COMP3221
Distributed Systems

Blockchain Scalability

Presented by A/Prof. Vincent Gramoli

THE UNIVERSITY OF SYDNEY

https://www.bankrate.com/investing/best-blockchain-etfs/

USER NOTE:
It is optional to include
this additional slide.

The University of Sydney's Camperdown Campus sits on the lands of the Gadigal people with campuses, teaching and research facilities on the lands of the Gamaraygal, Dharug, Wangal, Darkinyung, Burramadagal, Dharawal, Gandangara, Gamilaraay, Barkindji, Bundjalung, Wiradjuri, Ngunawal, Gureng Gureng, and Gagadju peoples.

THE UNIVERSITY OF SYDNEY

USER NOTE:
There are more Content Slide options available to add to your presentation by going to:
Home > New Slide > Drop down arrow

Syllabus

- Blockchain
  - History of Blockchain
- Consensus
- Security
- Robustness
- Scalability
- Other Challenges

Full Disclosure

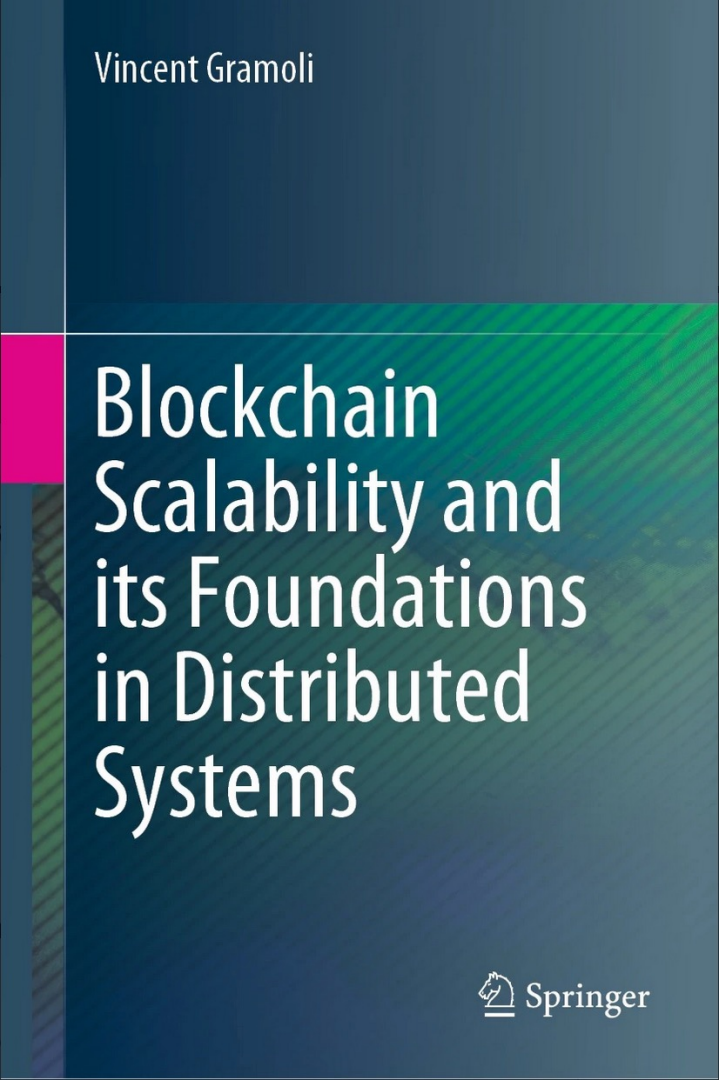
Australian Government
Australian Research Council

CSIRO DATA 61

REDBELLY

ethereum foundation

Algorand Centres of Excellence



Textbook:

Blockchain Scalability and its Foundations in Distributed Systems

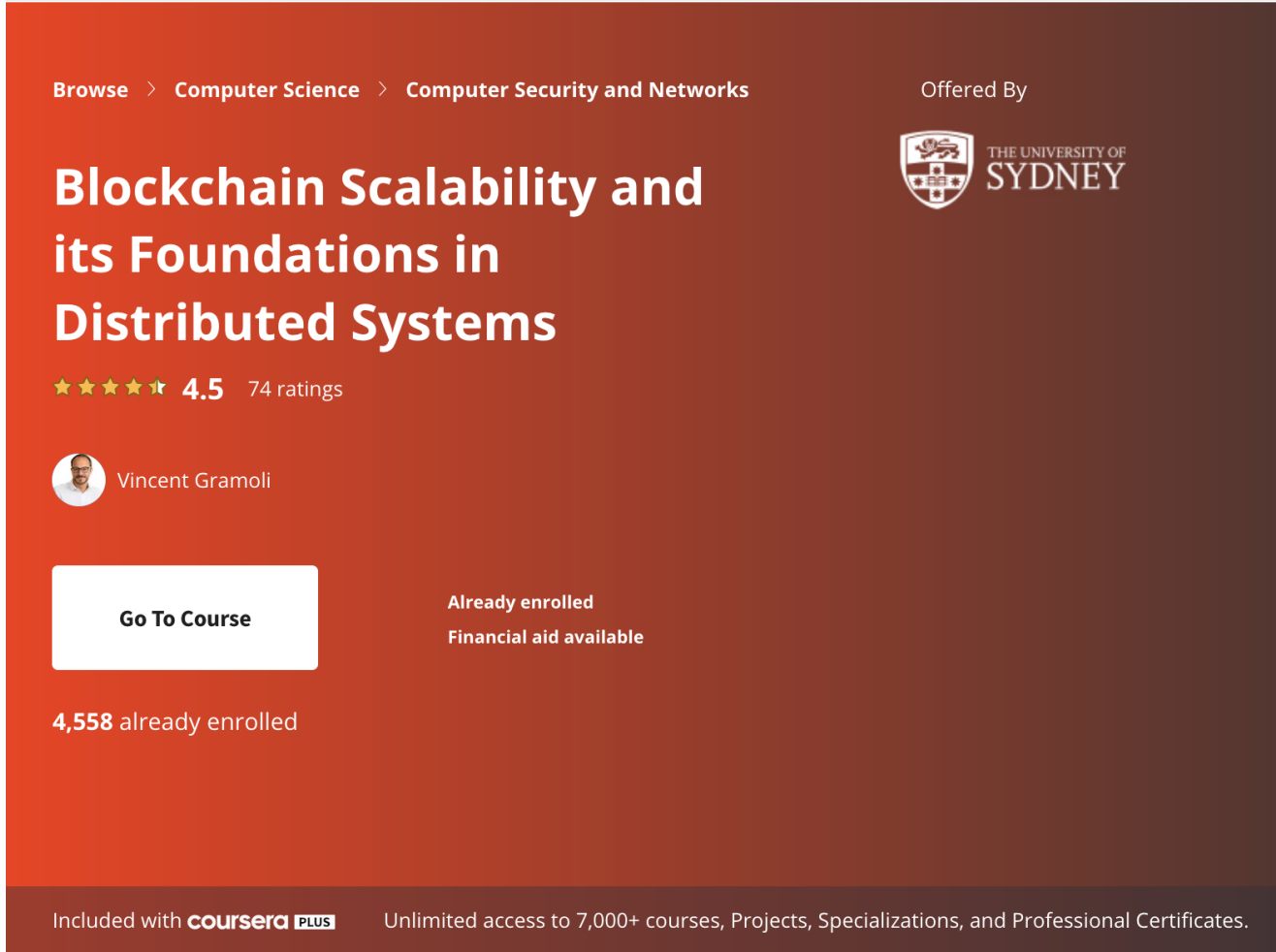
The techniques taught herein consist of enhancing blockchain security and making blockchain scalable by relying on the observation that no blockchain can exist without solving the consensus problem.

https://tinyurl.com/yc22he4v

Online Material: Blockchain Scalability MOOC

Blockchain promises to disrupt industries once it will be efficient at large scale. In this course, you will learn how to make blockchain scale. You will learn about the foundational problem of distributed computing, consensus, that is key to create blocks securely.
https://www.coursera.org/learn/blockchain-scalability

The University of Sydney



What to expect from these blockchain lectures

Fundamental knowledge
- Distributed computing
- Blockchain abstractions

Problem solving skills
- Impossibility results
- Theoretical analysis

Not technology-specific knowledge
- No Solidity v0.8.13 syntax
- Happy to provide pointers to this

Course taught at private and central banks

The Blockchain Scalability MOOC on Coursera is a good overview

Blockchain

Why Studying Blockchain?

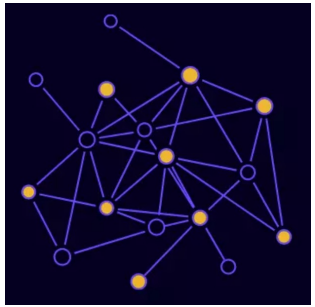
Cybersecurity is part of Australian Science and Research Priorities

Blockchain potential applications span various industries (real estate, aviation, retail, computing, etc.)

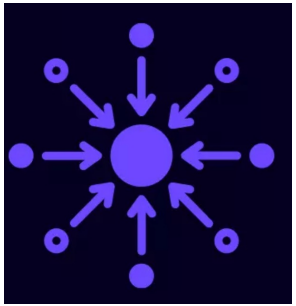
Blockchain was most desired skill on LinkedIn in 2020

Why Studying Blockchain?

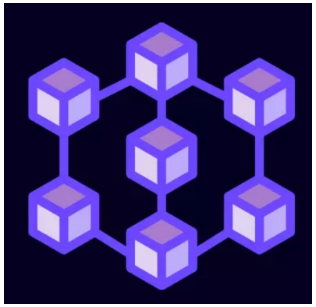
Web3, a decentralized ecosystem supported by blockchains, is a promising replacement to Web2



Web1.0 – desorganised access to information in the 90s



Web2.0 – data sharing, acquired by tech giant in the mid-2000s



Web3.0 – distributed ownership of data – 2015

"Web 2.0 very much still exists in this very centralized model" – Gavin Wood, co-inventor of Ethereum

"Social networking sites that do not allow users to extract the information they put into them is the problem" Tim Berners Lee – Turing Awardee

Payment with Blockchain

Bitcoin is both a currency (lower-case b) and a payment system (big B).

It implements a distributed ledger that is

- Immutable: transactions can only be appended to the ledger
- Pseudonymous: participants are identified by their address (pseudonym)

It uses:

- A Gossip-based protocol for maintaining an overlay
- Peer-to-peer network where peers are both clients and servers (miners)
- Public key cryptosystem, participant sign their transactions
- Consensus for the distributed system to decide a block at index I

What is a blockchain?

The Blockchain abstraction

Let $G = \langle B, P \rangle$ a directed acyclic graph (DAG) where blocks B point to each other with pointers P

$\langle b0, b1 \rangle \in P$ is a pointer from current block b1 to previous block b0

The *pointer* is a representation of a hash of the destination block that the source block contains

The *genesis block* is a special block known initially by all participants

Hash Function

HASH FUNCTION

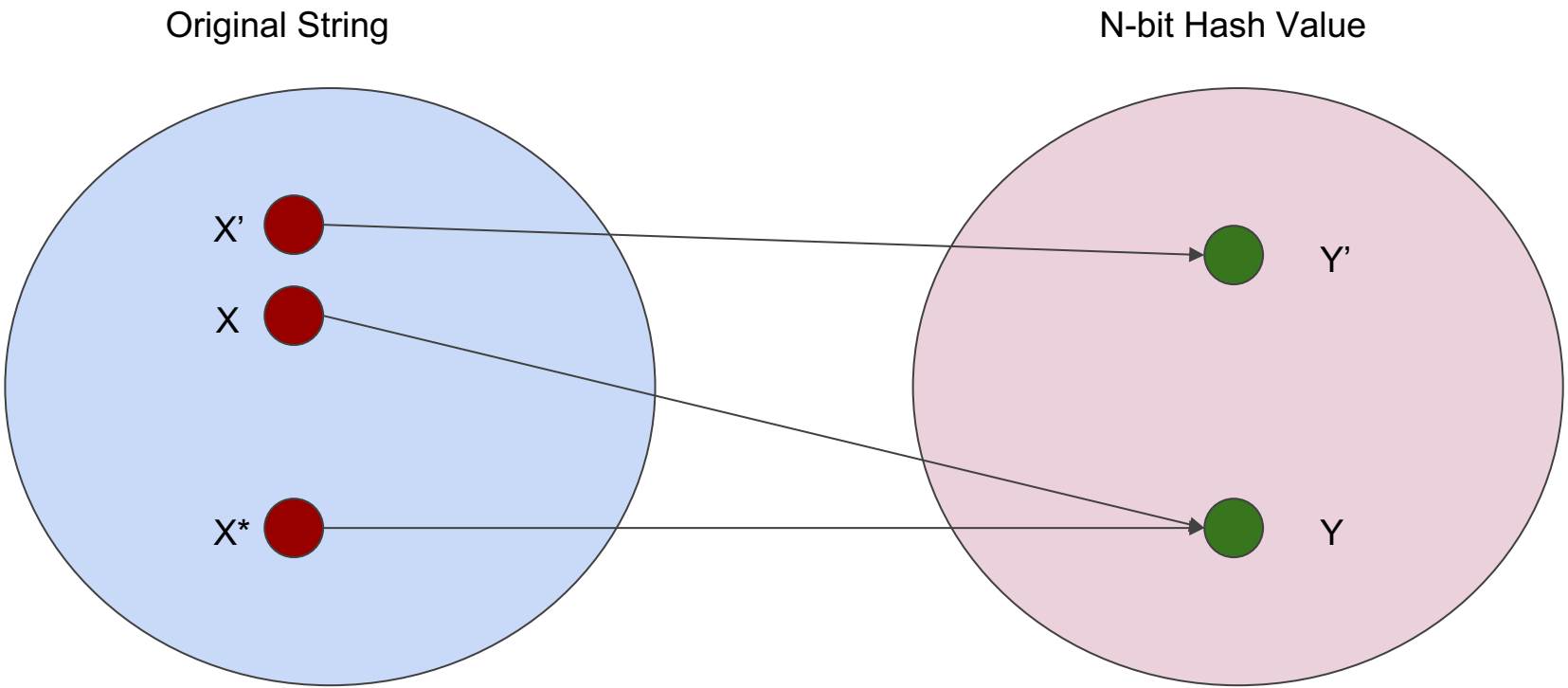


Figure 3.2: Illustration of Hash Function

Cryptographic Hash Function - Ideal Properties

1. **Deterministic**: always same output hash for the same input
2. **Fast**: quick to obtain the hash value
3. **Secure**: hard to find an input that will generate a given ouput hash
4. **Discontinuous**: a small change in input should change hash substantially
5. **Low Collision**: hard to find two inputs that give the same output hash

E.g., SHA256

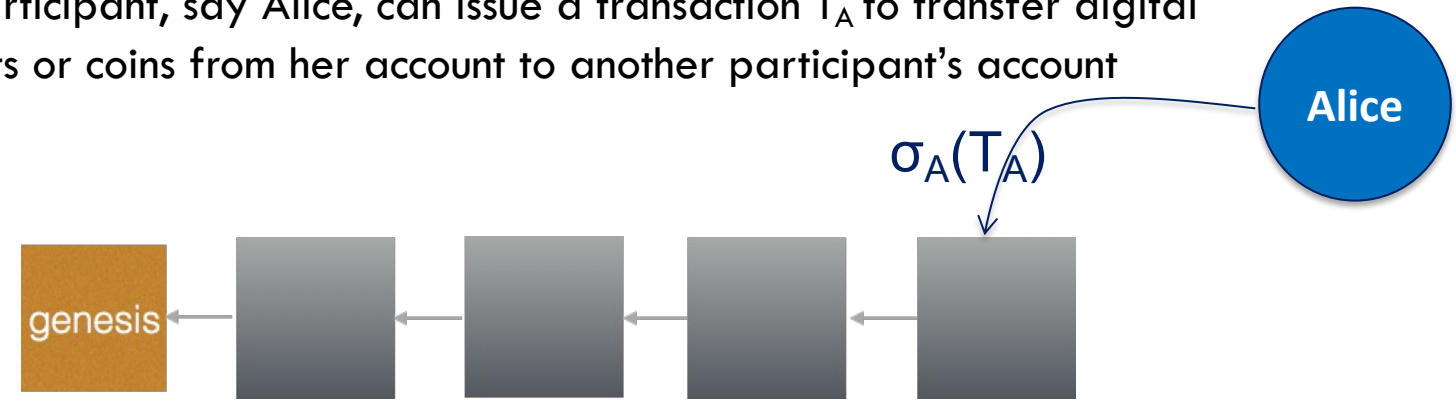
Applications of hash functions

- Github commit identifier
  - You have used a version of a software that works but the most up-to-date version does no longer work.
  - You check the commit hash of the version you use and roll-back the code to the working hash
- Software version check
  - You download an large image and want to know whether it is valid copy and not a version infected with malware or trojan.
  - You ask the developer to provide an MD5 hash of the image and compare it to the result of applying the hash function to the image.
- Message digest
  - Someone sends you a message along with a signed hash.
  - You compute the hash of the message content and compare it to the signed hash to detect tampering.

Signatures

Blocks contain transactions

A participant, say Alice, can issue a transaction T_A to transfer digital assets or coins from her account to another participant's account

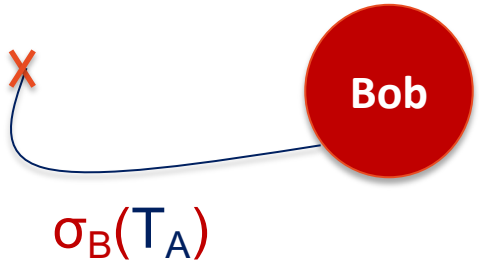


She simply signs her transaction T_A with her signature σ_A, then the signed transaction $\sigma_A(T_A)$ has to be stored in a block of the blockchain

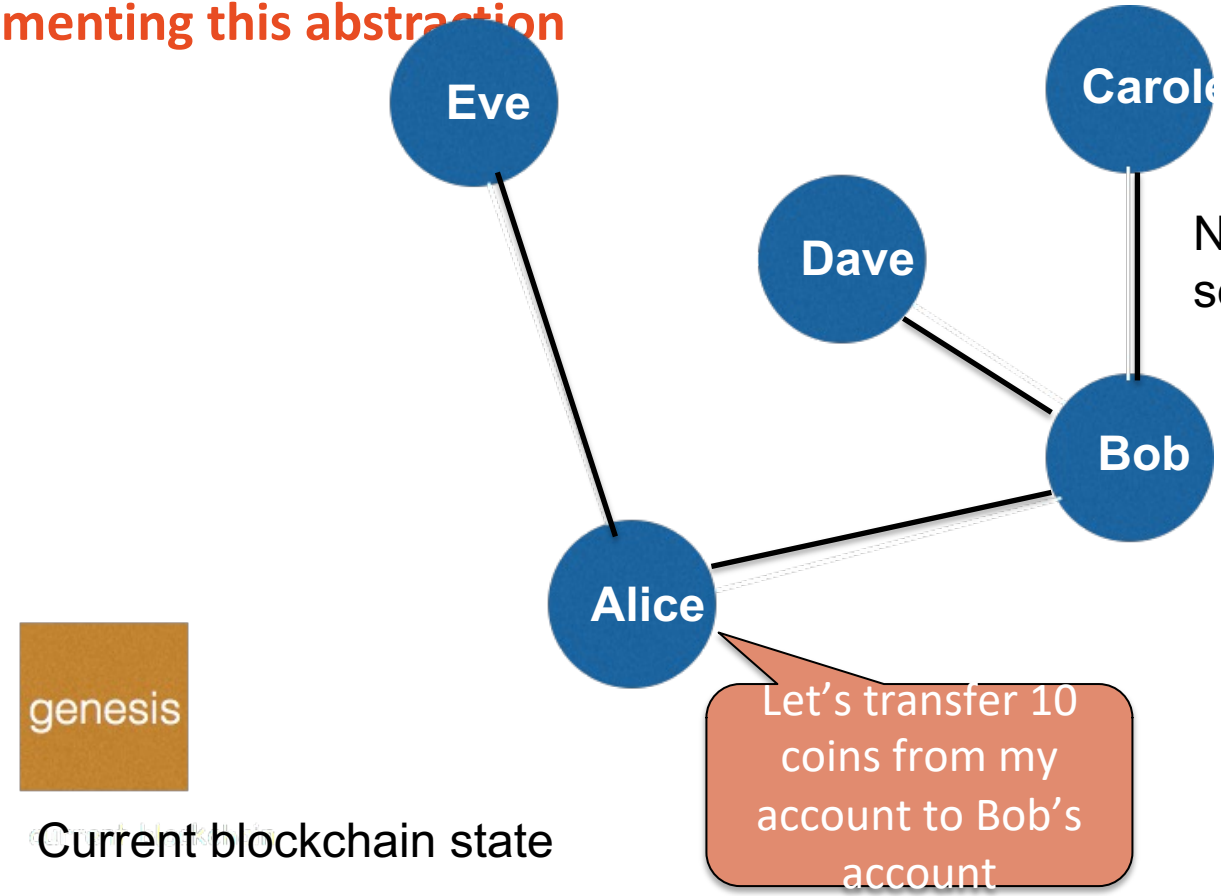
Transactions are securely signed

Because the issuer has to sign its transaction with its private key no other participant can withdraw his coins without his consent

