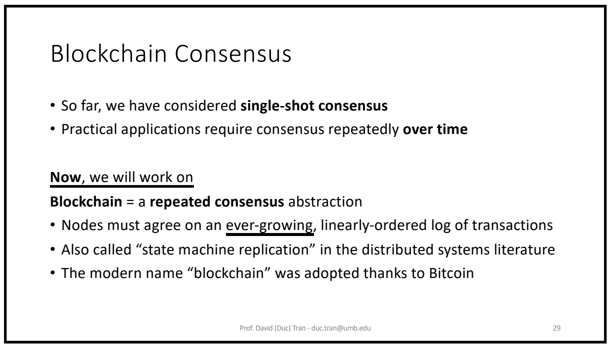
*Any **deterministic** Byzantine Broadcast incurs at least* **$\lfloor f/2 \rfloor^2$** *communication cost (number of bits sent by honest nodes)*

- Juan Garay et al. (2007): with high probability, and f < n-1

*Any **randomized** Byzantine Broadcast incurs at least* **2n/(n-f)-1** *rounds*

Prof. David (Duc) Tran - duc.tran@umb.edu 28

28



Blockchain Consensus

- So far, we have considered **single-shot consensus**
- Practical applications require consensus repeatedly **over time**

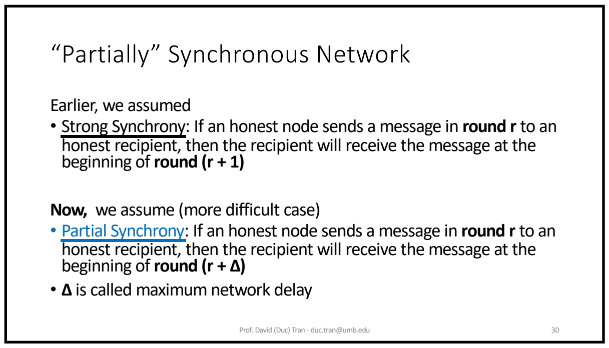
Now, we will work on

Blockchain = a **repeated consensus** abstraction

- Nodes must agree on an ever-growing, linearly-ordered log of transactions
- Also called “state machine replication” in the distributed systems literature
- The modern name “blockchain” was adopted thanks to Bitcoin

Prof. David (Duc) Tran - duc.tran@umb.edu 29

29



“Partially” Synchronous Network

Earlier, we assumed

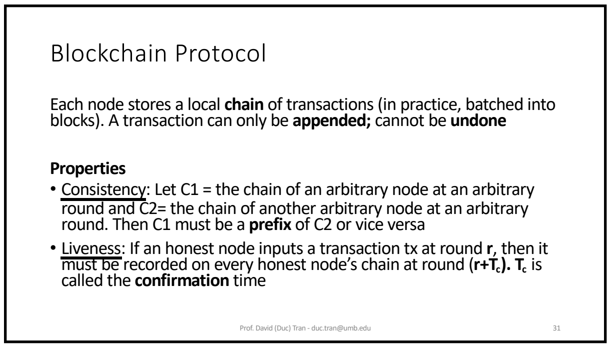
- Strong Synchrony: If an honest node sends a message in **round r** to an honest recipient, then the recipient will receive the message at the beginning of **round (r + 1)**

Now, we assume (more difficult case)

- Partial Synchrony: If an honest node sends a message in **round r** to an honest recipient, then the recipient will receive the message at the beginning of **round (r + Δ)**
- **Δ** is called maximum network delay

Prof. David (Duc) Tran - duc.tran@umb.edu 30

30



Blockchain Protocol

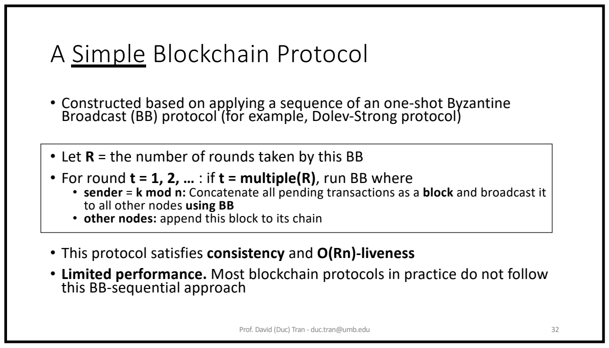
Each node stores a local **chain** of transactions (in practice, batched into blocks). A transaction can only be **appended;** cannot be **undone**

Properties

- Consistency: Let C1 = the chain of an arbitrary node at an arbitrary round and C2= the chain of another arbitrary node at an arbitrary round. Then C1 must be a **prefix** of C2 or vice versa
- Liveness: If an honest node inputs a transaction tx at round **r**, then it must be recorded on every honest node's chain at round **($r+T_c$).** **T_c** is called the **confirmation** time