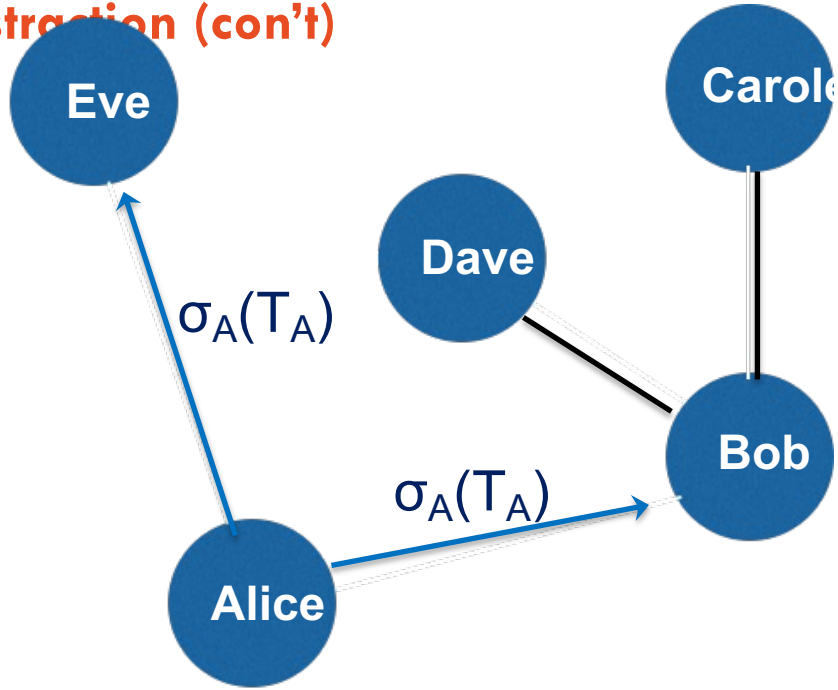
Any transaction not signed by the issuer will not be stored in a block



Implementing this abstraction



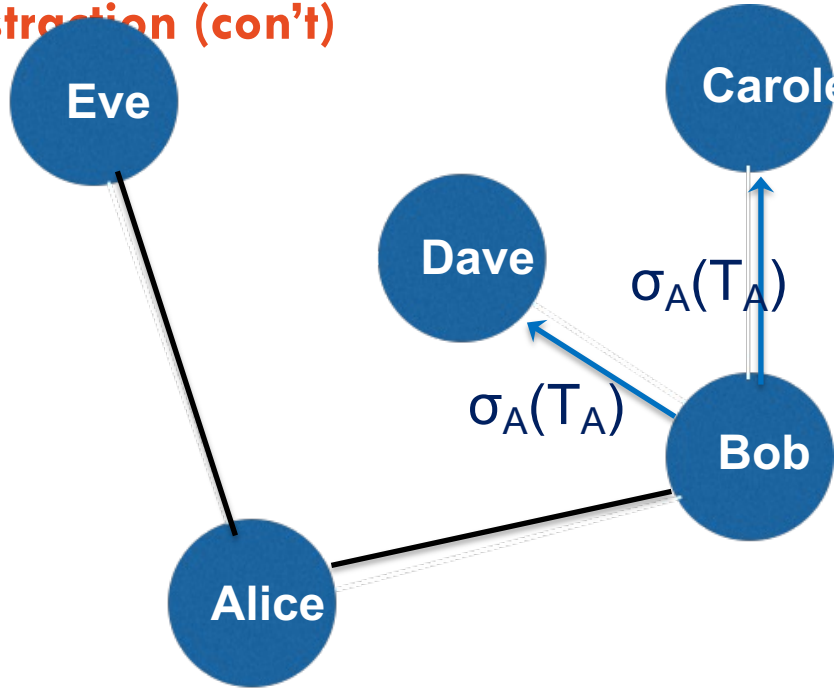
Implementing this abstraction (con't)



genesis

Current blockchain state

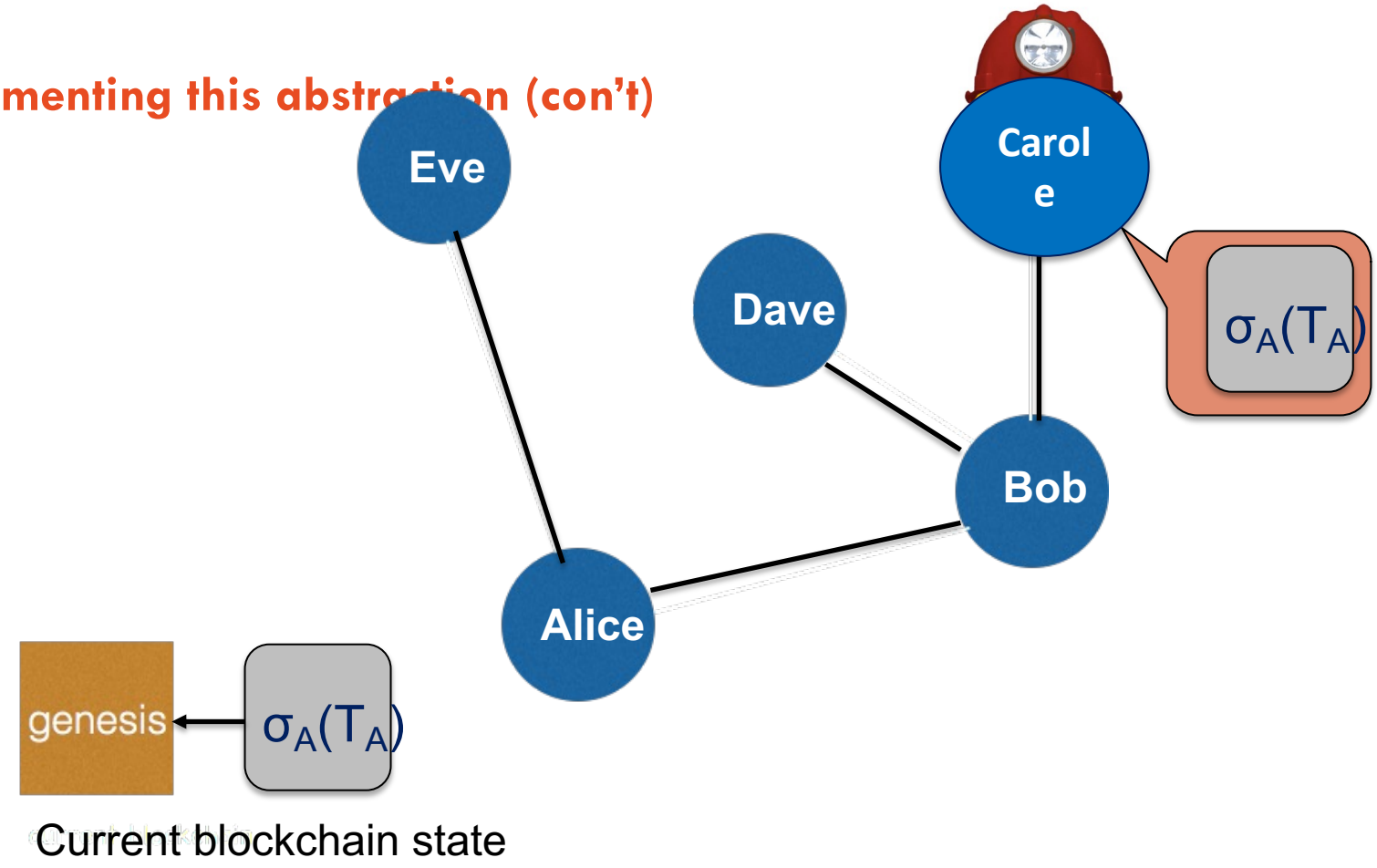
Implementing this abstraction (con't)



genesis

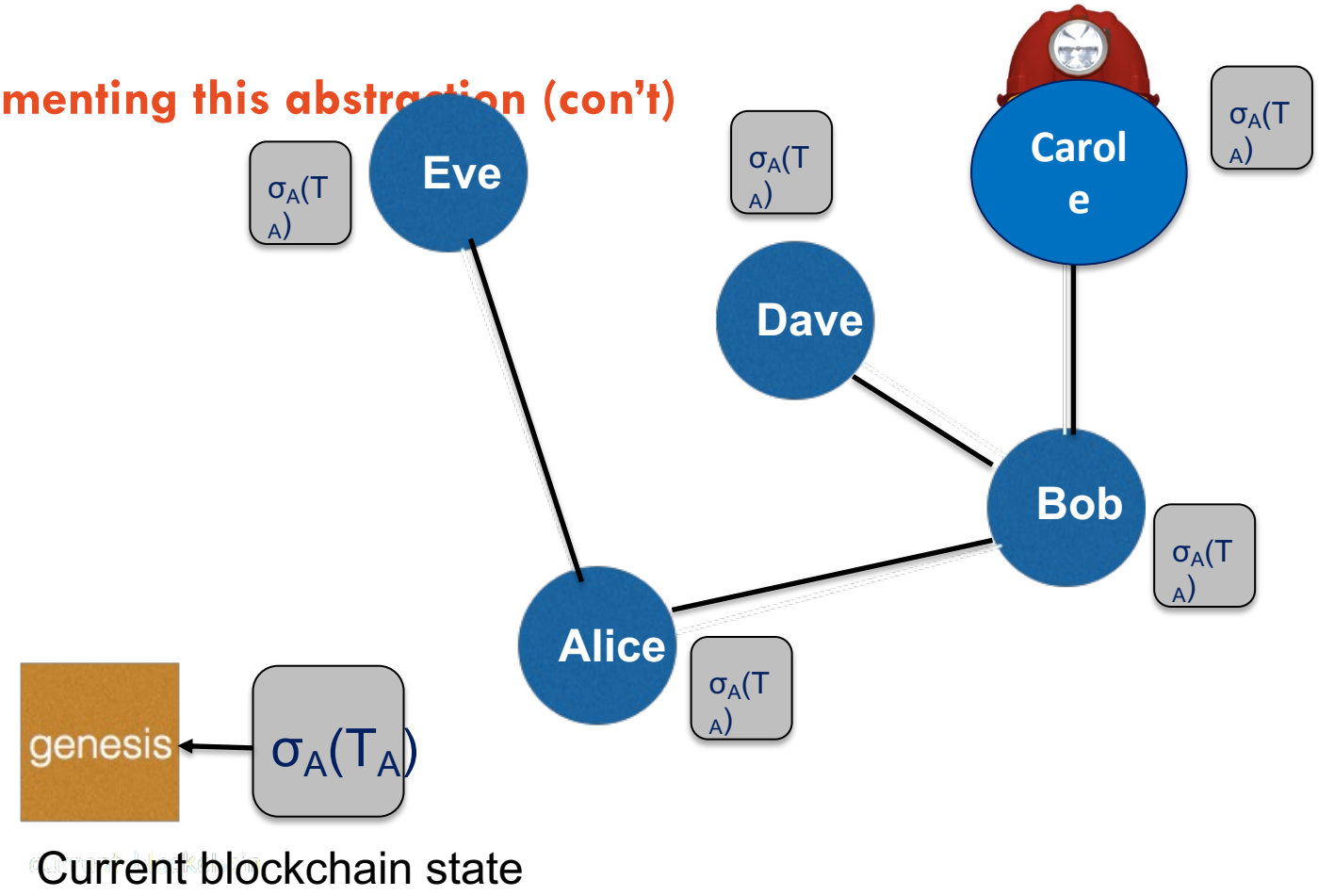
Current blockchain state

Implementing this abstraction (con't)



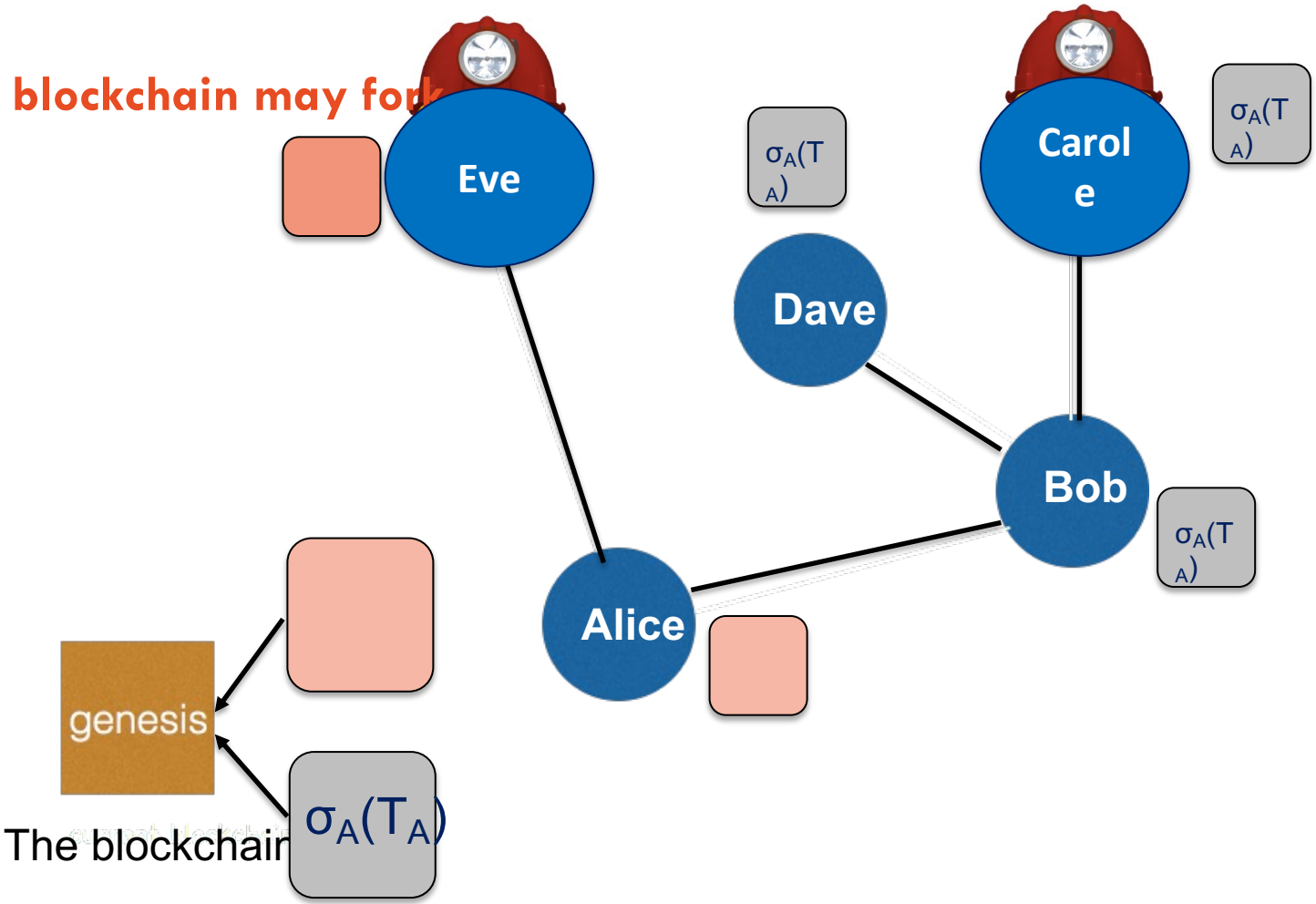
Current blockchain state

Implementing this abstraction (con't)

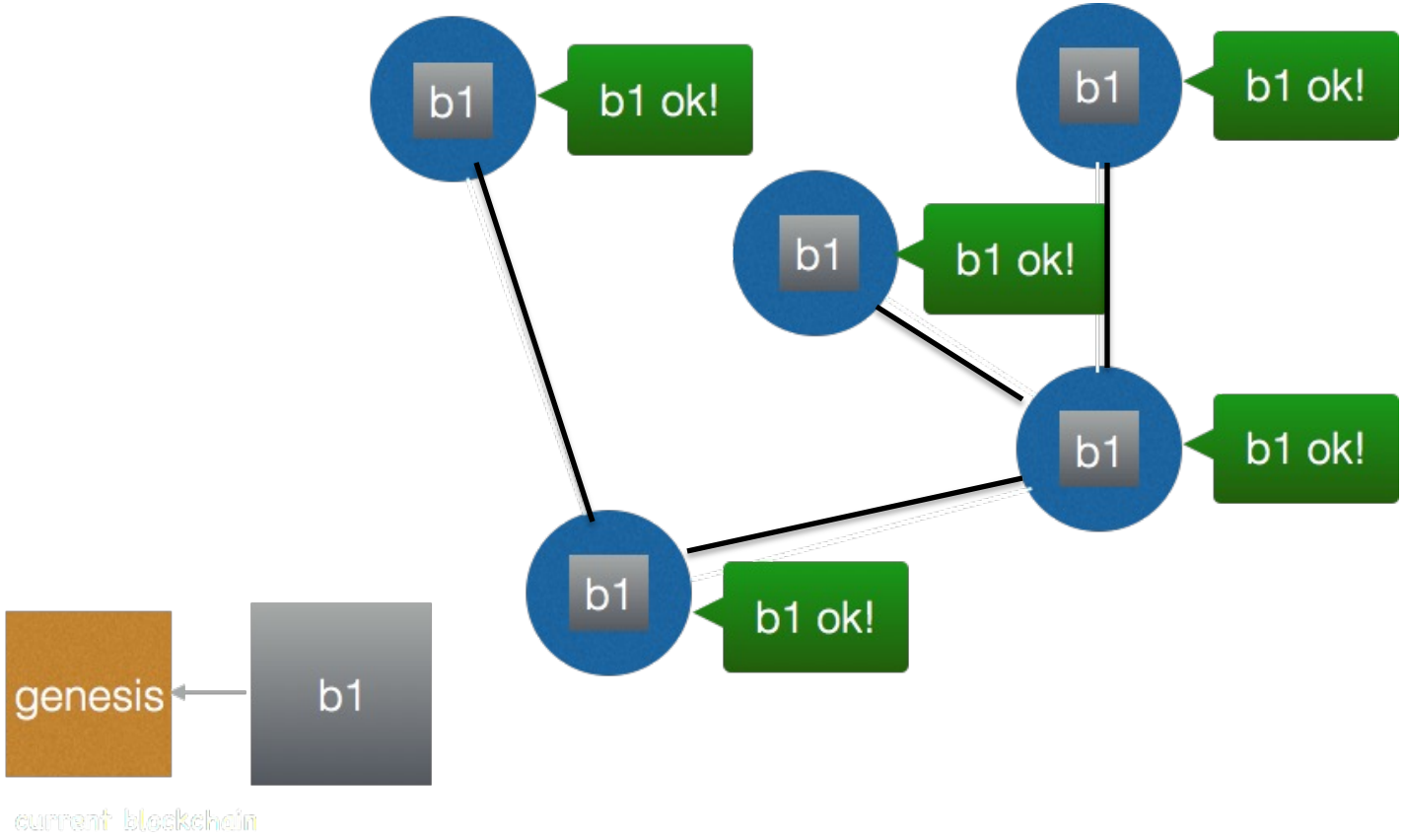


Current blockchain state

The blockchain may fork



Distributed implementation of a blockchain



History of Blockchains

Before Bitcoin

Cypherpunks and digital currency

- **1975 –** A symmetric key cryptosystem is published as the Data Encryption Standard.
- **1983** – David Chaum proposes blind signatures for a payment system that offers auditability and privacy.
- **1989** – He founds DigiCash Inc., an electronic money corporation, that sold its assets to eCash.
- **1992** – Eric Hughes, Timothy May and John Gilmore meet on a monthly basis in San Francisco and the cypherpunk mailing list is created.

[IBM75] IBM, "Data Encryption Standard", US Federal Register, 1975.

[Cha83] D. Chaum, "Blind Signatures for Untraceable Payments", Advances in Cryptology, Springer 1993. https://link.springer.com/chapter/10.1007/978-1-4757-0602-4_18

Before Bitcoin

"Privacy in an open society requires anonymous transaction systems. Until now, cash has been the primary such system. An anonymous transaction system is not a secret transaction system."

Eric Hugues, 1993

https://medium.com/coinmonks/a-cypherpunks-manifesto-a-definition-of-privacy-66c36f99e940

Before Bitcoin

Proof-of-work and double spending

- **1992** – Cynthia Dwork and Moni Naor require a user to compute a moderately complex function in order to avoid abuse of resource usage.
- **1997** – Adam Black proposes a hashcash mint, a partial hash collision generator, on the cypherpunk mailing list. The idea is to use partial hashes to be made arbitrarily expensive to compute but verified instantaneously to detect double spending: http://www.hashcash.org/papers/announce.txt
- **2002** – Adam Black publishes the Hashcash technical report.
- **2004 -** Hal Finney introduced "reusable proofs of work" and offers a public implementation of a transparent server whose integrity can be checked. https://cryptome.org/rpow.htm

[DN92] Dwork and Naor, "Pricing via Processing or Combatting Junk Mail", CRYPTO'92.

https://dl.acm.org/citation.cfm?id=705669

[Bla02] Black, "Hashcash - a denial of service counter-measure", Cypherspace, TR 2002.
http://www.hashcash.org/papers/hashcash.pdf

Before Bitcoin

Bit gold and B-money

- **1998 –** Wei Dai proposes b-money a system where users can transfer money between accounts represented as public keys by broadcasting messages
- **2005 –** Nick Szabo proposed Bit Gold crypto-currency that applies a chain of hash to guarantee immutability https://nakamotoinstitute.org/bit-gold/
- **2008** – Satoshi Nakamoto writes Bitcoin, a protocol to transfer a cryptocurrency peer-to-peer without the need of central intermediary.

[Sza05] Szabo. Bit gold. Technical report, 2005.

[Nak08] Nakamoto. *"Bitcoin: A Peer-to-Peer Electronic Cash System" (PDF). March 2014.*

Before Bitcoin

Bit gold

[S

[HS

[Nak

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to the network, they'll generate the longest chain and outpace attackers. The requires minimal structure. Messages are broadcast on a best effort leave and rejoin the network at will, accepting the longest of what happened while they were gone.

between

hash to

peer-to-

financial institutions serving as
works well enough for
ust based model.
tions cannot
the

After Bitcoin

- **2009** – Blockchain v1.0: bitcoin is released as an open source software.
- **2013** - Vitalik Buterin describes Ethereum (Blockchain v2.0) based on generic crypto-puzzle and offers services beyond cryptocurrency.
- **2015** – Gavin Wood writes the Ethereum yellow paper.
- **2016** – A Decentralized Autonomous Organization (DAO) was proposed as a smart contract, raised $150M and got hacked.
- **2017** – More secure models of blockchains appear (e.g., AlgoRand, Red Belly Blockchain).

[Woo15] – Etehreum: A Secure and Decentralized Generalized Transaction Ledger

[GHM17] – Gilad, Hemo, Micali, Vlachos, Zeldovich. AlgoRand: Scaling Byzantine Agreements for Cryptocurrencies. SOSP'17.

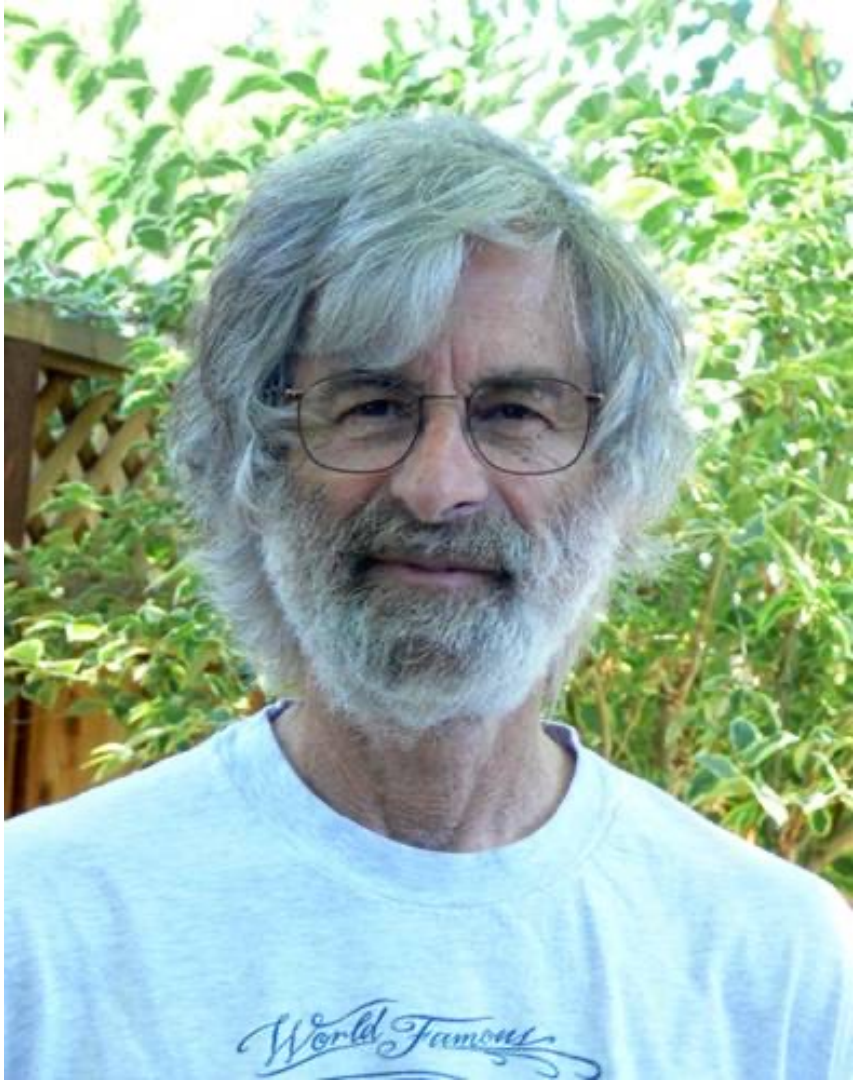
[Gra17] – The Red Belly Blockchain. Personal Presentation. MIT, USA.

Failures

Failures

"A distributed system is a system where I can't get my work done because a computer has failed that I've never even heard of."

Leslie Lamport



Failure taxonomy

- *Failure*: a system *fails* when it cannot meet its promises
- *Crash failure*: a machine halts but is working correctly until it halts
- *Arbitrary (Byzantine) failure*: a machine may produce arbitrary responses at arbitrary times or may not send responses
- The most serious failure is arbitrary (a.k.a. Byzantine), it can be of any type.

Failures

Motivations for arbitrary (Byzantine) failures:

- Malfunctioning hardware
- Buggy software
- Malicious attacks

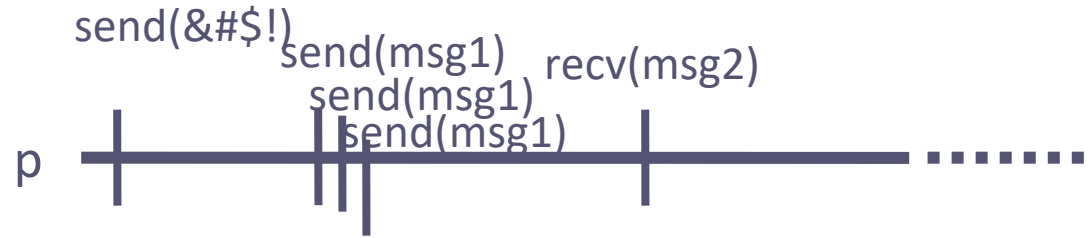
In blockchains, there is an incentive to steal digital assets, so malicious attacks are expected.

Failures

We will focus on

- *Crash failures:*

- *Byzantine failures:*



- A *correct* process never fails

Coping with failures

Consensus

Why Consensus?

The consensus problem has a lot of applications:

- With consensus you can decide on a leader
- Coordinate machines to do an attack

Similarities with other problems of distributed computing:

- Totally ordered broadcast
- Atomic commit
- Terminating reliable broadcast
- …

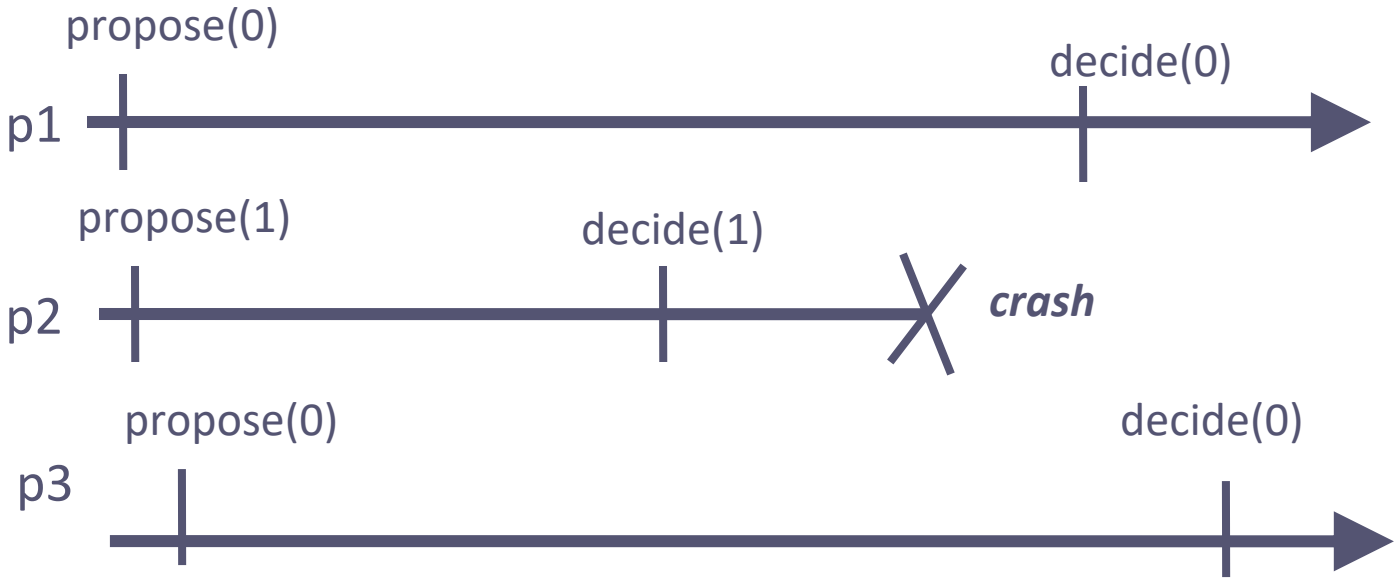
The Consensus Problem

In the consensus problem, the processes propose values and have to agree by deciding one among these values

Consensus Definition 1:

1. *Agreement*: No two correct processes decide differently
2. *Termination*: Every correct process eventually decides
3. *Validity*: Any decided value is a value proposed

The Consensus Problem (con't)



Consensus and Blockchain

Why Consensus in blockchain?

To construct a blockchain, distributed processes have to agree on a unique block at a given index

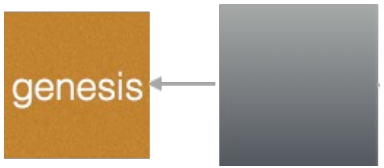
Assume all correct processes know the genesis block at index 0:

Why Consensus in blockchain?

To construct a blockchain, distributed processes have to agree on a unique block at a given index

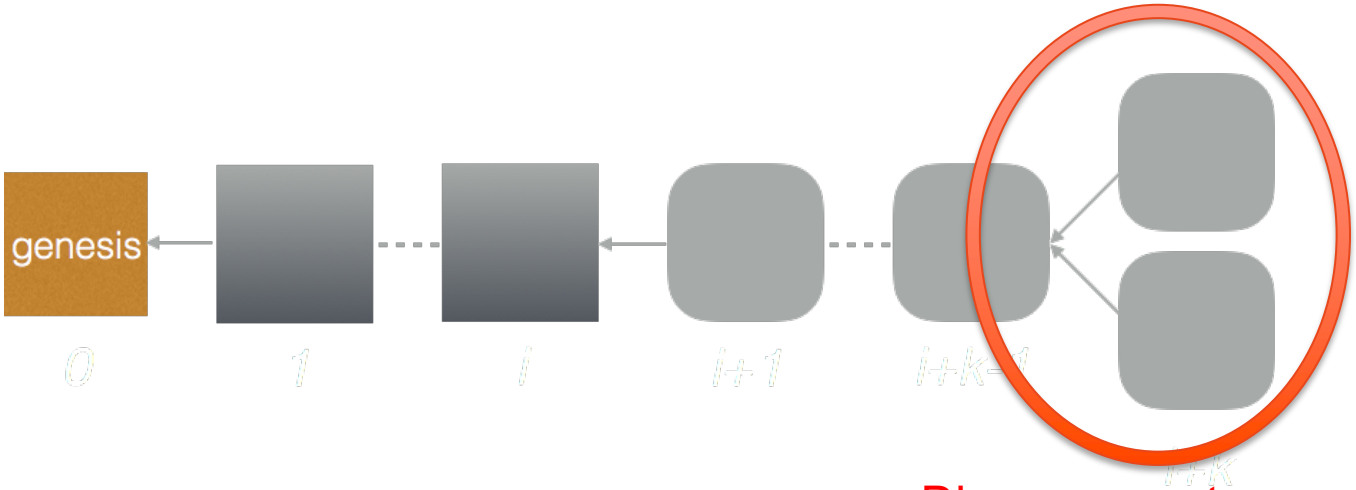
Assume all correct processes know the genesis block at index 0:

Then, they run consensus at index 1 to decide upon a new block:



Why Consensus in blockchain?

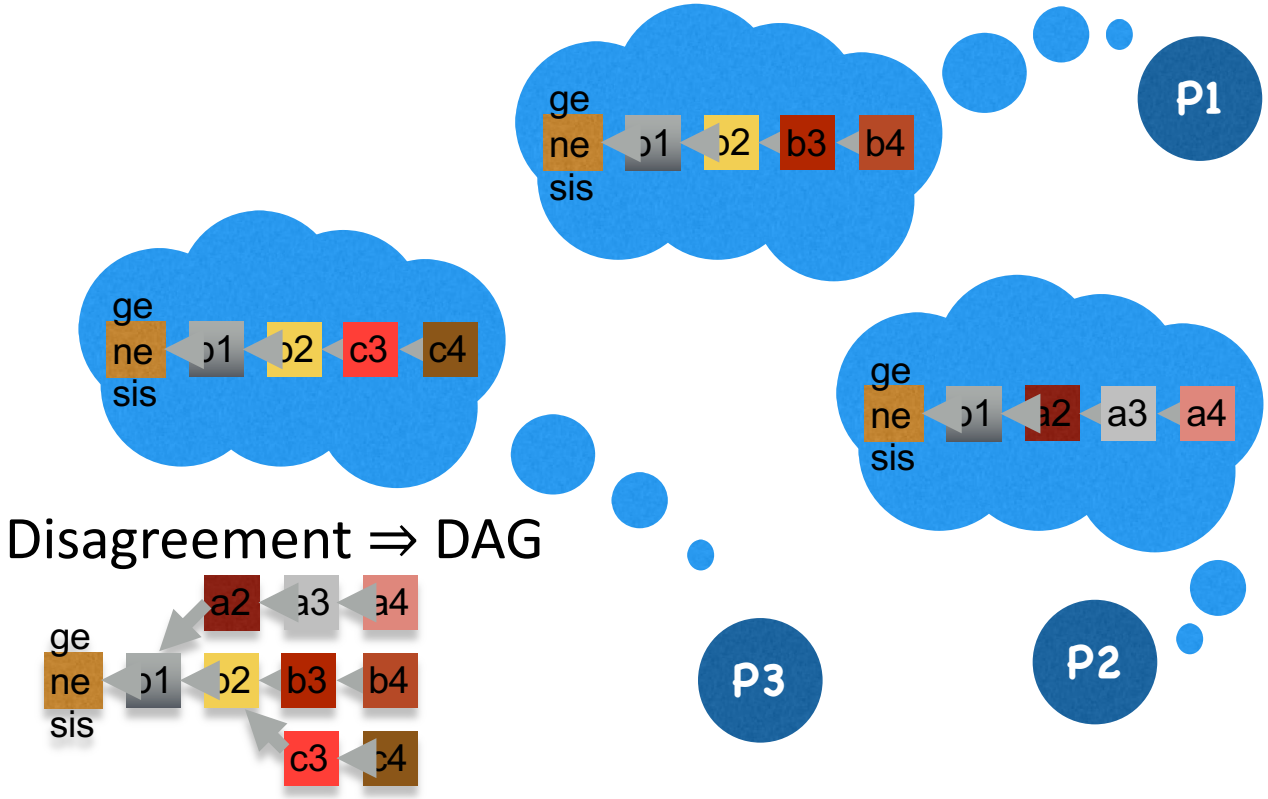
Consensus is necessary to totally order the blocks, maintaining the *chain*



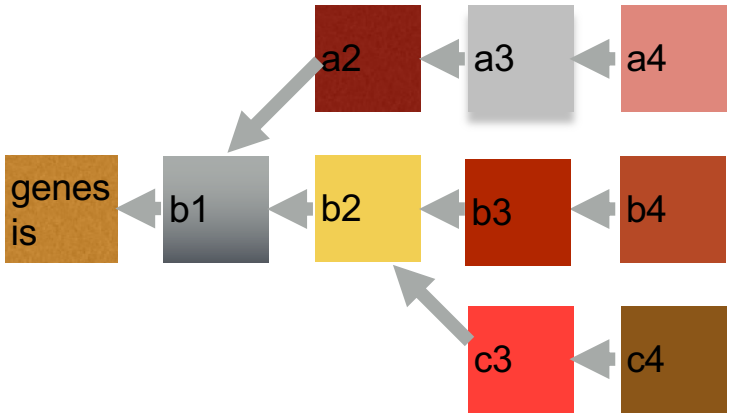
Disagreement ⇒ no chain but a DAG

Double Spending

Disagreement



Directed Acyclic Graph (DAG)



Double Spending

Let Alice be a Byzantine process willing to steal assets (coins)

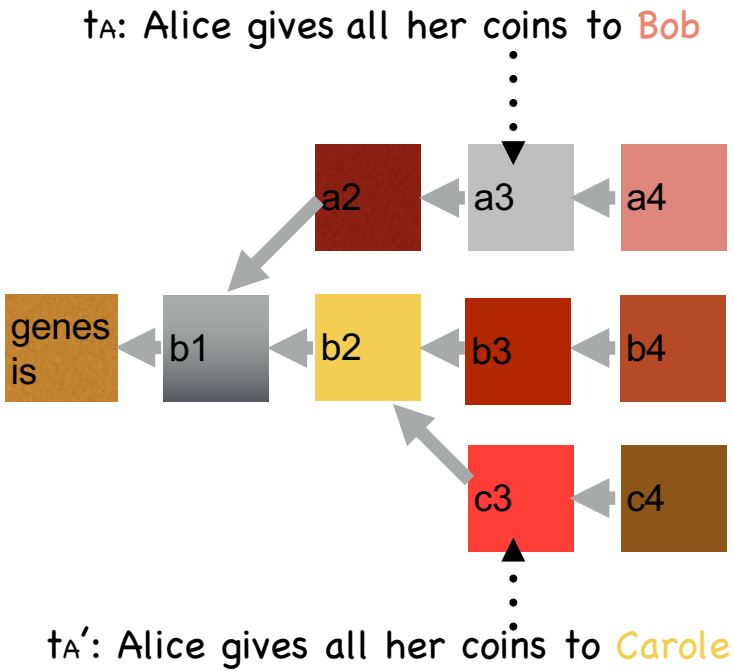
Let Bob and Carole be two merchants selling real products in exchange of coins

Alice issues:

- a transaction t_A spending all her coins with Bob
- a transaction $t_{A'}$ spending all her coins with Carole

If both Bob and Carole observe their transaction in the blockchain, they will ship the "supposedly paid" product to Alice.

Double Spending (con't)

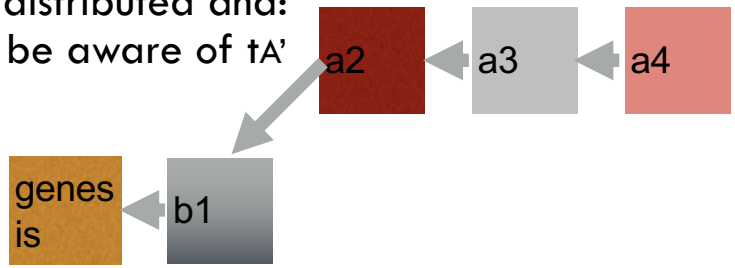


Double Spending (con't)

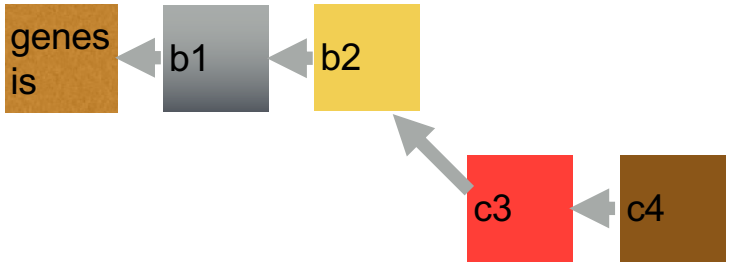
The problem is that t_A and $t_{A'}$ conflict

But the system is distributed and:

- Bob may not be aware of $t_{A'}$



- while Carole may not be aware of t_A



Blockchain Failures

Failure model

Model

- n nodes in the system
- f are faulty

Because blockchains protect ownership, there is an incentive for an attacker to steal the goods of others

- The fault model is Byzantine (i.e., arbitrary)

Solution for Byzantine consensus

Limiting the number f of failures is key to solving consensus

There are solutions when $f < n/3$ [CL02]

[CL02] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst., 20(4):398-461, Nov. 2002.

Theorem

There is no solution to the Byzantine consensus if f=1 and n=3.

Proof intuition.

Consider three processes p1, p2, p3. Byzantine process p2 can play p1 and p3 off against each other, telling p1 that p3 is Byzantine and p3 that p1 is Byzantine. Since p1 is telling p3 the same thing about p2 that p2 is saying about p1, p3 cannot tell the difference and does not know who to believe.