Prof. David (Duc) Tran - duc.tran@umb.edu 31

31



A Simple Blockchain Protocol

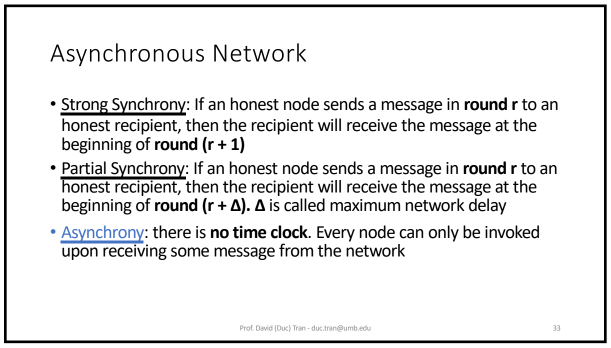
- Constructed based on applying a sequence of an one-shot Byzantine Broadcast (BB) protocol (for example, Dolev-Strong protocol)

- Let **R** = the number of rounds taken by this BB
- For round **t = 1, 2, …** : if **t = multiple(R)**, run BB where
  - **sender = k mod n:** Concatenate all pending transactions as a **block** and broadcast it to all other nodes **using BB**
  - **other nodes:** append this block to its chain

- This protocol satisfies **consistency** and **O(Rn)-liveness**
- **Limited performance.** Most blockchain protocols in practice do not follow this BB-sequential approach

Prof. David (Duc) Tran - duc.tran@umb.edu 32

32

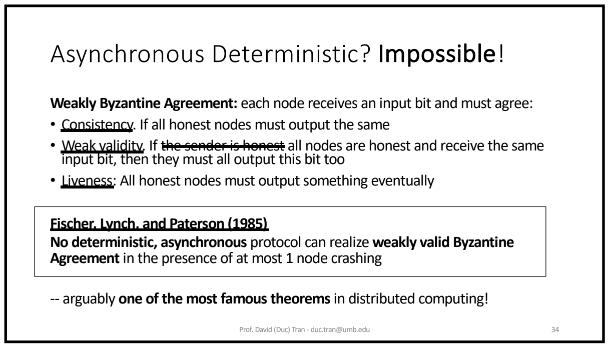


Asynchronous Network

- Strong Synchrony: If an honest node sends a message in **round r** to an honest recipient, then the recipient will receive the message at the beginning of **round (r + 1)**
- Partial Synchrony: If an honest node sends a message in **round r** to an honest recipient, then the recipient will receive the message at the beginning of **round (r + Δ).** **Δ** is called maximum network delay
- Asynchrony: there is **no time clock**. Every node can only be invoked upon receiving some message from the network

Prof. David (Duc) Tran - duc.tran@umb.edu 33

33



Asynchronous Deterministic? **Impossible**!

Weakly Byzantine Agreement: each node receives an input bit and must agree:

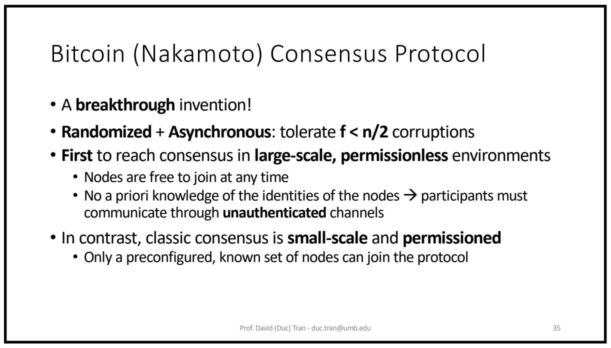
- Consistency. If all honest nodes must output the same
- Weak validity. If ~~the sender is honest~~ all nodes are honest and receive the same input bit, then they must all output this bit too
- Liveness: All honest nodes must output something eventually

Fischer, Lynch, and Paterson (1985)
No deterministic, asynchronous protocol can realize **weakly valid Byzantine Agreement** in the presence of at most 1 node crashing

-- arguably **one of the most famous theorems** in distributed computing!

Prof. David (Duc) Tran - duc.tran@umb.edu 34

34

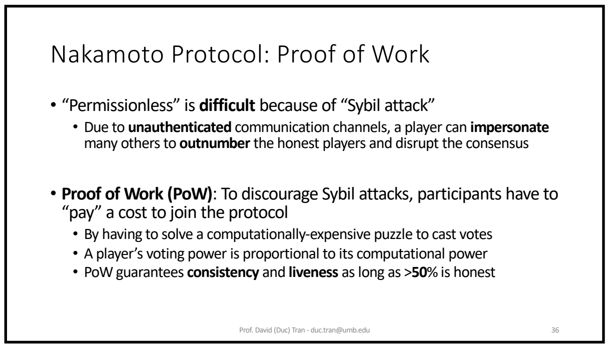


Bitcoin (Nakamoto) Consensus Protocol

- A **breakthrough** invention!
- **Randomized** + **Asynchronous**: tolerate **$f < n/2$** corruptions
- **First** to reach consensus in **large-scale, permissionless** environments
  - Nodes are free to join at any time
  - No a priori knowledge of the identities of the nodes → participants must communicate through **unauthenticated** channels
- In contrast, classic consensus is **small-scale** and **permissioned**
  - Only a preconfigured, known set of nodes can join the protocol

Prof. David (Duc) Tran - duc.tran@umb.edu 35

35



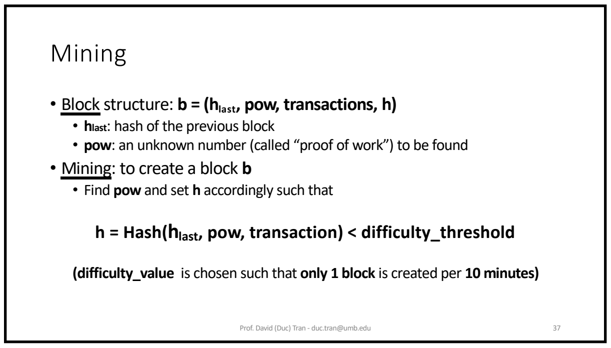
Nakamoto Protocol: Proof of Work

- "Permissionless" is **difficult** because of "Sybil attack"
  - Due to **unauthenticated** communication channels, a player can **impersonate** many others to **outnumber** the honest players and disrupt the consensus

- **Proof of Work (PoW)**: To discourage Sybil attacks, participants have to "pay" a cost to join the protocol
  - By having to solve a computationally-expensive puzzle to cast votes
  - A player's voting power is proportional to its computational power
  - PoW guarantees **consistency** and **liveness** as long as >**50%** is honest

Prof. David (Duc) Tran - duc.tran@umb.edu 36

36



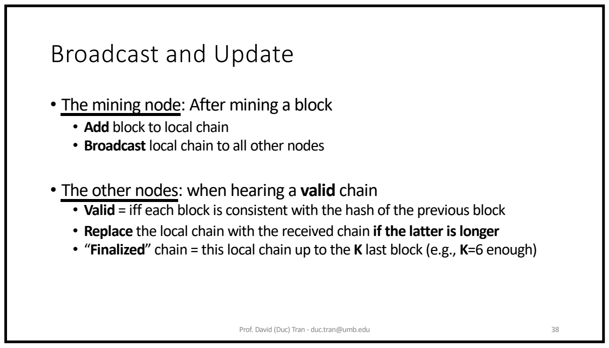
Mining

- Block structure: **b = (h_{last}, pow, transactions, h)**
  - **h_{last}**: hash of the previous block
  - **pow**: an unknown number (called "proof of work") to be found
- Mining: to create a block **b**
  - Find **pow** and set **h** accordingly such that

h = Hash(h_{last}, pow, transaction) < difficulty_threshold

(difficulty_value is chosen such that **only 1 block** is created per **10 minutes)**

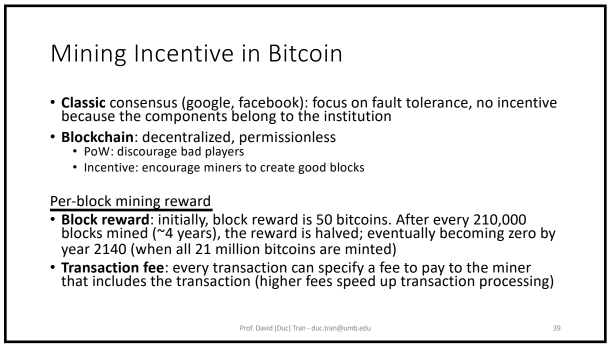
37



Broadcast and Update

- The mining node: After mining a block
  - **Add** block to local chain
  - **Broadcast** local chain to all other nodes
- The other nodes: when hearing a **valid** chain
  - **Valid** = iff each block is consistent with the hash of the previous block
  - **Replace** the local chain with the received chain **if the latter is longer**
  - "**Finalized**" chain = this local chain up to the **K** last block (e.g., **K**=6 enough)

38

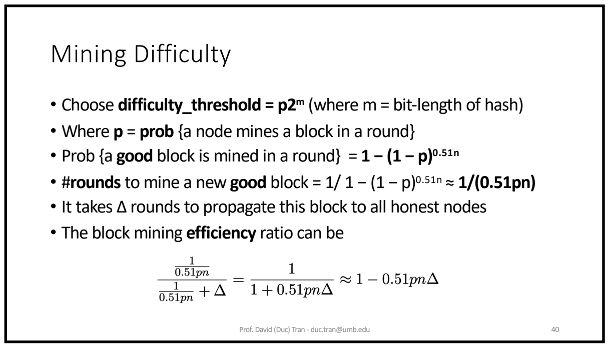


Mining Incentive in Bitcoin

- **Classic** consensus (google, facebook): focus on fault tolerance, no incentive because the components belong to the institution
- **Blockchain**: decentralized, permissionless
  - PoW: discourage bad players
  - Incentive: encourage miners to create good blocks

Per-block mining reward
- **Block reward**: initially, block reward is 50 bitcoins. After every 210,000 blocks mined (~4 years), the reward is halved; eventually becoming zero by year 2140 (when all 21 million bitcoins are minted)
- **Transaction fee**: every transaction can specify a fee to pay to the miner that includes the transaction (higher fees speed up transaction processing)