Theorem

There is no solution to the Byzantine consensus if $f \geq n/3$.

Proof.

By reduction of the impossibility result of when $f \geq n/3$ to when $f=1$ and $n=3$.

Suppose that there were an algorithm that solved Byzantine agreement with $n=3f$ processes. Group the processes into groups of size f. Let each of the $n=3$ processes simulate one group of processes receiving the same input. Then we get a protocol for $n=3$ and $f=1$, which is a contradiction.

Proof of Work

Sybil attack

A *Sybil attack* is an attack where a malicious user forges identities

It is named after the subject of the book *Sybil*, a case study of a woman diagnosed with dissociative identity disorder.

Some solutions [CL02] are prone to Sybil attacks where an adversary generates fake faulty nodes to have $f \geq n/3$ ⇒consensus impossible.

Miners

Specialised peers, called *miners*, receive a reward for verifying transactions provably solving a cryptopuzzle [Bla02] to append a new transaction block to the blockchain.

Cryptopuzzle: given a block and a threshold, a miner repeatedly:

- selects a nonce and
- applies a pseudo-random function to this block and the selected nonce

…until it obtains a result lower than the threshold.

[Bla02] A. Black, "Hashcash - a denial of service counter-measure", Cypherspace, TR 2002. http://www.hashcash.org/papers/hashcash.pdf

Proof-of-Work

The nonce is included in the block, this is the *proof-of-work* [DN93]*:*

- finding the nonce takes time, but
- validating that the nonce is correct is easy.

Everyone can verify that someone lied about having solved the puzzle

[DN93] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '92, pages 139-147, 1993.

Cryptopuzzle

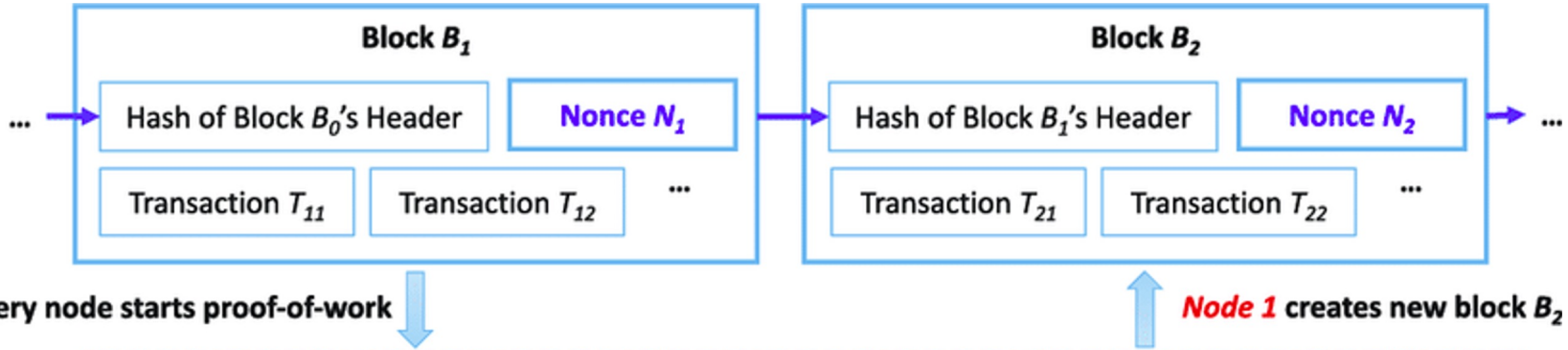
The Bitcoin Protocol - Proof of Work

All nodes on the network try to create a block. There is a “lottery” that selects one node every ten minutes or so to be the “miner” of the next block.

The lottery is implemented by having all computers try, in parallel, to solve a hard cryptographic puzzle called the “Proof of Work” puzzle.

The difficulty of the puzzle is automatically adjusted over time so that it is harder to solve when there are more nodes - on average 10 minutes to pick a winner.

PROOF OF WORK



Node 1

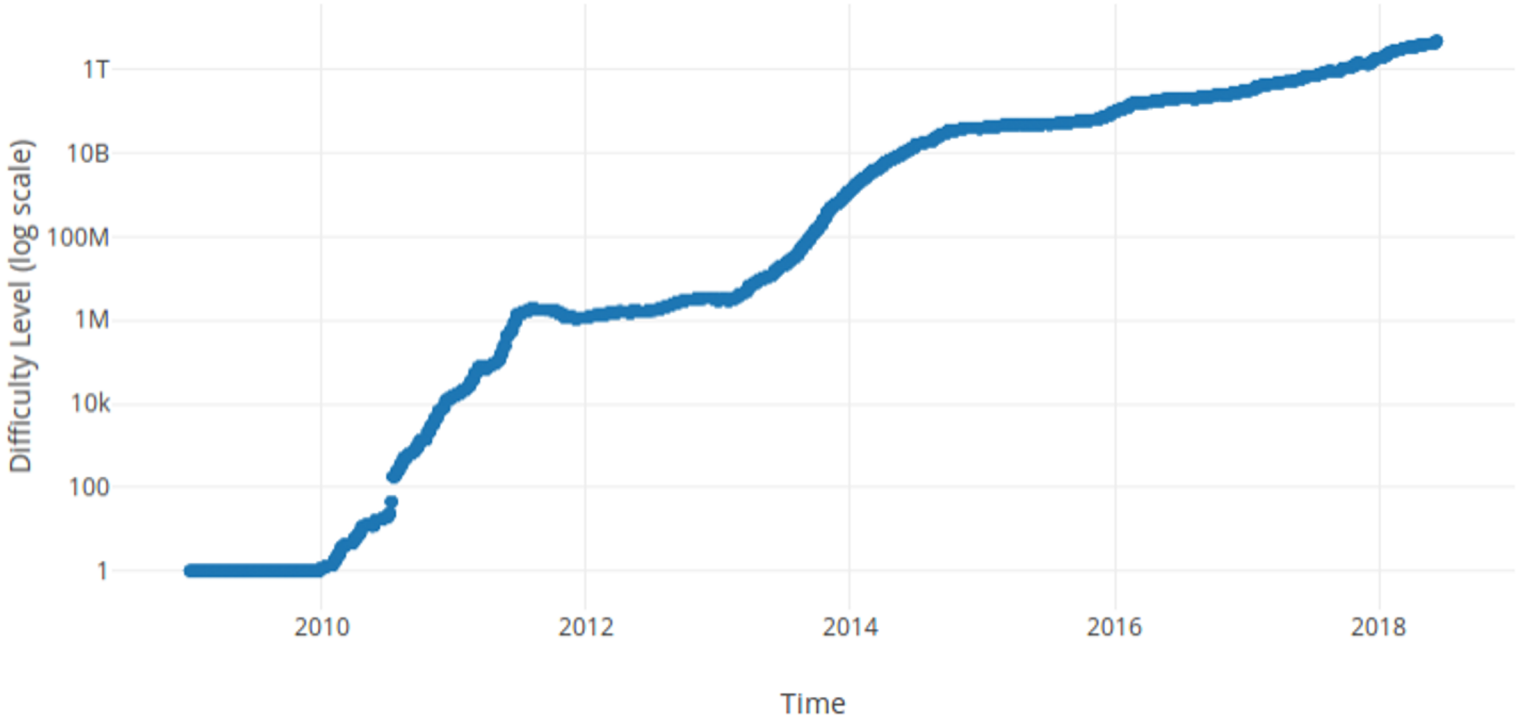
...
Nonce N_2 = "7C 4D DB 29" → Hash of B_2's header = "2D F8 8E 32 ... 10 9A FE 1C", Failed (time = 10:14:20)
Nonce N_2 = "7C 4D DB 30" → Hash of B_2's header = "41 2A B3 DC ... 94 29 AB B5", Failed (time = 10:14:25)
Nonce N_2 = "7C 4D DB 31" → Hash of B_2's header = "00 00 4F 65 ... 2F ED 31 09", Succeeded (time = 10:14:30)

Node 2

...
Nonce N_2 = "61 0A 3F 3A" → Hash of B_2's header = "A8 C7 08 C9 ... 3D F1 A2 F9", Failed (time = 10:14:23)
Nonce N_2 = "61 0A 3F 3B" → Hash of B_2's header = "2A E9 84 66 ... 91 B4 58 CE", Failed (time = 10:14:28)
Stopped after identifying that *Node 1* has completed proof-of-work at time = 10:14:30

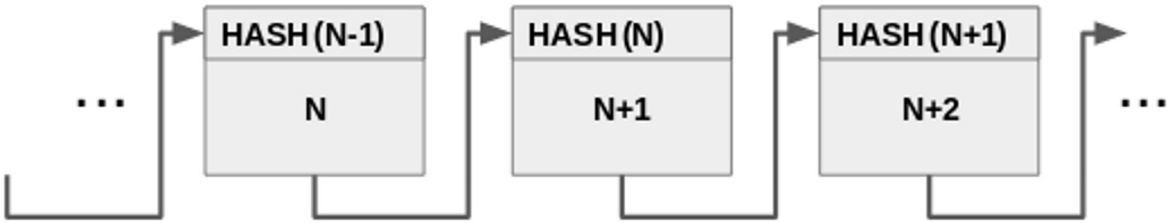
Node 3

...
Nonce N_2 = "99 06 10 13" → Hash of B_2's header = "FB 2F 26 D9 ... 39 F5 C1 0B", Failed (time = 10:14:21)
Nonce N_2 = "99 06 10 14" → Hash of B_2's header = "E2 1C 09 05 ... 25 3E AA CF", Failed (time = 10:14:26)
Stopped after identifying that *Node 1* has completed proof-of-work at time = 10:14:30



Bitcoin difficulty level over time

Proof of Work also helps with immutability

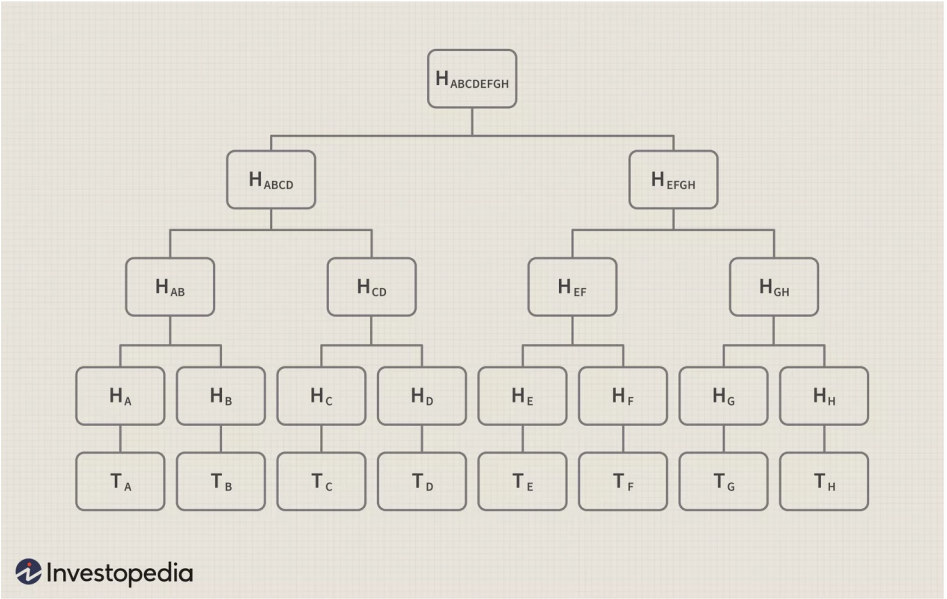


- Each new block added to the chain incorporates the hash (summary) of the previous block
- If someone wants to modify / change a block, they would have to change the hash in the next block, which changes that block, so they would have to change the hash in the block after and so on...
- The proof of work puzzle makes it hard to recreate a block after modification.
- And thus it makes it hard to modify older blocks
- For all intents and purposes, practically, any block that is 4-6 blocks “deep” into the chain cannot be modified, i.e. the blockchain makes transactions in those blocks **immutable**

The Bitcoin Protocol - Merkle Tree

The transaction stored in each block are stored in a data structure called the Merkle tree that makes it computationally efficient to confirm membership.

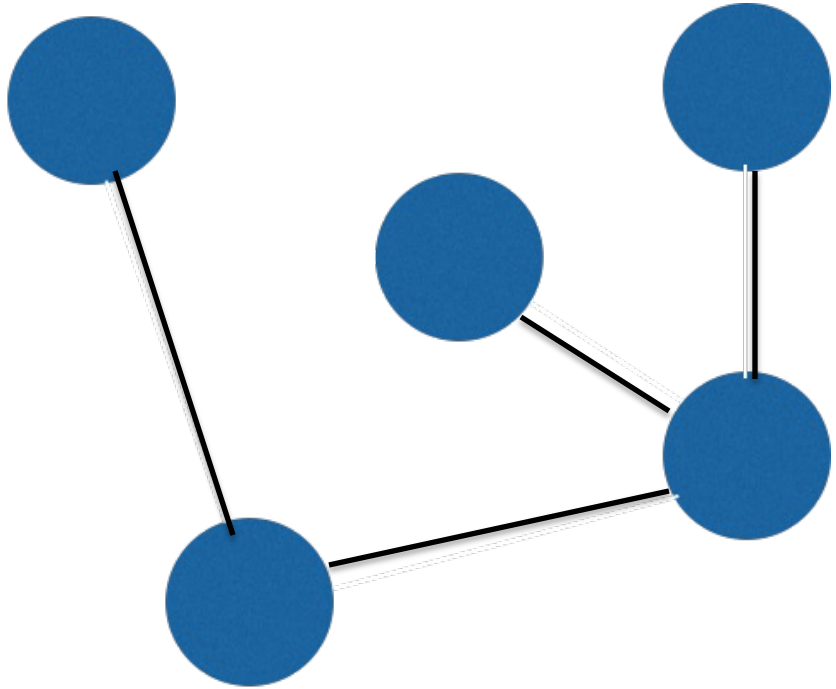
Verifying membership of a leaf node requires a log number of computations