39

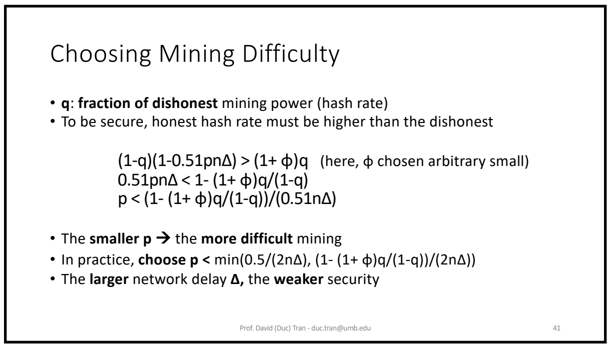


Mining Difficulty

- Choose **difficulty_threshold = $p2^m$** (where m = bit-length of hash)
- Where **p** = **prob** {a node mines a block in a round}
- Prob {a **good** block is mined in a round} = **$1 - (1 - p)^{0.51n}$**
- **#rounds** to mine a new **good** block = $1/ 1 - (1 - p)^{0.51n} \approx$ **1/(0.51pn)**
- It takes Δ rounds to propagate this block to all honest nodes
- The block mining **efficiency** ratio can be

$$\frac{\frac{1}{0.51pn}}{\frac{1}{0.51pn}+\Delta}=\frac{1}{1+0.51pn\Delta}\approx 1-0.51pn\Delta$$

40



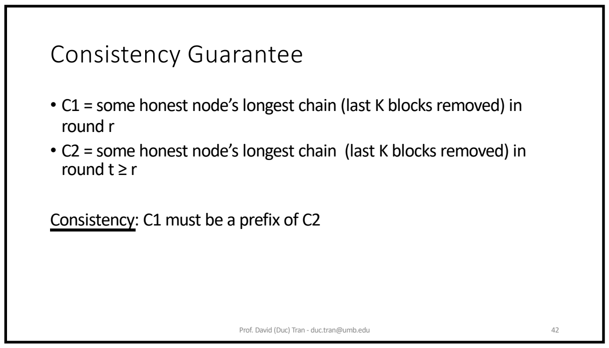
Choosing Mining Difficulty

- **q**: **fraction of dishonest** mining power (hash rate)
- To be secure, honest hash rate must be higher than the dishonest

$(1-q)(1-0.51pn\Delta) > (1+ \phi)q$ (here, ϕ chosen arbitrary small)
$0.51pn\Delta < 1- (1+ \phi)q/(1-q)$
$p < (1- (1+ \phi)q/(1-q))/(0.51n\Delta)$

- The **smaller p** → the **more difficult** mining
- In practice, **choose p <** $\min(0.5/(2n\Delta), (1- (1+ \phi)q/(1-q))/(2n\Delta))$
- The **larger** network delay **Δ,** the **weaker** security

41



Consistency Guarantee

- C1 = some honest node's longest chain (last K blocks removed) in round r
- C2 = some honest node's longest chain (last K blocks removed) in round t ≥ r

Consistency: C1 must be a prefix of C2

42

Chain Growth Guarantee

- #honest nodes that mine a block in each round = **(1-q)np**
- #good blocks added > **(1-q)np(1- 2nΔ)**

After any **t >= K/(qnp) rounds**, any honest node's chain will have added at least **(1-q)np(1- 2nΔ)t** blocks

For any honest node; its chain is always growing

43

Liveness Guarantee

In every window of consecutive **K** blocks in honest nodes' longest chains, more than μ := 1 – 1/(1+ϕ) fraction are mined by honest nodes

What that means
- Every now and then, an honest block is added to the blockchain
- Hence, **liveness**: transactions submitted will be recorded in honest nodes' finalized chains fairly soon (after **Θ(K/((1-q)np) + Δ)** rounds)

44

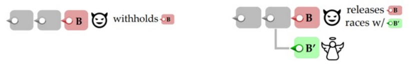
Mining Fairness: It is not fair!

- **Fairness** iff the honest block creation rate = the honest hash rate
- the **honest block rate** is μ := 1 – 1/(1+ϕ)
- the **honest hash rate** is 1-**q**
- Assume **Δ=0** (ideal case for honest nodes) and set **(1-q) = (1+ ϕ)q**
- We have **μ := 1 – 1/(1+ϕ) = 1-q/(1-q) <** (1-**q**) (not fair!)
- Attack: a coalition with 49% hash rate can control 96% the blocks!

45

Selfish Mining Attack

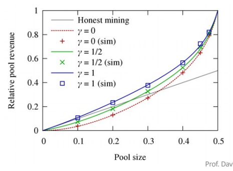
- Capitalize on the unfairness of Nakamoto consensus: damage transactions, earn block mining rewards
- Selfish miner: mine a block B, but **withhold** it until some honest miner also mines a block B' at the same length (block number) as B
  - When this happens, immediately releases B
  - If B is **transmitted faster** than B', honest nodes' work (B') is **wasted**
  - **Feasible:** bad nodes collude with the network relay to deliver blocks to miners

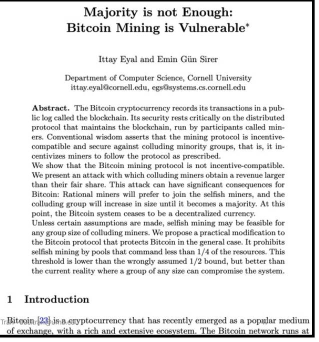


46

Selfish Mining Attack

- A coalition of selfish miners with 49% mining power can control 96% the blocks!




**Majority is not Enough:
Bitcoin Mining is Vulnerable***

Ittay Eyal and Emin Gün Sirer

Department of Computer Science, Cornell University
ittay.eyal@cornell.edu, egs@systems.cs.cornell.edu

Abstract. The Bitcoin cryptocurrency records its transactions in a public log called the blockchain. Its security rests critically on the distributed protocol that maintains the blockchain, run by participants called miners. Conventional wisdom asserts that the mining protocol is incentive-compatible and secure against colluding minority groups, that is, it incentivizes miners to follow the protocol as prescribed.
We show that the Bitcoin mining protocol is not incentive-compatible. We present an attack with which colluding miners obtain a revenue larger than their fair share. This attack can have significant consequences for Bitcoin: Rational miners will prefer to join the selfish miners, and the colluding group will increase in size until it becomes a majority. At this point, the Bitcoin system ceases to be a decentralized currency.
Unless certain assumptions are made, selfish mining may be feasible for any group size of colluding miners. We propose a practical modification to the Bitcoin protocol that protects Bitcoin in the general case. It prohibits selfish mining by pools that command less than 1/4 of the resources. This threshold is lower than the wrongly assumed 1/2 bound, but better than the current reality where a group of any size can compromise the system.

1 Introduction

Bitcoin [23] is a cryptocurrency that has recently emerged as a popular medium of exchange, with a rich and extensive ecosystem. The Bitcoin network runs at

47

The Double Spending Problem

- A malicious miner makes a payment, then secretively creates a second conflicting transaction in a new block, which allows him to recover the funds
- Feasible if he controls **q>50%** hashrate → mining **faster** than the rest of the network → make his local chain **the longest**
- But due to randomness, if **q<50%,** there is **still a non-zero chance**
- **How to minimize risk?** When somebody pays you, **wait some time** before delivering service. In bitcoin, wait for **6 block confirmations**

48

Why 6 Block Confirmations?

- Consider a miner with a fraction $0 < p \le 1$ of the total hash rate
- The whole network takes on average τ_0 = 10 minutes to create a block
- The miner takes on average $t_0 = \tau p$ time to create a block
- T = T1, T2, . . . , Tn: inter-block mining time of block 1, block 2, …
- Because mining is a Markov process (i.e., memoryless), T follows an **exponential** distribution

$$f_\tau(t) = \alpha e^{-\alpha t}$$

where $\alpha = 1/t_0$

49

Poisson Law

The time needed to discover n blocks is $S_n = T_1+T_2+\ldots+T_n$

N(t): #blocks validated at time t is **Poisson** with mean value αt

$$\mathbf{S}_n = \mathbf{T}_1 + \mathbf{T}_2 + \ldots + \mathbf{T}_n \,.$$

The random variable $\mathbf{S}_n$ follows the n-convolution of the exponential distribution and, as is well known, this gives a Gamma distribution with parameters (n, α),

$$f_{\mathbf{S}_n}(t) = \frac{\alpha^n}{(n-1)!} t^{n-1} e^{-\alpha t}$$

with cumulative distribution

$$F_{\mathbf{S}_n}(t) = \int_0^t f_{\mathbf{S}_n}(u)du = 1 - e^{-\alpha t} \sum_{k=0}^{n-1} \frac{(\alpha t)^k}{k!} \,.$$

From this we conclude that if $\mathbf{N}(t)$ is the process counting the number of blocks validated at time $t > 0$, $\mathbf{N}(t) = \max\{n \ge 0; \mathbf{S}_n < t\}$, then we have

$$\mathbb{P}[\mathbf{N}(t) = n] = F_{\mathbf{S}_n}(t) - F_{\mathbf{S}_{n+1}}(t) = \frac{(\alpha t)^n}{n!} e^{-\alpha t} \,,$$

Cyril Grunspan, Ricardo Pérez-Marco. The mathematics of Bitcoin. European Mathematical Society Newsletter, 2020. hal-02486029

50

Winning the race against the malicious

- **q**: hash rate of the malicious
- **N'(t) =** #malicious blocks at time t, which is Poisson
- **X_n = N'(S_n)** : #malicious blocks for every n consecutive honest blocks
  - S_n = time at which the honest has mined n blocks
  - X_n = a **negative binomial variable** with parameters (n, p)

$$\mathbb{P}[\mathbf{X}_n = k] = p^k q^k \binom{k+n-1}{k} \,.$$

How likely the malicious wins if behind the dishonest by z blocks? (similar to the classical **gambler's ruin problem**)

- E.g., if q=0.1, z=6, the P(z) = 1%

$$P(z) = I_{4pq}(z, 1/2)$$

where $I_x(a, b)$ is the incomplete regularized beta function

$$I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1}\, dt \,.$$

Cyril Grunspan, Ricardo Pérez-Marco. The mathematics of Bitcoin. European Mathematical Society Newsletter, 2020. hal-02486029