E.g. given H_{root}; to verify T_D is present, only need H_C, H_{AB}, H_{EFGH}

Execution

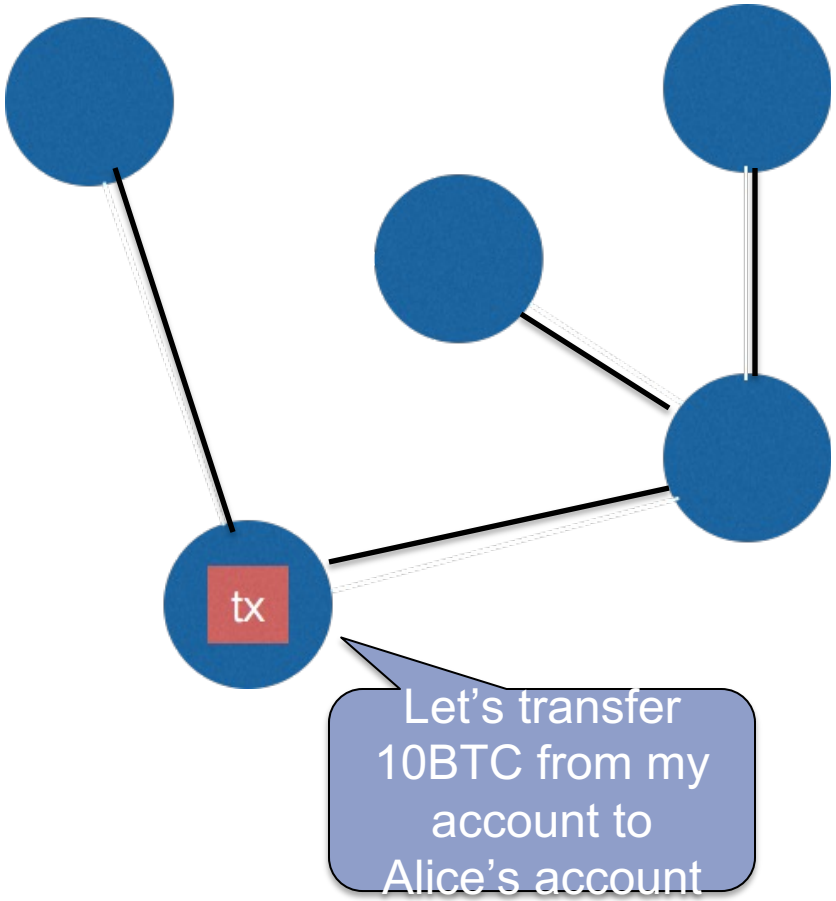
Gossip-based protocol



genesis

Current blockchain state

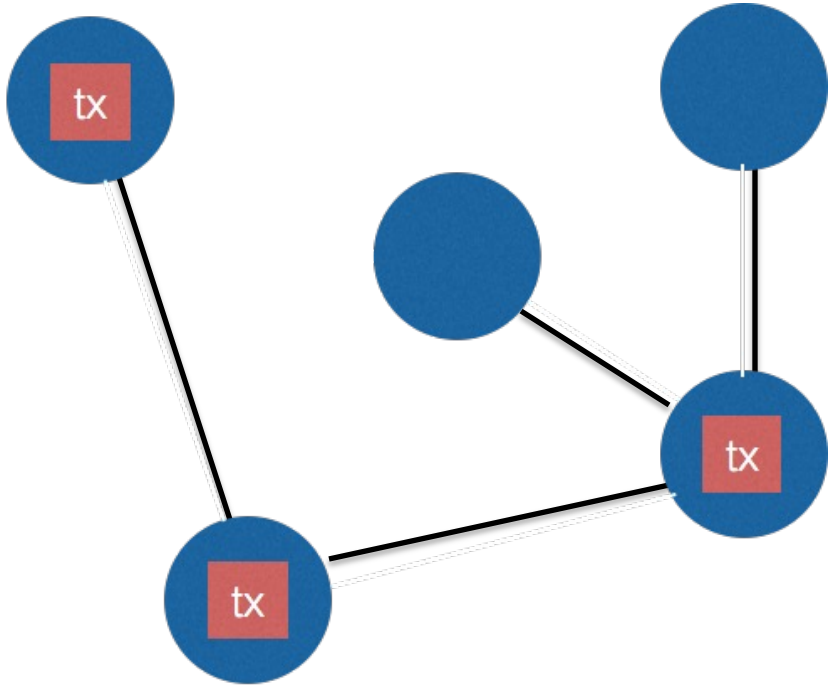
New transaction



genesis

current blockchain

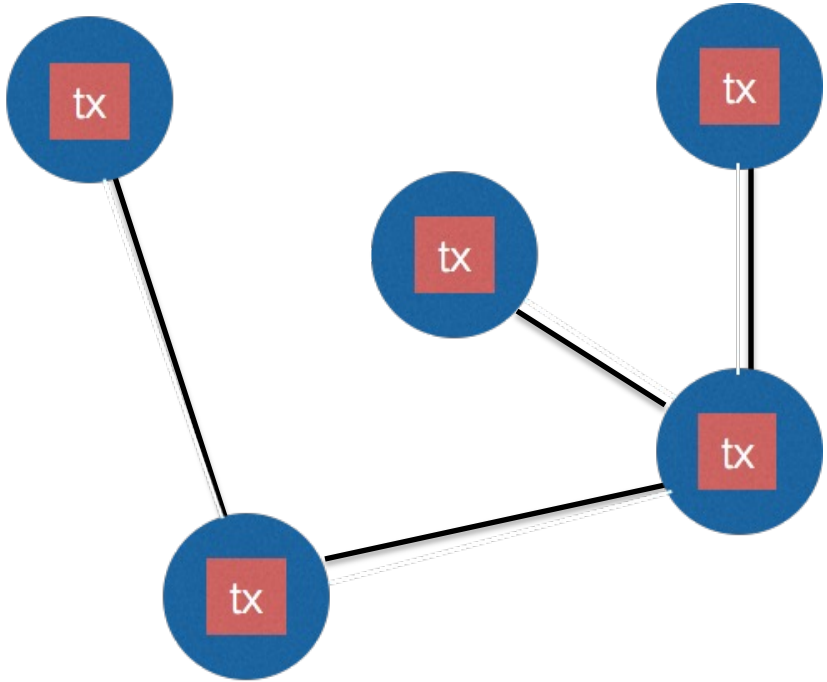
Broadcast



genesis

current blockchain

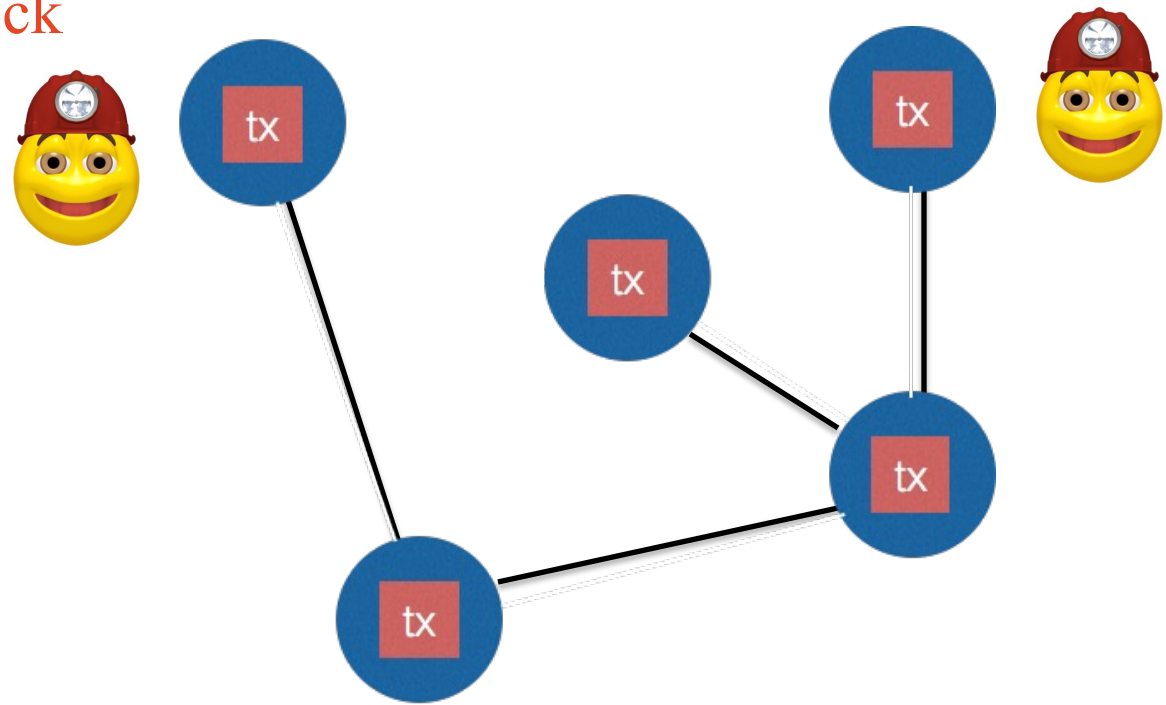
Broadcast



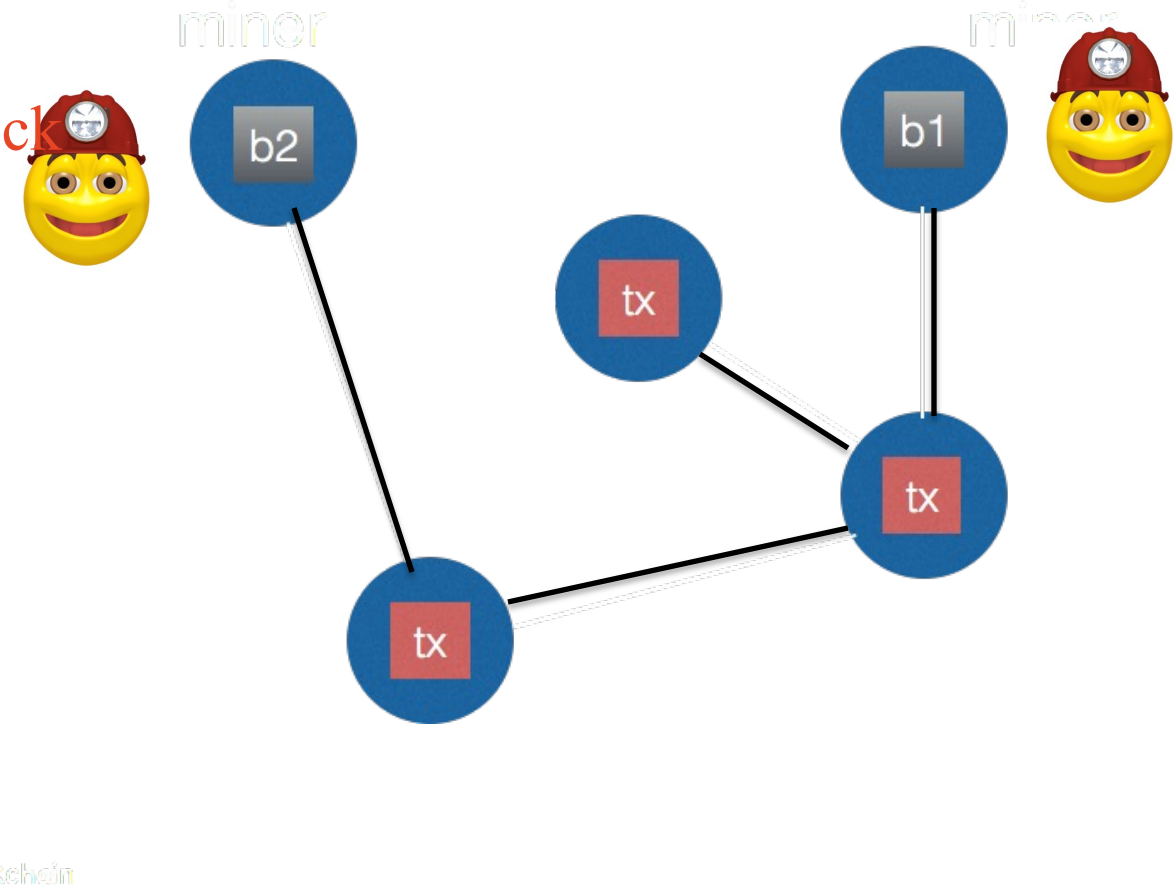
genesis

current blockchain

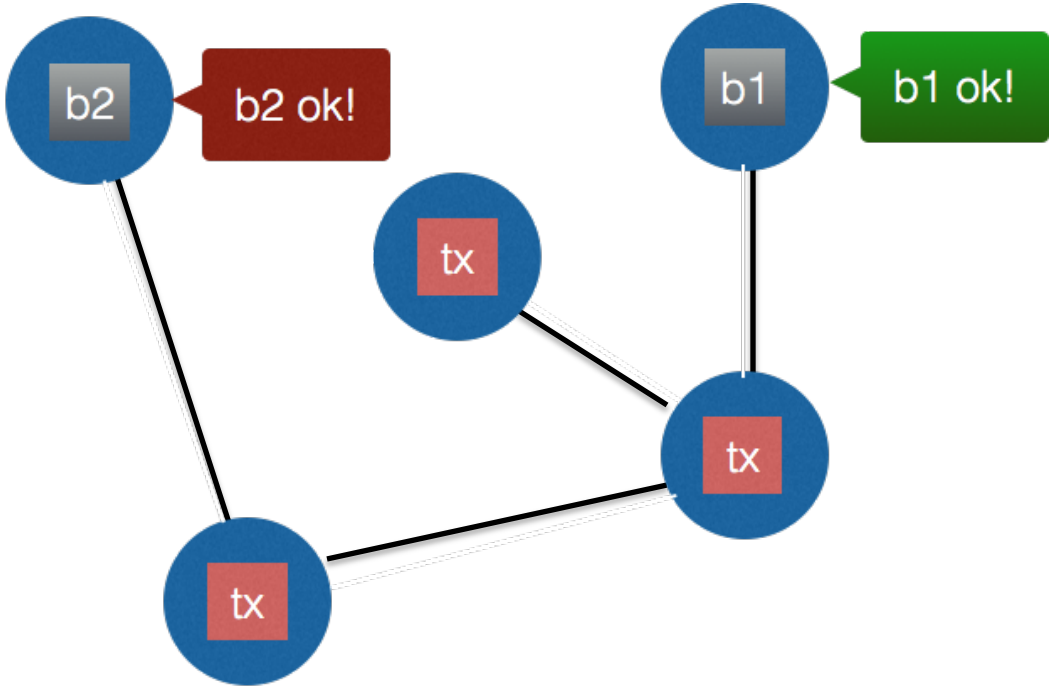
Mining a block



Mining a block



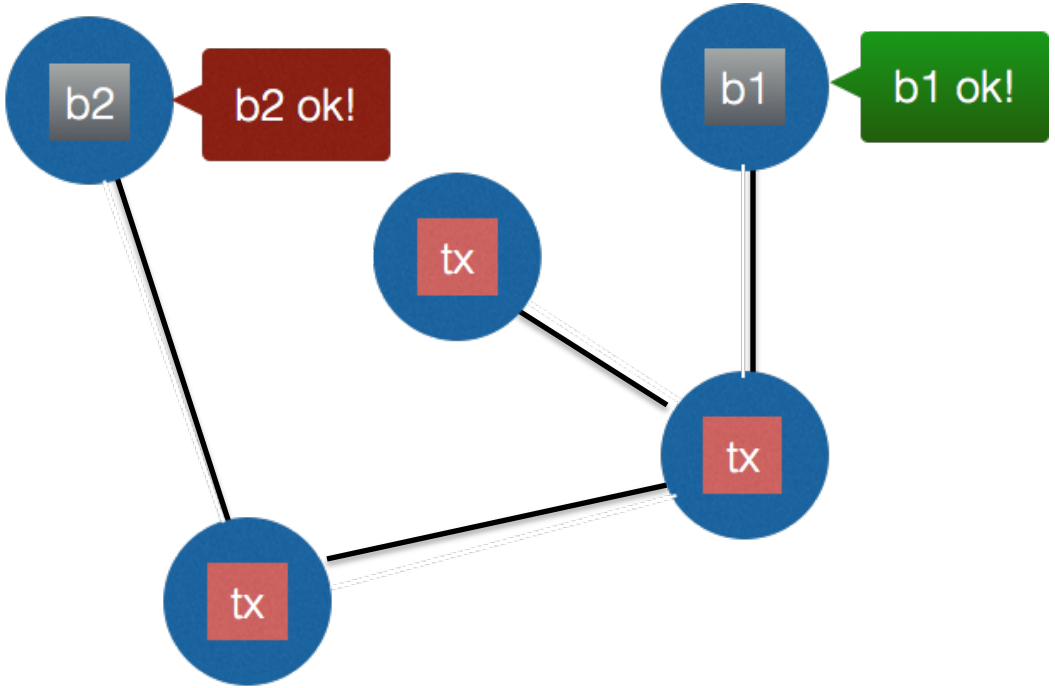
Consensus



genesis

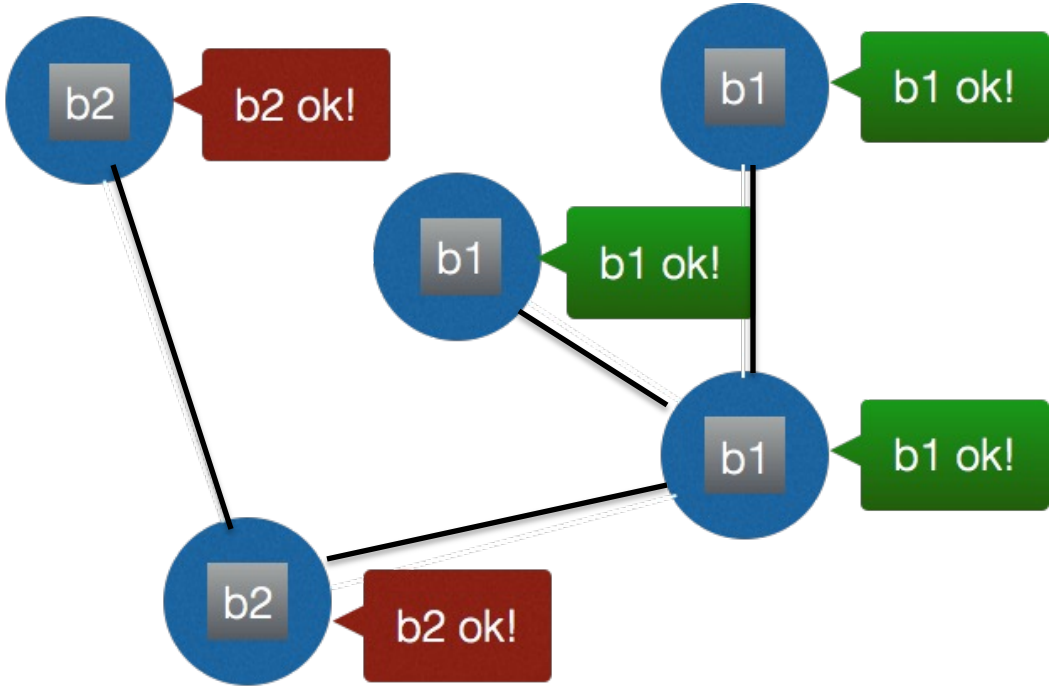
current blockchain

Consensus



current blockchain

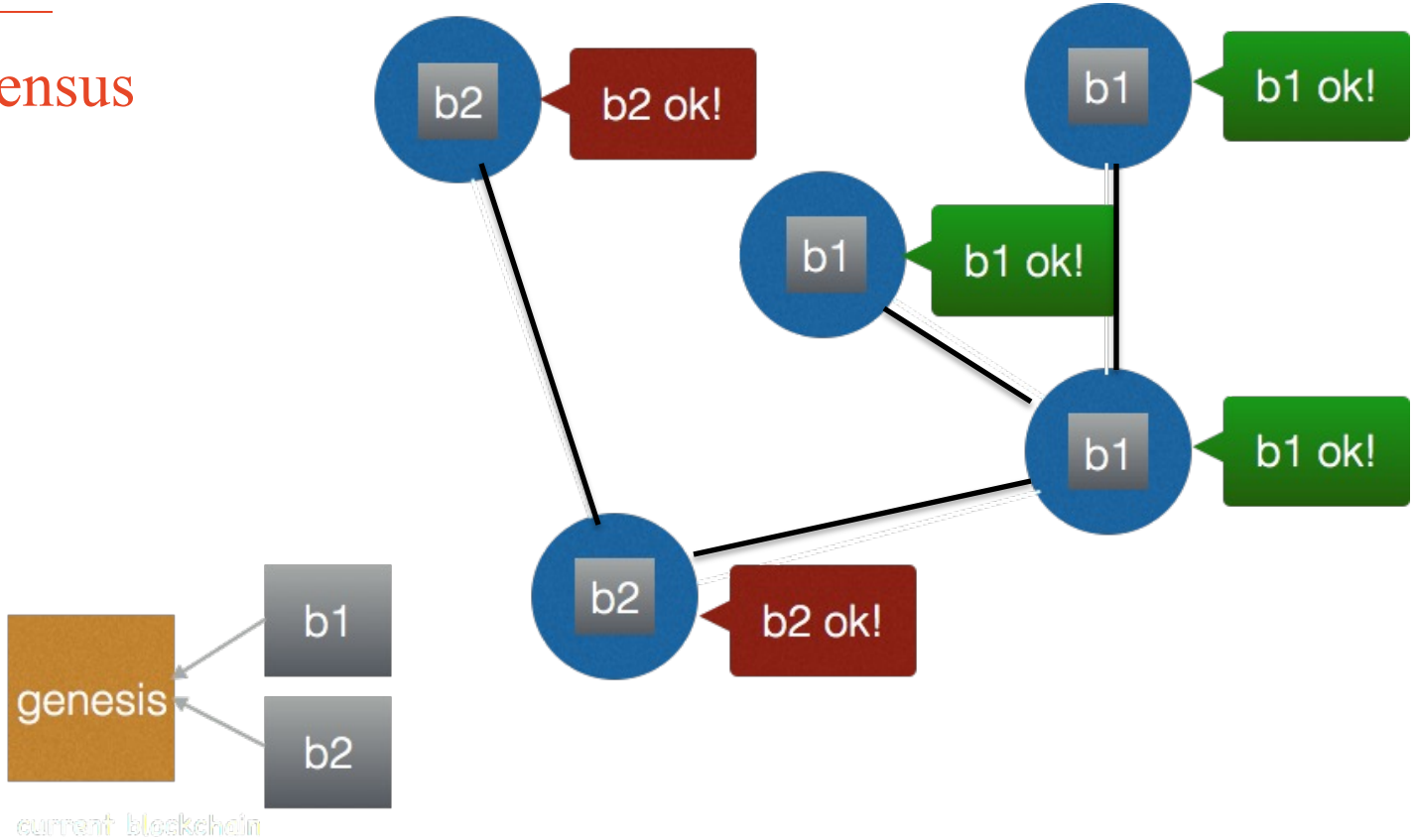
Consensus



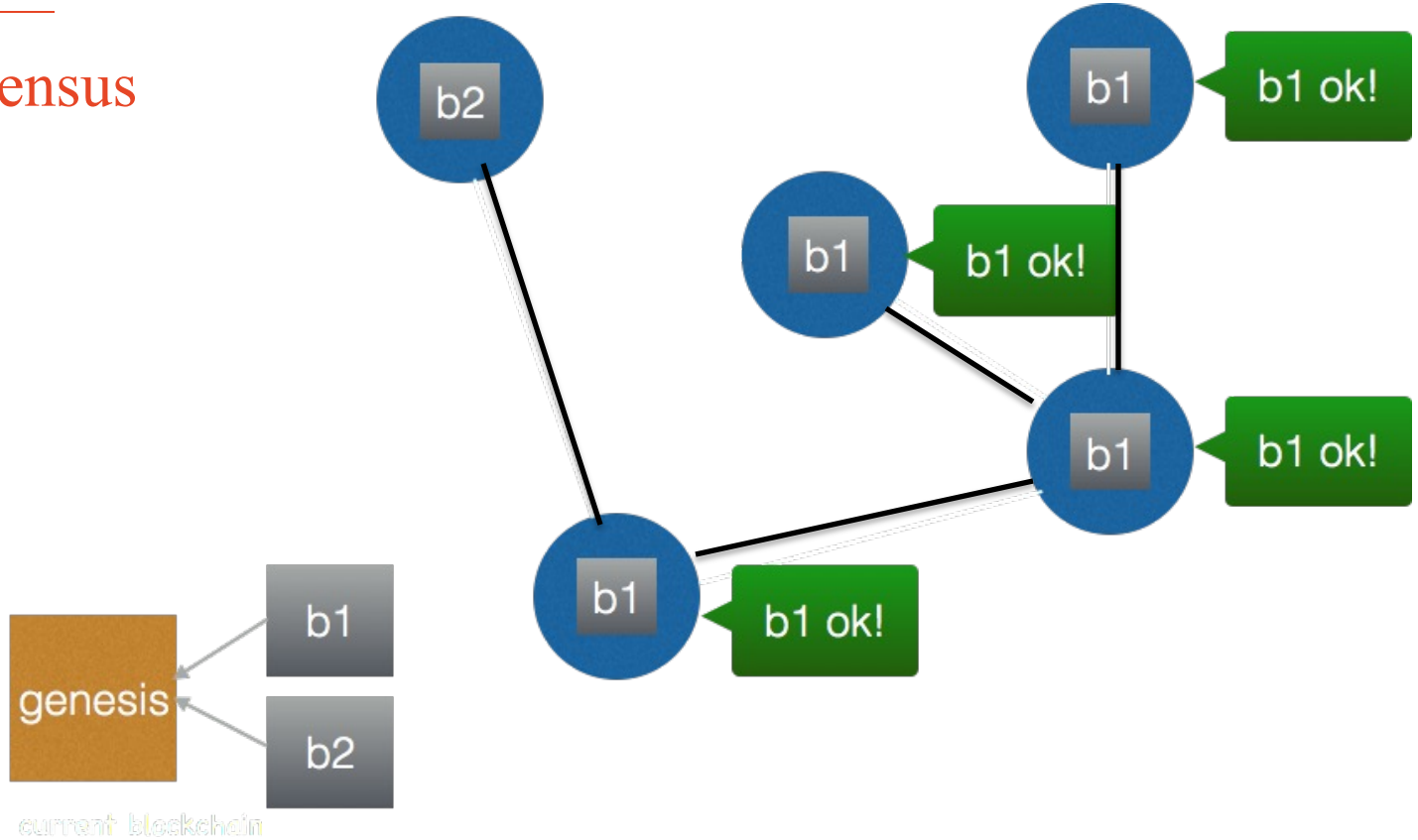
genesis

current blockchain

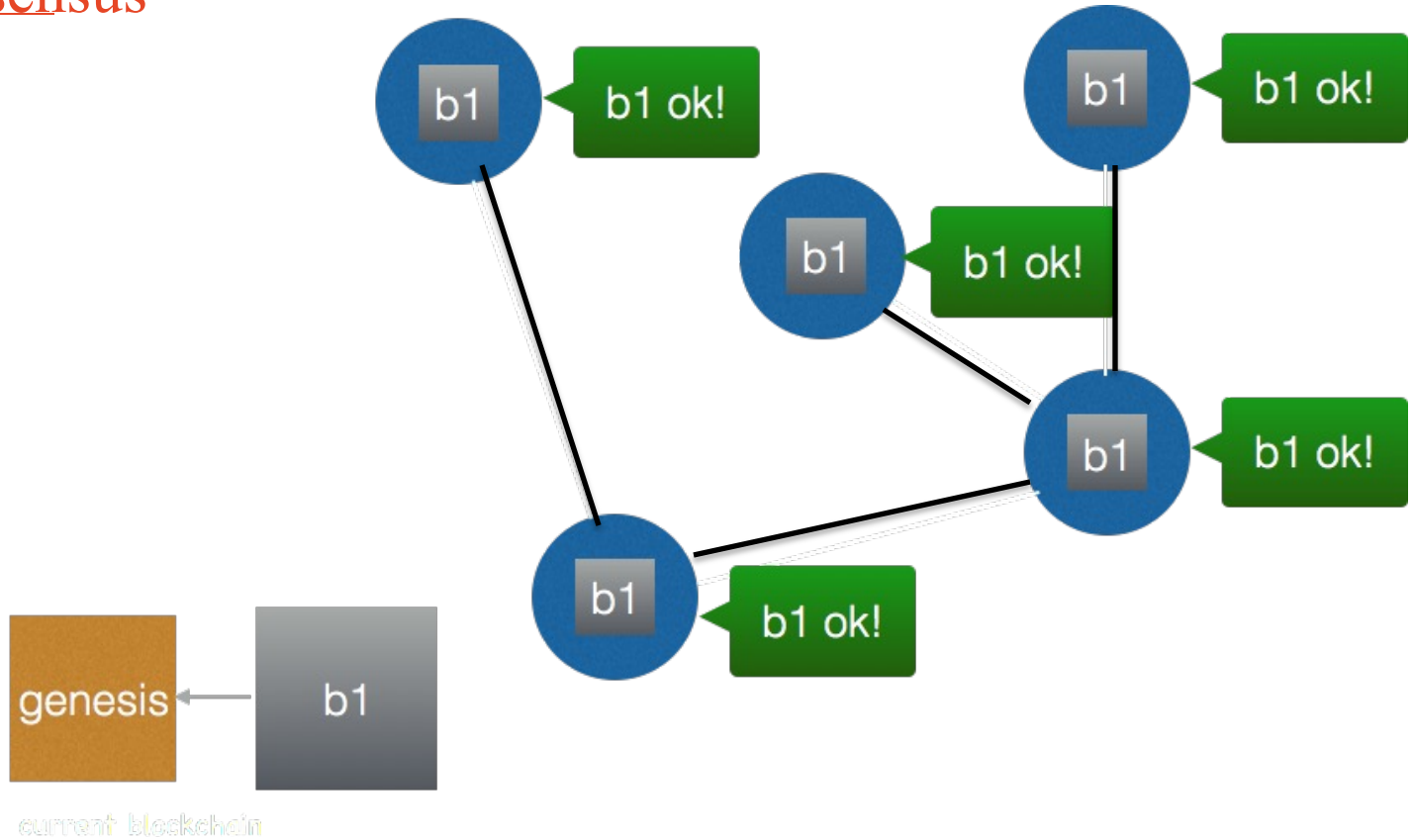
Consensus


b2
b2 ok!
b1
b1 ok!
b1
b1 ok!
b1
b1 ok!
b2
b2 ok!
genesis
b1
b2
current blockchain