51

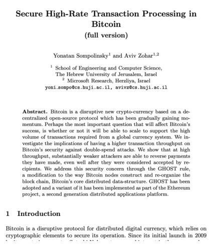
GHOST Consensus Protocol: Motivation

- Recall Bitcoin protocol
  - A node maintains a chain
  - When chain is updated, broadcast it
  - **Longest** chain wins
- If validation is too quick (for example, (1 minute instead of 10 minutes/block) → many stale blocks
  - A successfully mines a block X, broadcasts it
  - B successfully mines a block Y (after X is mined at A but before X arrives at B) → wasted effort → stale block Y
- If a mining pool is too strong → grow chain quickly → win most of the time → centralization of power

52

GHOST Consensus Protocol

- **Winning chain** : not the longest, but representing the most amount of PoW
- A node maintains **a tree** of all valid blocks it received (all uncles included)
  - Heavy tree implies a lot of PoW done
- When tree is updated, broadcast it
- **Greedy Heaviest Observed Subtree** (GHOST) rule to pick the winning chain



Secure High-Rate Transaction Processing in Bitcoin
(full version)

Yonatan Sompolinsky[1] and Aviv Zohar[1,2]

[1] School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel
[2] Microsoft Research, Herzliya, Israel
yoni_sompo@cs.huji.ac.il, avivz@cs.huji.ac.il

Abstract. Bitcoin is a disruptive new crypto-currency based on a decentralized open-source protocol which has been gradually gaining momentum. Perhaps the most important question that will affect Bitcoin's success, is whether or not it will be able to scale to support the high volume of transactions required from a global currency system. We investigate the implications of having a higher transaction throughput on Bitcoin's security against double-spend attacks. We show that at high throughput, substantially weaker attackers are able to reverse payments they have made, even well after they were considered accepted by recipients. We address this security concern through the GHOST rule, a modification to the way Bitcoin nodes construct and re-organize the block chain, Bitcoin's core distributed data-structure. GHOST has been adopted and a variant of it has been implemented as part of the Ethereum project, a second generation distributed applications platform.

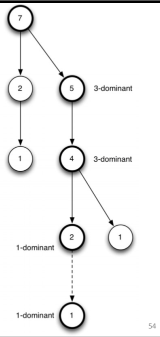
1 Introduction

Bitcoin is a disruptive protocol for distributed digital currency, which relies on cryptographic elements to secure its operation. Since its initial launch in 2009

53

GHOST: Example

- Each block has a **score =** weight of its subtree
  - **Winning chain = path** of blocks with **highest scores**, starting from root
- Each time a block is added to tree, score will be updated accordingly
- If block is **honest** -> increase score of the winning chain → less likely to be attacked



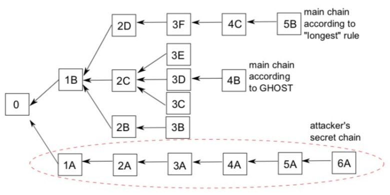
54

GHOST: More difficult to attack

Adversary

must grow the chain **much longer** to beat the heavy-weight honest chain

→ more difficult


main chain according to "longest" rule
main chain according to GHOST
attacker's secret chain
0
1B
2D
2C
2B
3F
3E
3D
3C
3B
4C
4B
5B
1A
2A
3A
4A
5A
6A

Taken from: SPECTRE: Serialization of Proof-of-Work Events, Confirming Transactions via Recursive Elections (by Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar)

55

GHOST: Implementation in Ethereum

- B = block to be added
- B must include a parent, and 0 or more uncles
- The included uncles must satisfy
  - A descendent of the 7th **generation** ancestor of B
  - Valid block header, not necessarily previously verified/validated
  - Different from uncles included in previous blocks
- For each uncle U in block B
  - Miner of B gets an additional 3.125% of block reward
  - Miner of U gets 93.75%

56

Nominated Proof of Stake (NPOS)

- Used in Polkadot blockchain (https://polkadot.network/)
- POS: Require a **small** set of "validator" nodes
- Many nodes want to serve as validators, but there can only be a few validators. And they are the only ones to receive block rewards
- **NPOS**: Give opportunities for all nodes to earn block rewards
  - Any node can be a "**nominator**": stake money in 1 or more validator candidates
  - If their candidates win:
    - the block **rewards** of a validator will be **shared** with its nominators
    - If a validator **behaves** badly, all the deposited stake will be **slashed**

57

Validator Nomination and Election

- Any node may choose to become a **validator candidate** or a **nominator**
- Each candidate: indicate how much money he wants to stake and his desired commission fee
- Each nominator: locks some stake and publish a list of preferred candidates
- Validator election: based on nomination ballots to choose a small set of validators with the most backing
- When: Validator election occurs in every era (roughly 1 day)

58

Election Protocol

- Solve a **multi-winner election problem**
  - A committee = set of validators, a minority = subset of nominators
  - Voting strength = stake money
- **Decentralization** objective
  - Each minority is represented proportionally to vote strength
  - No minority is under-represented
  - → Avoid centralization of power in the election
- **Security** objective:
  - If a nominator gets 2+ candidates elected, split stake among them
  - Maximize and balance the aggregate support for validators
  - → Expensive for an adversary to gain control over one validator,
  - → Expensive slashing penalty as a result of a validator's misconduct