Consensus


b2
b1
b1 ok!
b1
b1 ok!
b1
b1 ok!
b1
b1 ok!
genesis
b1
b2
current blockchain

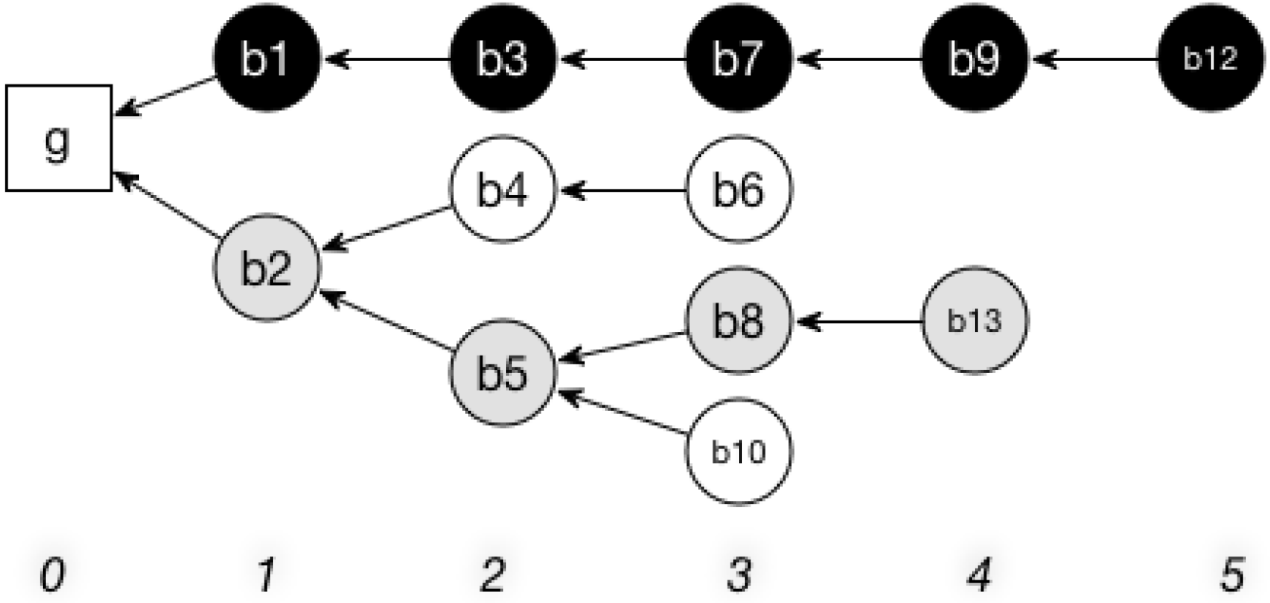
Consensus


b1
b1 ok!
b1
b1 ok!
b1
b1 ok!
b1
b1 ok!
b1
b1 ok!
genesis
b1
current blockchain

Recovering from disagreements

Resolving a fork

Assume that the current blockchain state is this DAG



Bitcoin's consensus chooses the deepest branch

State
⟨Bi,Pi⟩ the local blockchain view

Each peer of the blockchain executes:

Receive blocks ⟨Bj, Pj⟩ from j
  Bi = Bi ∪Bj
  Pi = Pi ∪Pj

depth(b):
  if children(b) = ∅ then return 1
  else return 1 + max $_{c \in children(b)}$ depth(c)

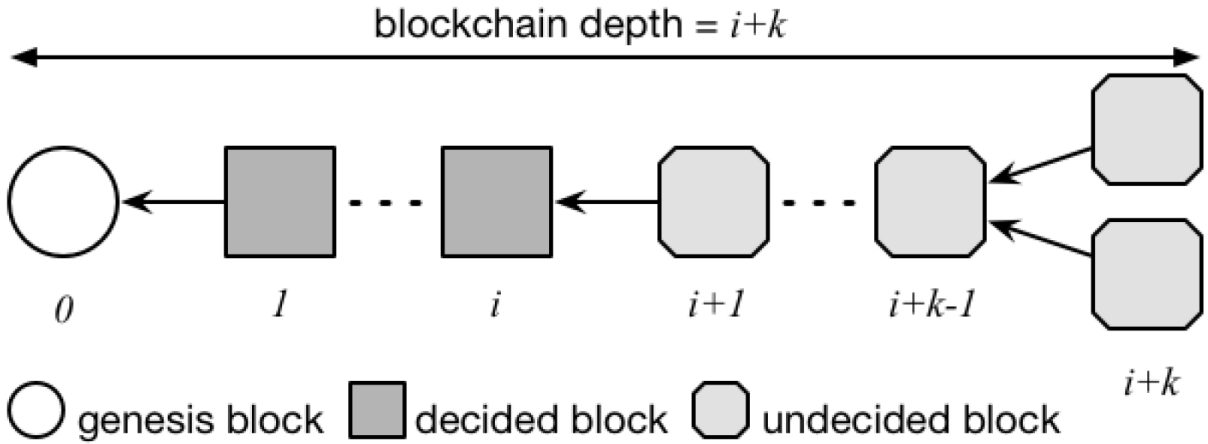
Prune shortest branches at i
  b = genesis-block(Bi)
  while b.next ≠⊥
    block = argmax $_{c \in children(b)}$ { depth(c)
}
    B = B ∪ {block}
    P = P ∪ {⟨block,b⟩}
    b = block
  ⟨Bi,Pi⟩ = ⟨B,P⟩

[Nak08] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008, http://www.bitcoin.org.

When is a transaction committed?

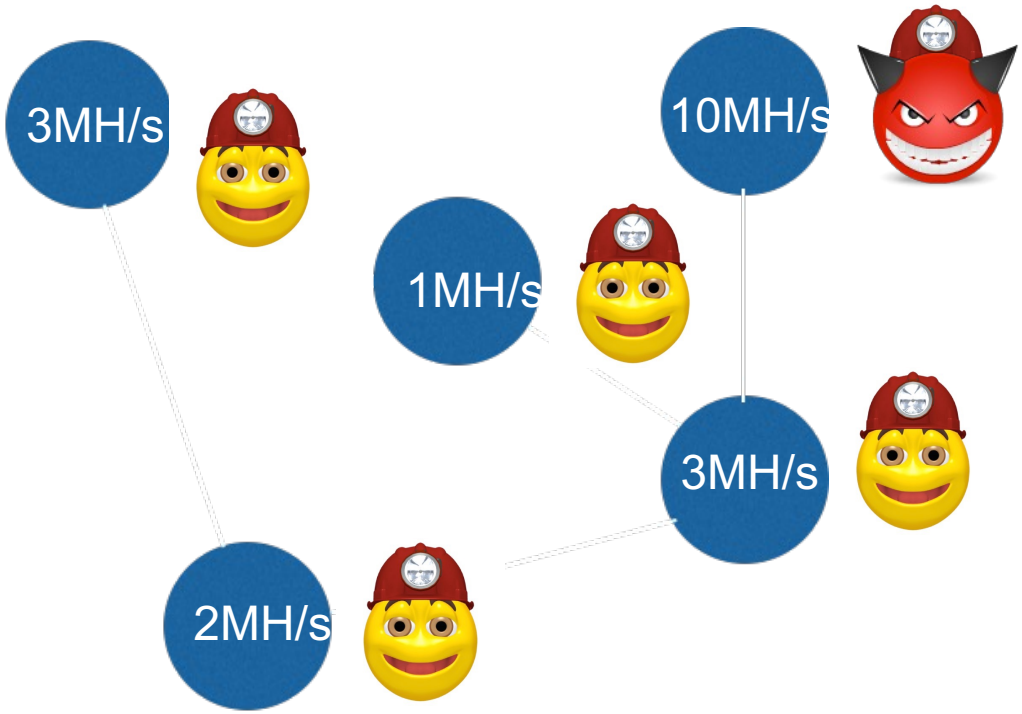
Committed transaction

- Given a blockchain with parameter k, a block at index i is *decided* when the chain depth reaches i+k
- A transaction is *committed* if it belongs to a decided block



The 51% attack

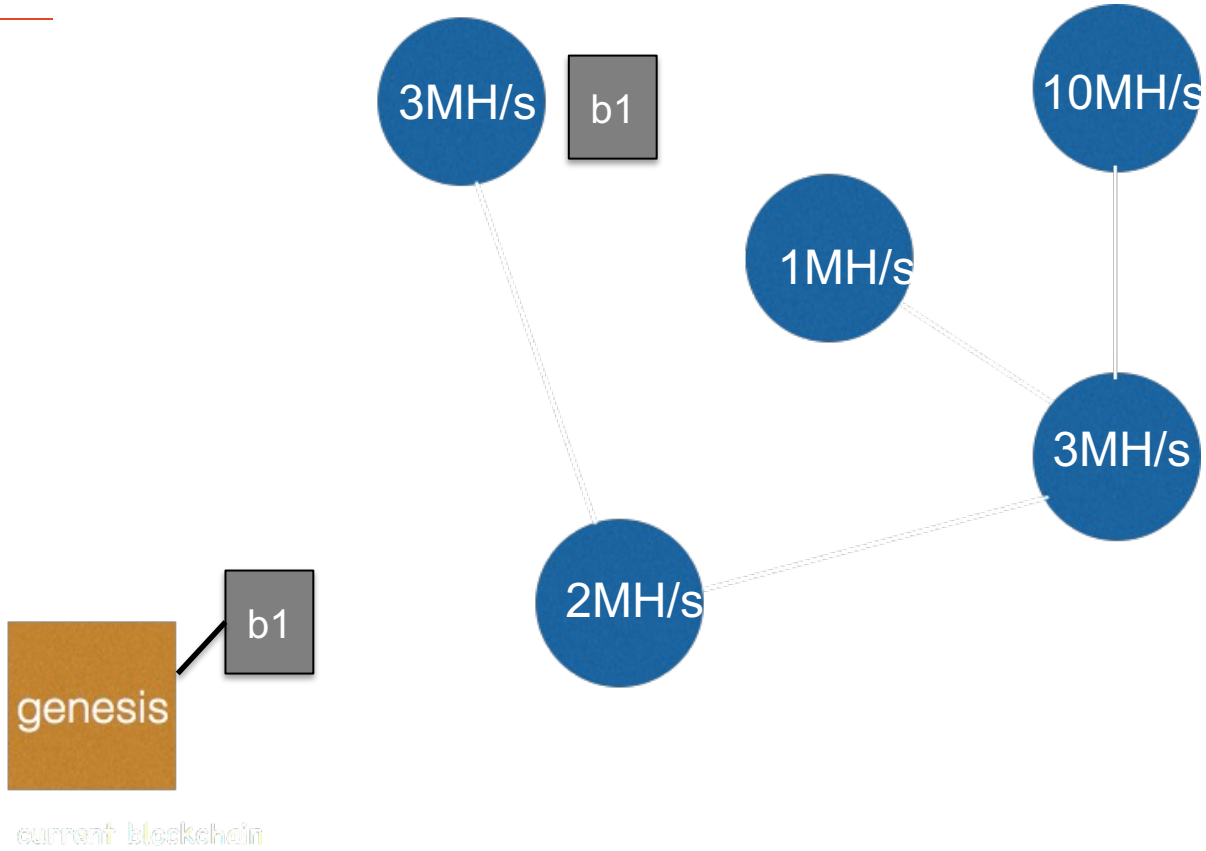
Impact of mining power



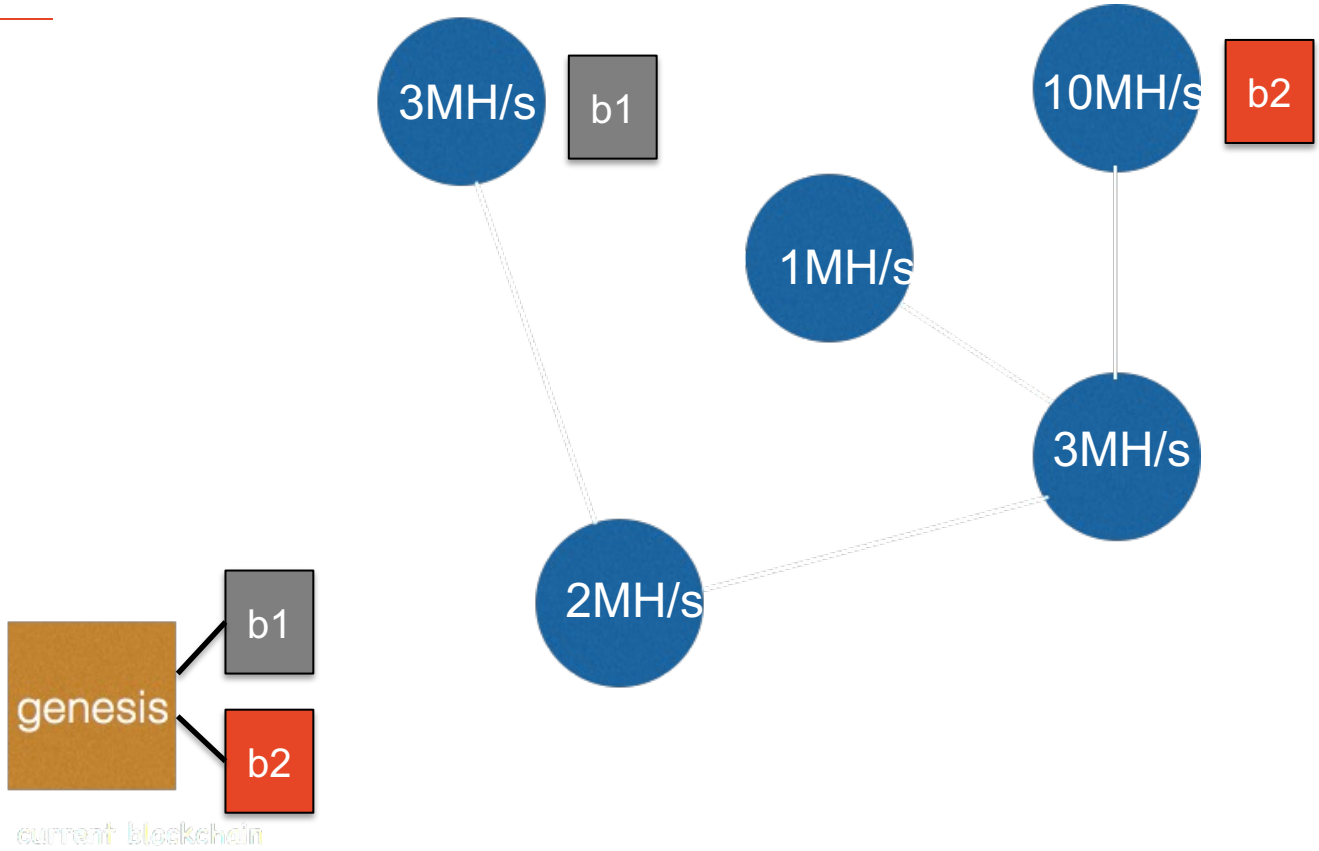
genesis

current blockchain

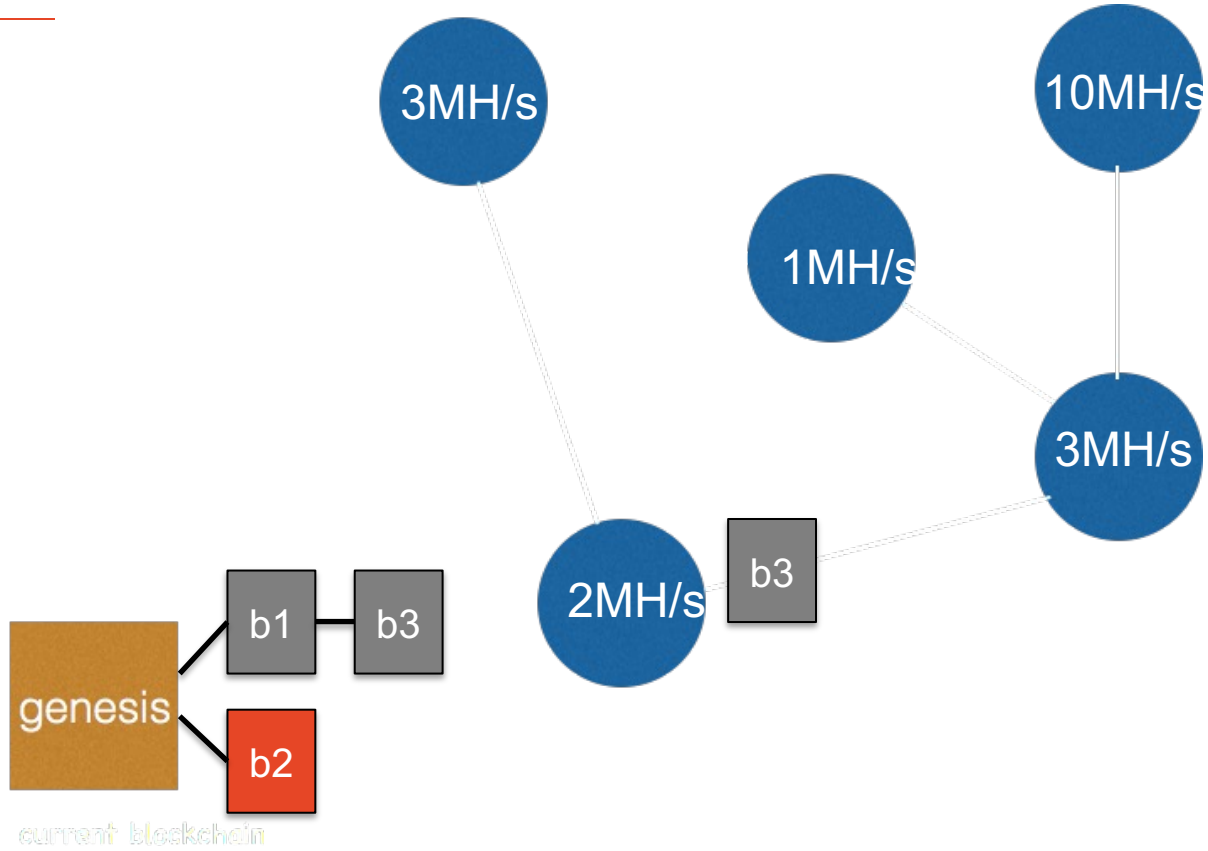
Impact of mining power



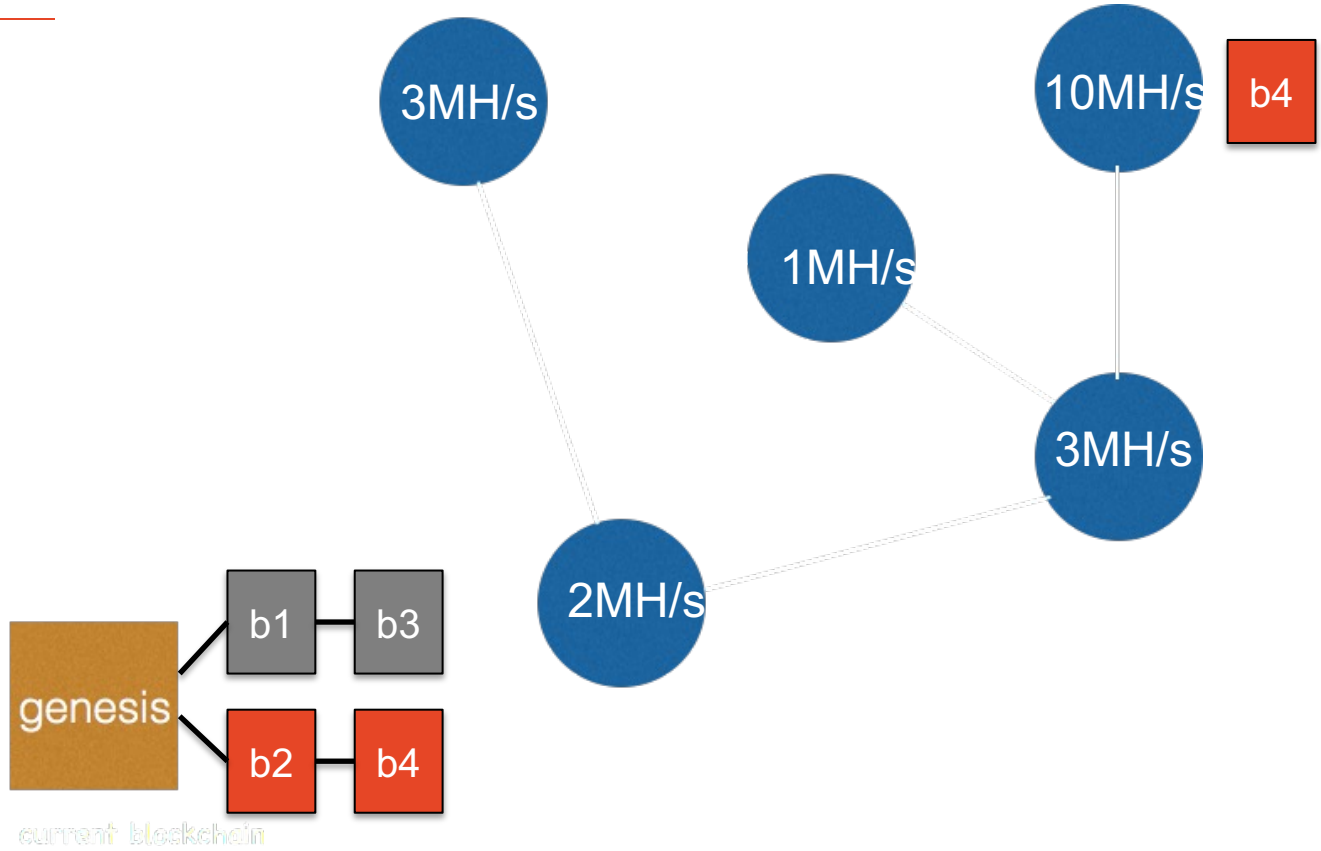
Impact of mining power



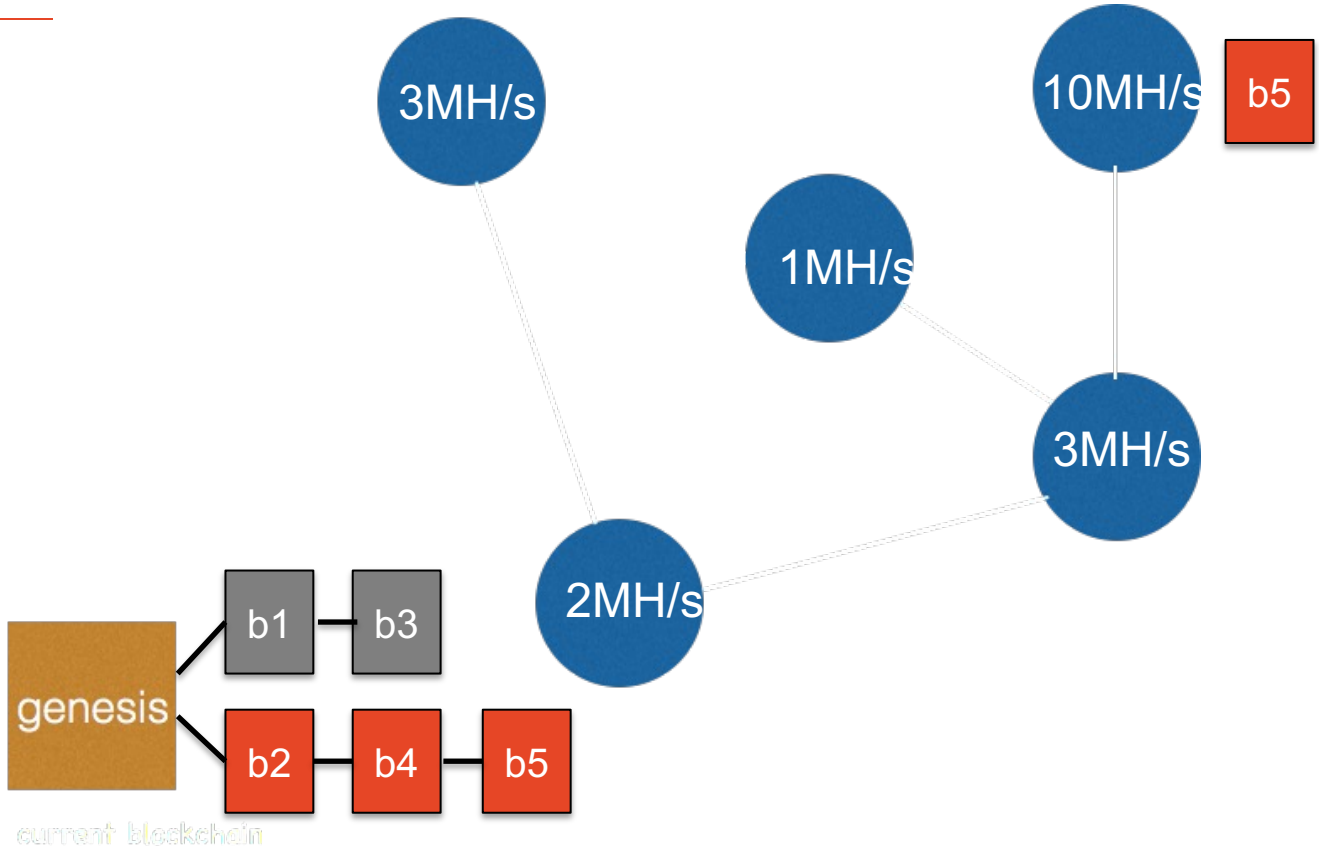
Impact of mining power



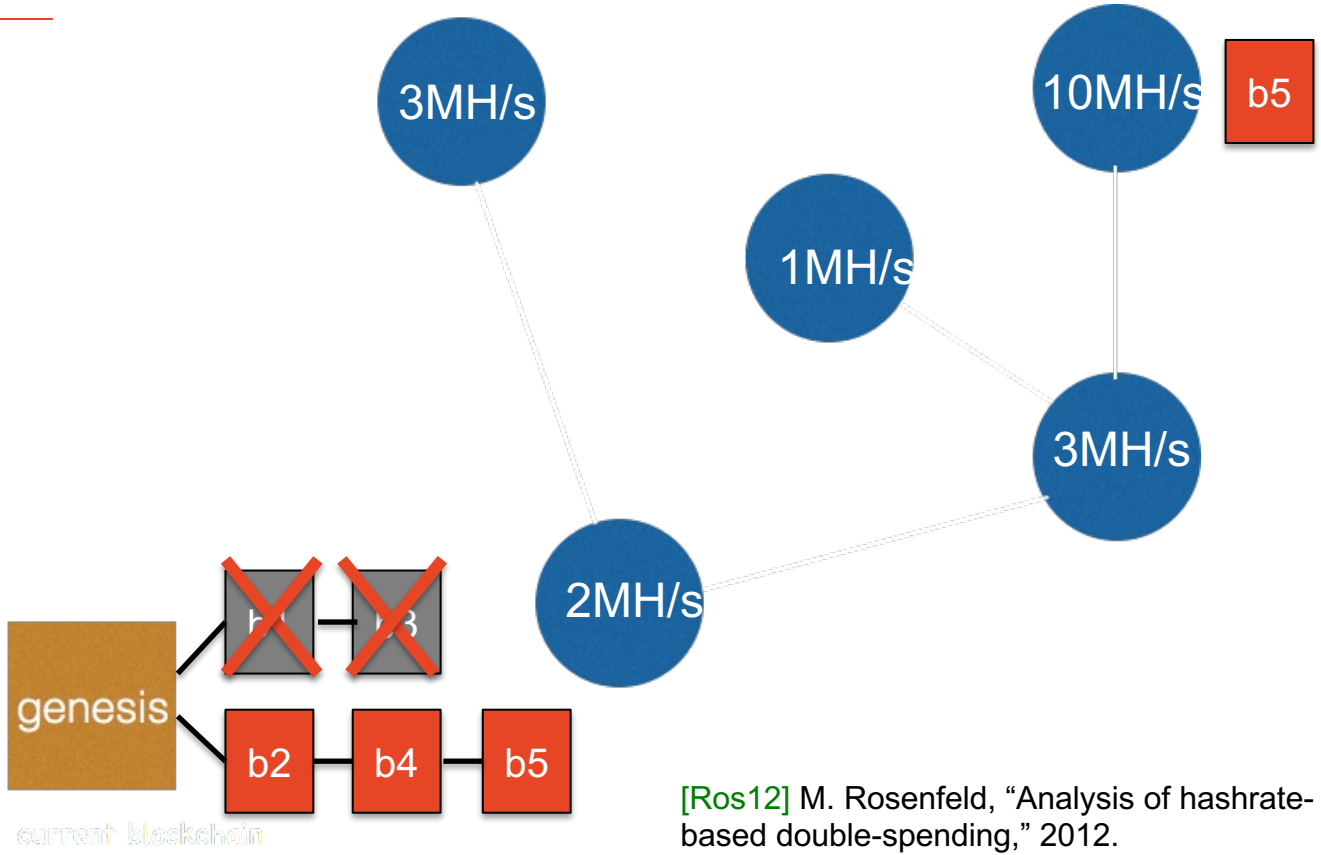
Impact of mining power



Impact of mining power



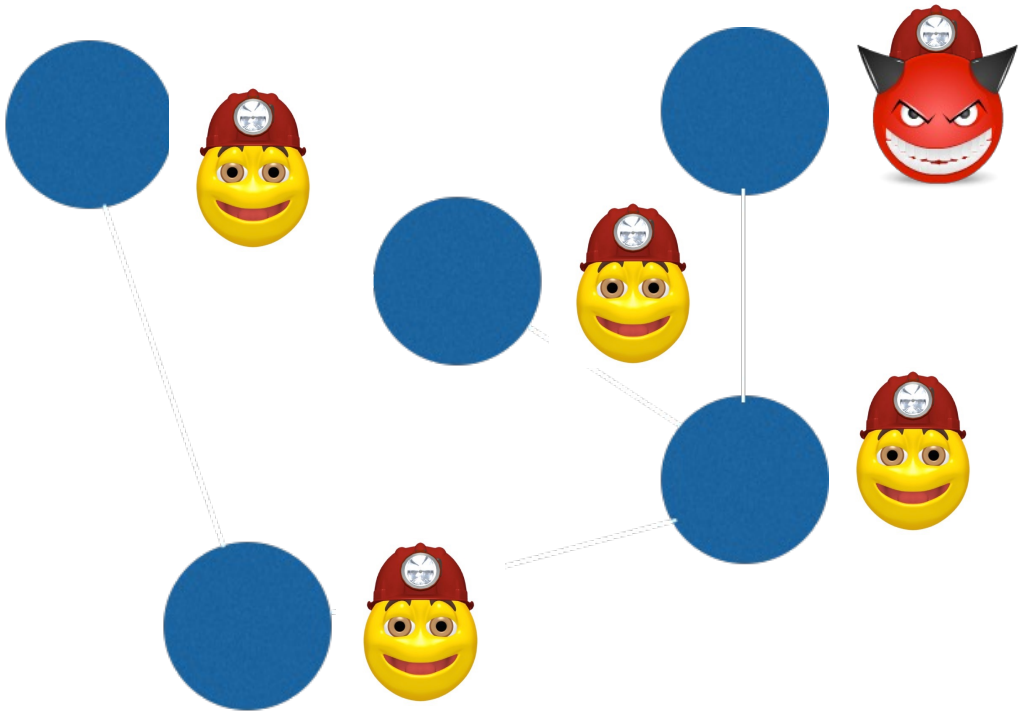
Impact of mining power



[Ros12] M. Rosenfeld, “Analysis of hashrate-based double-spending,” 2012.

The impact of network delays

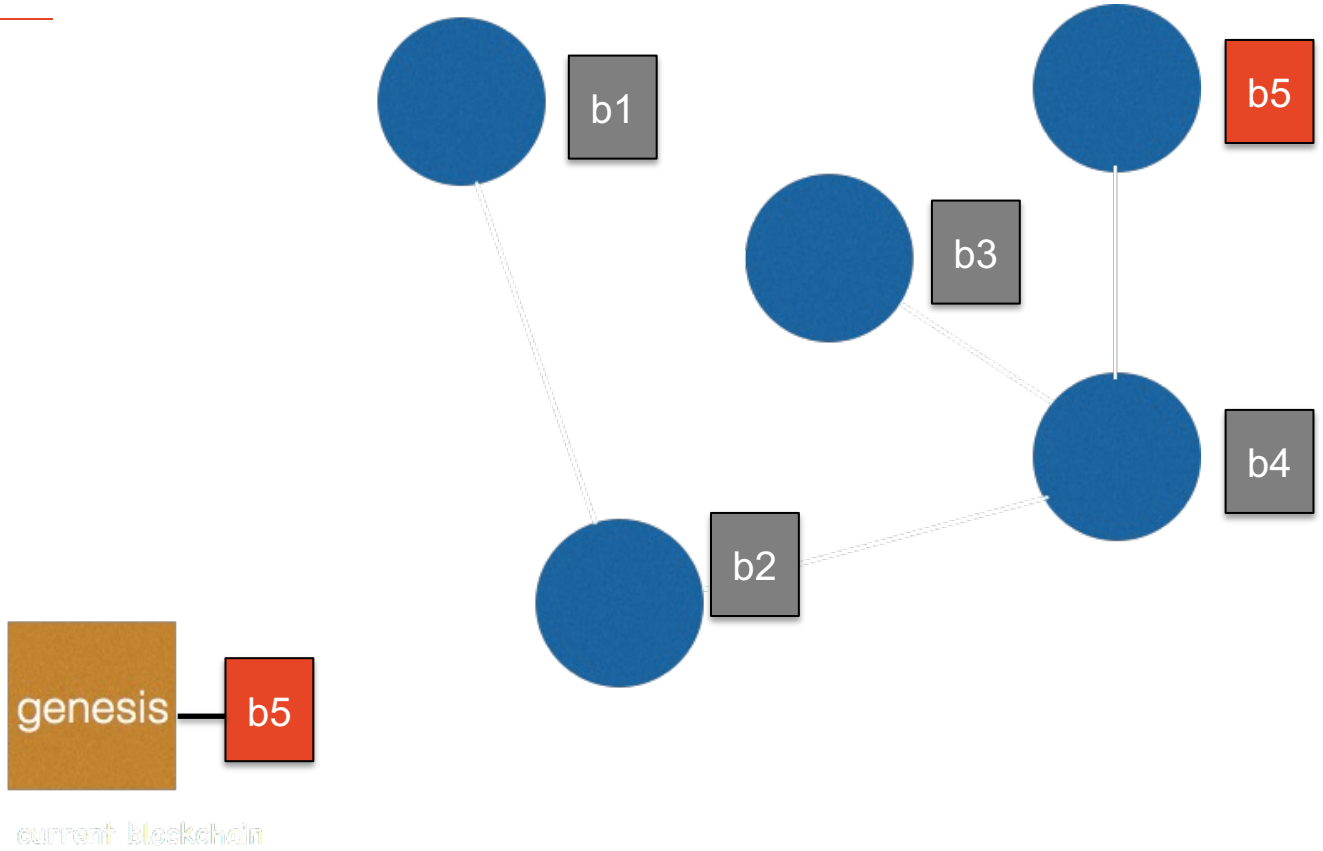
Impact of communication delay



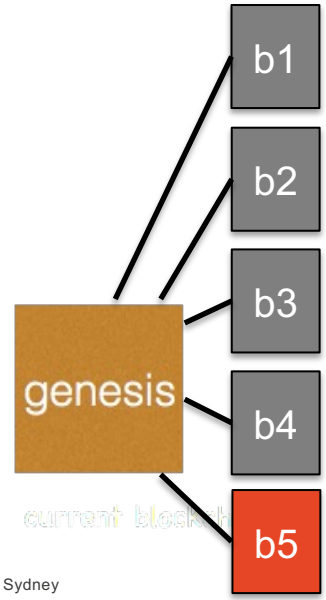
genesis

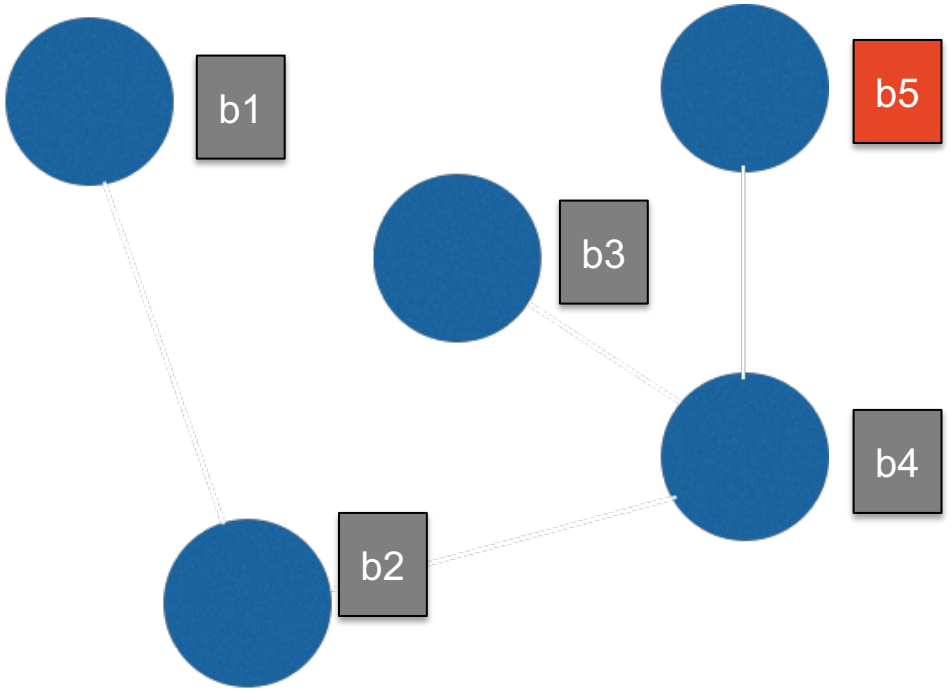
current blockchain

Impact of communication delay



Impact of communication delay





Ethereum and GHOST

Ethereum consensus was inspired by the choice of the greedily heaviest observable subtree (GHOST)

State
⟨Bi,Pi⟩ the local blockchain view

Each peer of the blockchain executes:

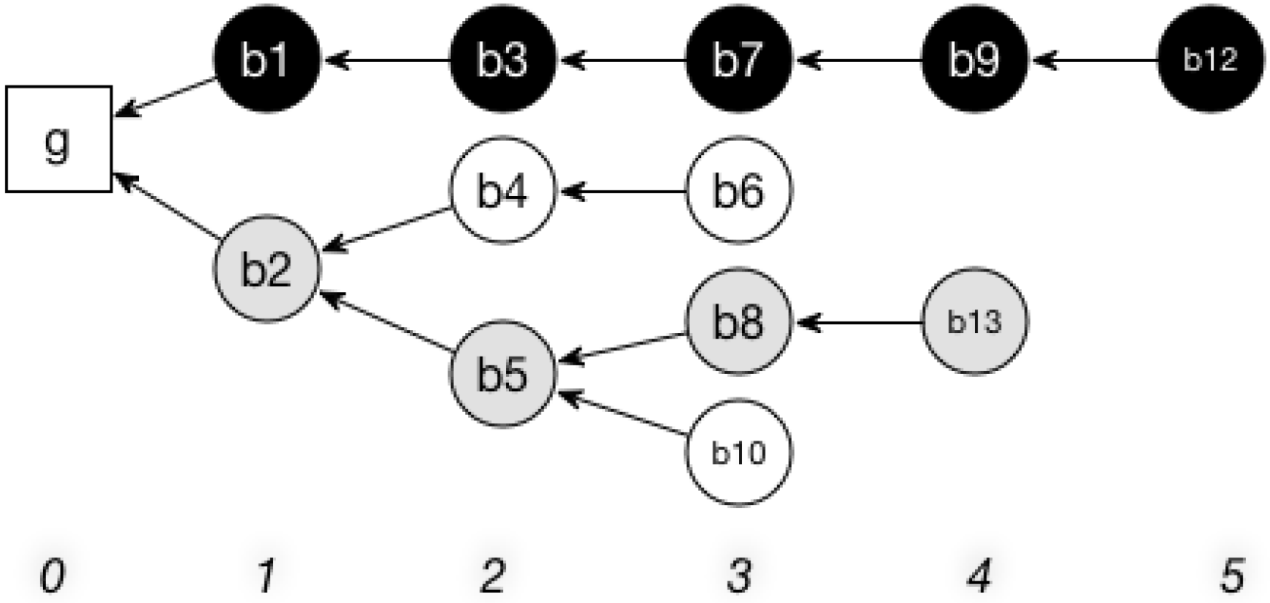
Receive blocks ⟨Bj, Pj⟩ from j
  Bi = Bi ∪Bj
  Pi = Pi ∪Pj

num-desc(b):
  if children(b) = ∅ then return 1
  else return 1 + $\sum_{c \in children(b)}$ num-desc(c)

Prune lightest branches at i
  b = genesis-block(Bi)
  while b.next ≠⊥
    block = argmax $_{c \in children(b)}$ {num-desc (c)}
    B = B ∪ {block}
    P = P ∪ {⟨block,b⟩}
    b = block
  ⟨Bi,Pi⟩ = ⟨B,P⟩

[SZ15] Y. Sompolinsky and A. Zohar, “Secure high-rate transaction processing in bitcoin,” in Financial Cryptography and Data Security FC’15, 2015, pp. 507–527.