Solve a **Proportional Representation Problem** in voting theory

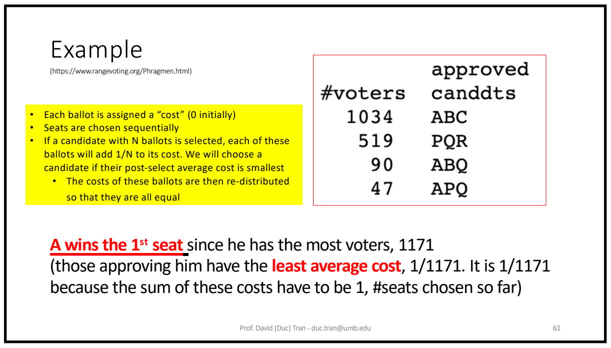
Solve a **Maximin Support Problem**

59

Proportional Representation

- Work by Edvard Phragmen and Thorvald Thiele in **1894** (https://www.rangevoting.org/Phragmen.html)
- Recently, considerable research efforts to formalize the notion of proportional representation, revisit the methods by Phragmen and Thorvald, and optimize them algorithmically
- Validator Election in NPOS is an adaptation of **Phragmen's method** which satisfies the technical properties of **Proportional Justified Representation** (Fernandez et al, 2017)

60



Example

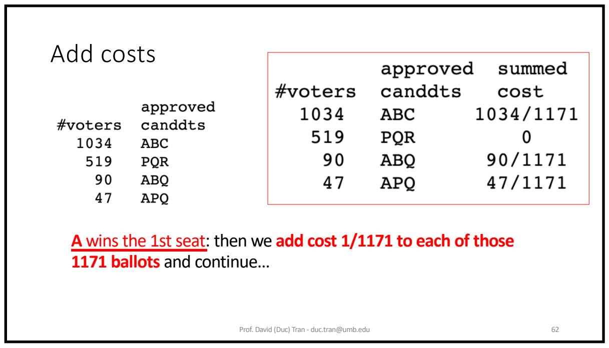
(https://www.rangevoting.org/Phragmen.html)

- Each ballot is assigned a "cost" (0 initially)
- Seats are chosen sequentially
- If a candidate with N ballots is selected, each of these ballots will add 1/N to its cost. We will choose a candidate if their post-select average cost is smallest
  - The costs of these ballots are then re-distributed so that they are all equal

#voters	approved canddts
1034	ABC
519	PQR
90	ABQ
47	APQ

A wins the 1st seat since he has the most voters, 1171 (those approving him have the **least average cost**, 1/1171. It is 1/1171 because the sum of these costs have to be 1, #seats chosen so far)

61



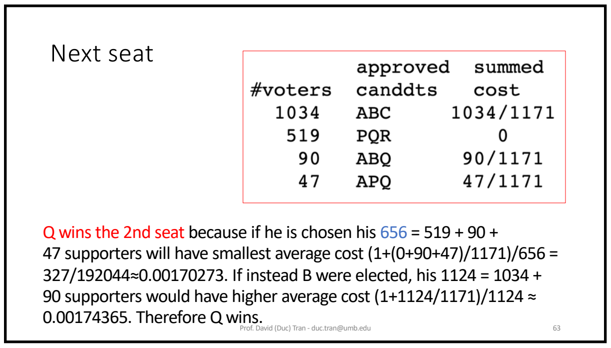
Add costs

#voters	approved canddts
1034	ABC
519	PQR
90	ABQ
47	APQ

#voters	approved canddts	summed cost
1034	ABC	1034/1171
519	PQR	0
90	ABQ	90/1171
47	APQ	47/1171

A wins the 1st seat: then we **add cost 1/1171 to each of those 1171 ballots** and continue…

62

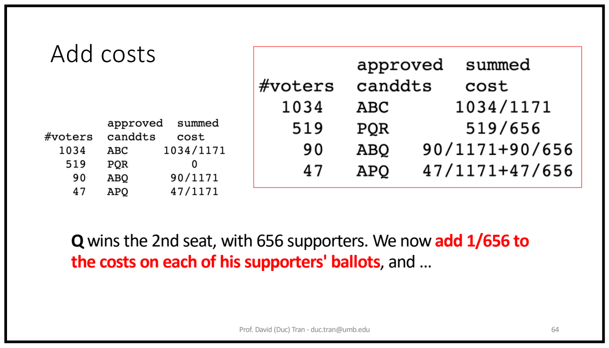


Next seat

#voters	approved canddts	summed cost
1034	ABC	1034/1171
519	PQR	0
90	ABQ	90/1171
47	APQ	47/1171

Q wins the 2nd seat because if he is chosen his 656 = 519 + 90 + 47 supporters will have smallest average cost (1+(0+90+47)/1171)/656 = 327/192044≈0.00170273. If instead B were elected, his 1124 = 1034 + 90 supporters would have higher average cost (1+1124/1171)/1124 ≈ 0.00174365. Therefore Q wins.

63



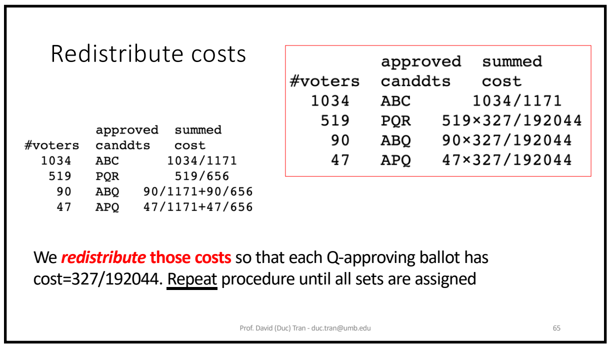
Add costs

#voters	approved canddts	summed cost
1034	ABC	1034/1171
519	PQR	0
90	ABQ	90/1171
47	APQ	47/1171

#voters	approved canddts	summed cost
1034	ABC	1034/1171
519	PQR	519/656
90	ABQ	90/1171+90/656
47	APQ	47/1171+47/656

Q wins the 2nd seat, with 656 supporters. We now **add 1/656 to the costs on each of his supporters' ballots**, and …

64



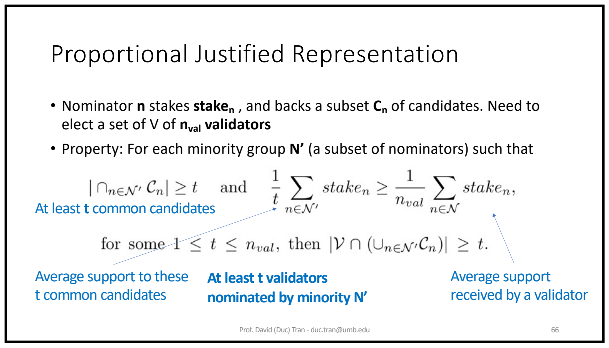
Redistribute costs

#voters	approved canddts	summed cost
1034	ABC	1034/1171
519	PQR	519/656
90	ABQ	90/1171+90/656
47	APQ	47/1171+47/656

#voters	approved canddts	summed cost
1034	ABC	1034/1171
519	PQR	519×327/192044
90	ABQ	90×327/192044
47	APQ	47×327/192044

We ***redistribute* those costs** so that each Q-approving ballot has cost=327/192044. Repeat procedure until all sets are assigned

65



Proportional Justified Representation

- Nominator **n** stakes **stake$_n$**, and backs a subset **C$_n$** of candidates. Need to elect a set of V of **n$_{val}$ validators**
- Property: For each minority group **N'** (a subset of nominators) such that

$$|\cap_{n \in \mathcal{N}'} \mathcal{C}_n| \geq t \quad \text{and} \quad \frac{1}{t}\sum_{n \in \mathcal{N}'} stake_n \geq \frac{1}{n_{val}} \sum_{n \in \mathcal{N}} stake_n,$$

At least **t** common candidates

$$\text{for some } 1 \leq t \leq n_{val}, \text{ then } |\mathcal{V} \cap (\cup_{n \in \mathcal{N}'} \mathcal{C}_n)| \geq t.$$

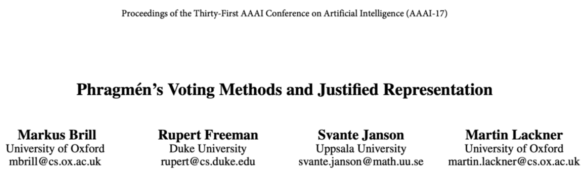
Average support to these t common candidates

At least t validators nominated by minority N'

Average support received by a validator

66

Recommended Reading



Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Phragmén's Voting Methods and Justified Representation

Markus Brill University of Oxford mbrill@cs.ox.ac.uk
Rupert Freeman Duke University rupert@cs.duke.edu
Svante Janson Uppsala University svante.janson@math.uu.se
Martin Lackner University of Oxford martin.lackner@cs.ox.ac.uk

67

Maxmin Support Problem (NP-hard)

- Need to compute the distribution of each nominator's stake among her chosen validators

$$f : \mathcal{N} \times \mathcal{V} \to \mathbb{R}_{\geq 0}$$

such that

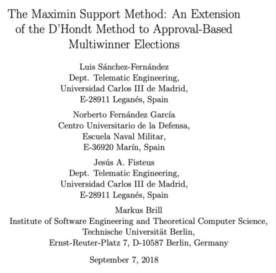
$$\sum_{v \in \mathcal{V} \cap \mathcal{C}_n} f(n, v) = stake_n \quad \text{for each nominator } n \in \mathcal{N},$$

- Objective

$$\max_{(\mathcal{V}, f)} \min_{v \in \mathcal{V}} support_f(v), \quad \text{where } support_f(v) := \sum_{n \in \mathcal{N}:\ v \in \mathcal{C}_n} f(n, v).$$

68

Recommended Reading



The Maximin Support Method: An Extension of the D'Hondt Method to Approval-Based Multiwinner Elections

Luis Sánchez-Fernández
Dept. Telematic Engineering,
Universidad Carlos III de Madrid,
E-28911 Leganés, Spain

Norberto Fernández García
Centro Universitario de la Defensa,
Escuela Naval Militar,
E-36920 Marín, Spain

Jesús A. Fisteus
Dept. Telematic Engineering,
Universidad Carlos III de Madrid,
E-28911 Leganés, Spain

Markus Brill
Institute of Software Engineering and Theoretical Computer Science,
Technische Universität Berlin,
Ernst-Reuter-Platz 7, D-10587 Berlin, Germany

September 7, 2018

69