Bitcoin vs. Ethereum/GHOST



Ethereum branch selection

Ethereum codebase evolved dramatically over the past two years

While originally stated as inspired by GHOST, Ethereum's branch selection algorithm is currently different:

Ethereum selects the branch with the highest total difficulty, which is the sum of the difficulties of all the cryptopuzzles of its last block and its ancestors.

Quiz

Question 1

What is the PoW mechanism useful for?

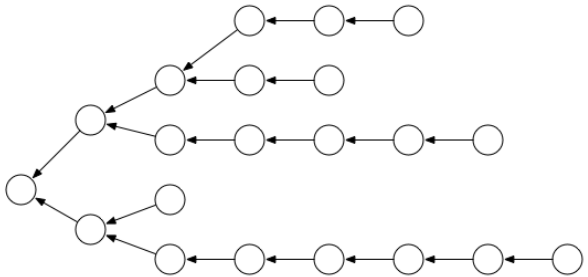
Question 2

What is the PoW mechanism useful for?

What is its major drawback?

Question 3

What are the blocks of the branch selected by Bitcoin?

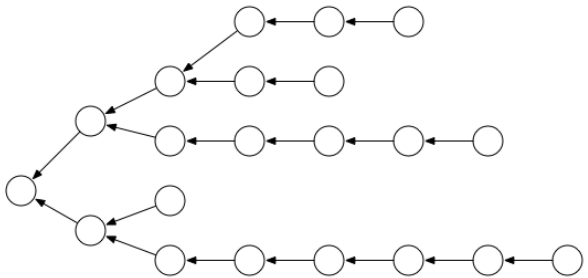


Consensus

Consensus without failures

Question 4

What are the blocks of the branch selected by Bitcoin?



What are the blocks of the branch selected by the GHOST protocol?

Consensus without failures

Each process:

1. Broadcasts its own value

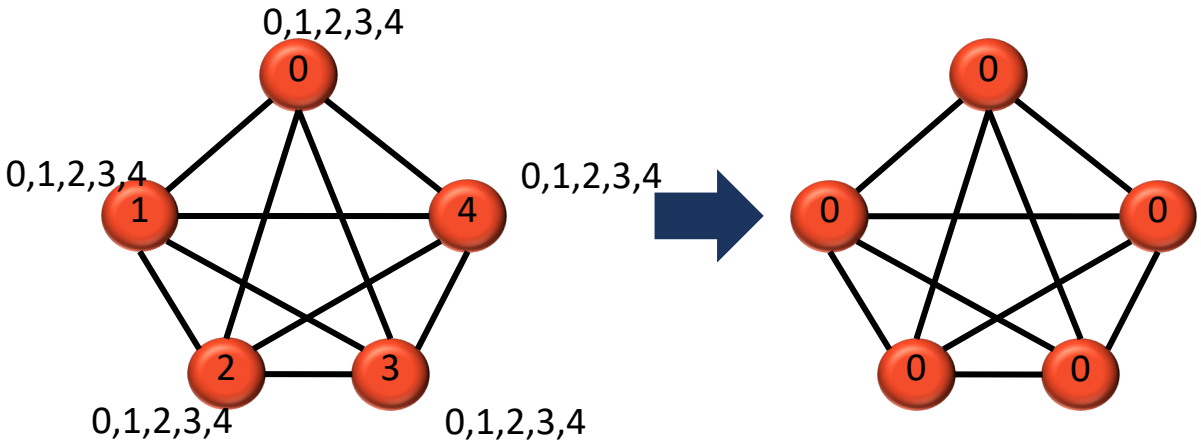
2. Decides on the minimum of all received values

State of process p_i:
V, the set of known values, initially {x} the proposed value of p_i

Basic consensus algorithm for p_i:
send {v in V that p_i has not sent yet} to all p_j *// broadcast values just found*
receive S_j from all p_j, j≠i *// receive values*
V = V∪S_j for all j *// records the values*
y := min(V) *// at the end, output min*

Consensus without failures

- Broadcast values and decide on minimum → Consensus!
- Validity is satisfied because the decided value is a proposed one
- Agreement is satisfied because all decide the same



Quiz

Complexity of Consensus w/o failure algorithm?

- Number of messages needed to complete?
- Number of bits exchanged needed to complete?
- Time needed to complete?

State of process p_i:
V, the set of known values, initially {x} the proposed value of p_i

Basic consensus algorithm for p_i:
send {v in V that p_i has not sent yet} to all p_j *// broadcast values just found*
receive S_j from all p_j $j \neq i$ *// receive values*
$V = V \cup S_j$ for all j *// records the values*
y := min(V) *// at the end, output min*

Complexity of Consensus w/o failure algorithm?

Message complexity. Note that only one communication round is needed!
Each process:

1. Broadcast own value → $O(n^2)$
2. Decide on the minimum of all received values

The message complexity is thus $O(n^2)$.

Communication complexity.
Each message is an integer, leading to a communication complexity of $O(bn^2)$, where b is the number of bits necessary to encode any value

Time complexity.
As all messages can be exchanged in parallel, it only takes O(1) message delay to terminate.

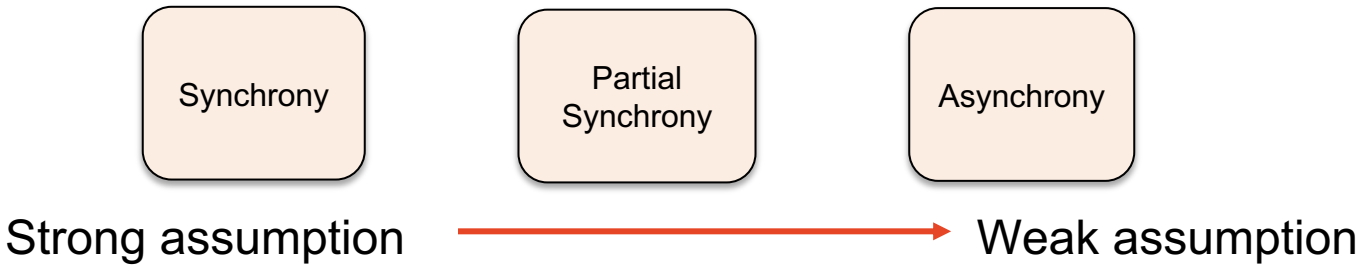
Synchrony and asynchrony

Synchrony vs. Asynchrony

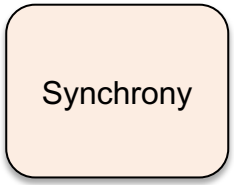
So far, we assumed *synchrony:* that all messages are received within a bounded time that is known from the algorithms

What will happen to the previous algorithms if we relax this assumption?

1. *Partial synchrony*: the bound on message delay is unknown
2. *Asynchrony*: message delays are unbounded



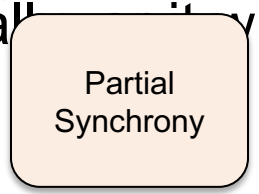
Synchrony vs. Asynchrony

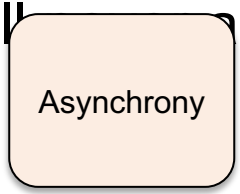


Synchrony helps a lot actually:

- If you know how much time a message takes to be transmitted and how much time a process needs to treat a message and sends a response, then one can detect:
  - Crashed processes or
  - Message losses

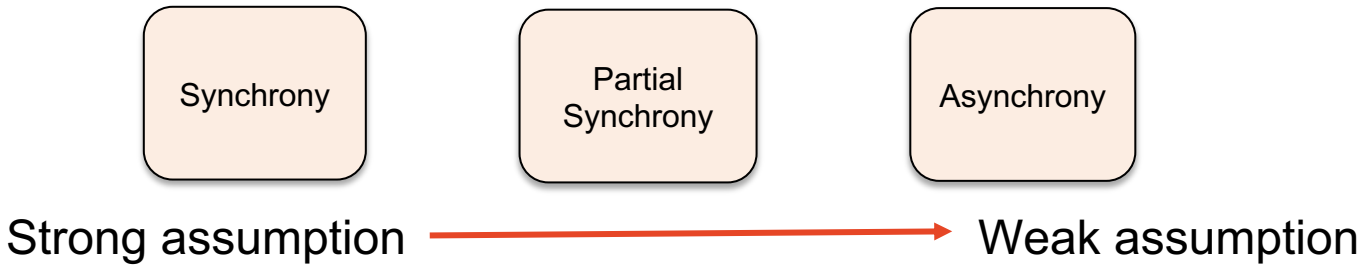
- Otherwise, you cannot distinguish whether a message is in-transit and will arrive eventual[illegible]rrive.





Partial synchrony is a stronger assumption than asynchrony

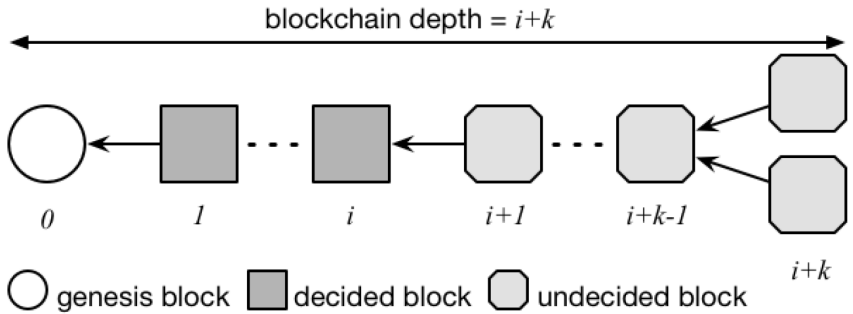
If a protocol does not work under partial synchrony then it cannot work under asynchrony as partial synchrony is a stronger assumption



Quiz

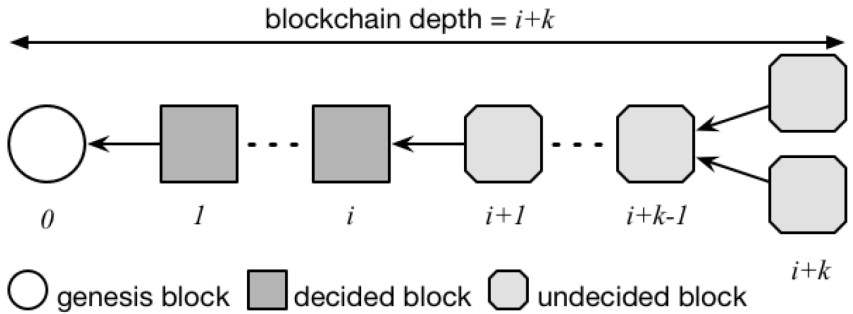
Question 1

What assumption is needed here for Ethereum to work?



Question 1

What assumption is needed here for Ethereum to work?



Synchrony

- Without synchrony, no one can distinguish a committed transaction from a transaction that is not yet committed and that can possibly be rolled back later on

Robustness

Consensus with crash failures

Crash Tolerant Consensus

- Assumptions
  - •Synchronous communication using unicast
  - •Crash failures
  - •A maximum of f processes can fail

State of process p_i:
V, the set of known values, initially {x} the proposed value of p_i
f, the maximum number of processors that can fail

Basic consensus algorithm for p_i:

```
for (k=1; k ≤ f+1; k++) {                          // in each f+1 round
  send {v in V that p_i has not sent yet} to all p_j.  // broadcast values just found
  receive S_j from p_j j≠i                          // receive values
  V = V∪S_j for all j                               // records the values
}
y := min(V)                                         // at the end, output min
```

Crash Tolerant Consensus

- Intuition of the proof: at least one iteration (i.e., *round*) without failure
  - •Termination: the loop has a bounded number of iterations, so y is assigned
  - •Agreement: at the end we have $V_i = V_j$ for any i and j so $y_i = y_j$
  - •Validity: there exists a p_j such that its input x_j is the decided value

State of process p_i:
V, the set of known values, initially {x} the proposed value of p_i
f, the maximum number of processors that can fail

Basic consensus algorithm for p_i:

```
for (k=1; k ≤ f+1; k++) {                    // in each f+1 round
  send {v in V that p_i has not sent yet} to all p_j    // broadcast values just found
  receive S_j from p_j j≠i                              // receive values
  V = V∪S_j for all j                                   // records the values
}
y := min(V)                                             // at the end, output
```

Complexity of Crash Tolerant Consensus?

Complexity of Crash Tolerant Consensus?

Message complexity. There are f+1 rounds where f<n.
The number of messages is at most $O(n^2)$ in each round.
The total number of messages is thus $O((f+1)\ n^2)$.

Communication complexity. Each message contains up to n values, and each value is represented with b bits, the communication complexity is $O(b\ (f+1)\ n^3)$ bits.

Time complexity. There are f+1 rounds, in each of these rounds, messages can be exchange in parallel. This leads to a time complexity of O(f+1) message delay.

Theorem

At least f+1 rounds are necessary to solve consensus where f<n.

Proof.
In the worst case scenario, one process fails in each round sending to only one process but not the others.
With f+1 rounds, there is at least one round without failure, in which all messages are successfully delivered, so that all correct processes can decide upon the same value.

Quiz

Question

What are the message, communication and time complexities of the Crash Tolerant Consensus algorithm?

State of process p_i:
V, the set of known values, initially {x} the proposed value of p_i
f, the maximum number of processors that can fail

Basic consensus algorithm for p_i:
for (k=1; k ≤ f+1; k++) { *// in each f+1 round*
  send {v in V that p_i has not sent yet} to all p_j *// broadcast values just found*
  receive S_j from p_j j≠i *// receive values*
  V = V∪S_j for all j *// records the values*
}
y := min(V) *// at the end, output*

Complexity of Crash Tolerant Consensus?

Message complexity. There are f+1 rounds where f<n.
The number of messages is at most $O(n^2)$ in each round.
The total number of messages is thus $O((f+1)\ n^2)$.

Communication complexity. Each message contains up to n values, and each value is represented with b bits, the communication complexity is $O(b\ (f+1)\ n^3)$ bits.

Time complexity. There are f+1 rounds, in each of these rounds, messages can be exchange in parallel. This leads to a time complexity of O(f+1) message delay.

Consensus with Byzantine failures

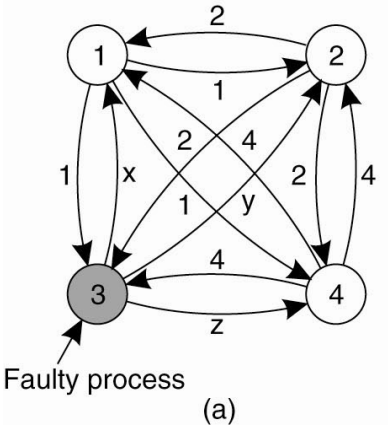
The Byzantine Consensus Problem

In the consensus problem, the processes propose values and have to agree by deciding one among these values

Consensus Definition 2:

1. *Agreement*: No two correct processes decide differently
2. *Termination*: Every correct process eventually decides
3. *Validity*: if all correct processes propose the same value, then this value is decided moreover, if all processes are correct and some process decides v, then v was proposed by some process

Byzantine Tolerant Consensus



- Assumptions
  - Synchronous communication using unicast
  - Byzantine failures
  - A maximum of f < n/3 processes can fail among n

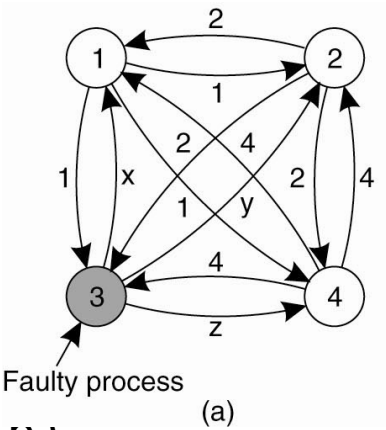
1	Got(1, 2, x, 4)
2	Got(1, 2, y, 4)
3	Got(1, 2, 3, 4)
4	Got(1, 2, z, 4)

(b)

1 Got	2 Got	4 Got
(1, 2, y, 4)	(1, 2, x, 4)	(1, 2, x, 4)
(a, b, c, d)	(e, f, g, h)	(1, 2, y, 4)
(1, 2, z, 4)	(1, 2, z, 4)	(i, j, k, l)

(c)

Byzantine Tolerant Consensus



- Example of solution for n=4 and f=1
  - •(a) every process sends its input to all others
  - •(b) the collected vectors (i^{th} coordinate received from p_i)
  - •(c) every process sends the collected vector and obtains these 3 new vectors
  - •A majority vector is computed using for each coordinate the one appearing a majority of times (or ⊥ if there is none)
  - •(1, 2, ⊥, 4)

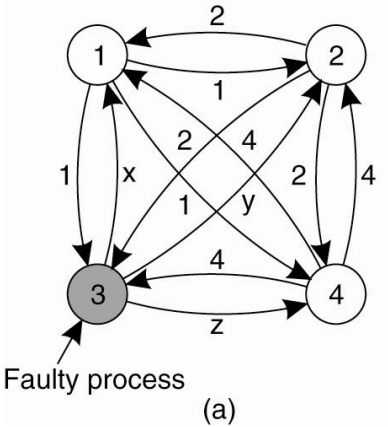
1	Got(1, 2, x, 4)
2	Got(1, 2, y, 4)
3	Got(1, 2, 3, 4)
4	Got(1, 2, z, 4)

(b)

1 Got	2 Got	4 Got
(1, 2, y, 4)	(1, 2, x, 4)	(1, 2, x, 4)
(a, b, c, d)	(e, f, g, h)	(1, 2, y, 4)
(1, 2, z, 4)	(1, 2, z, 4)	(i, j, k, l)

(c)

Byzantine Tolerant Consensus



- Assumptions
  - Synchronous communication using unicast
  - Byzantine failures
  - A maximum of f < n/3 processes can fail among n

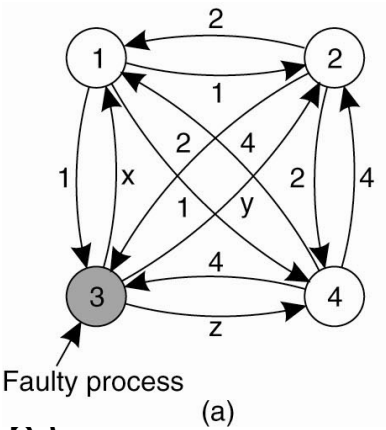
1	Got(1, 2, x, 4)
2	Got(1, 2, y, 4)
3	Got(1, 2, 3, 4)
4	Got(1, 2, z, 4)

(b)

1 Got	2 Got	4 Got
(1, 2, y, 4)	(1, 2, x, 4)	(1, 2, x, 4)
(a, b, c, d)	(e, f, g, h)	(1, 2, y, 4)
(1, 2, z, 4)	(1, 2, z, 4)	(i, j, k, l)

(c)

Byzantine Tolerant Consensus



- Example of solution for n=4 and f=1
  - •(a) every process sends its input to all others
  - •(b) the collected vectors (i^{th} coordinate received from p_i)
  - •(c) every process sends the collected vector and obtains these 3 new vectors
  - •A majority vector is computed using for each coordinate the one appearing a majority of times (or ⊥ if there is none)
  - •(1, 2, ⊥, 4)

1	Got(1, 2, x, 4)
2	Got(1, 2, y, 4)
3	Got(1, 2, 3, 4)
4	Got(1, 2, z, 4)

(b)

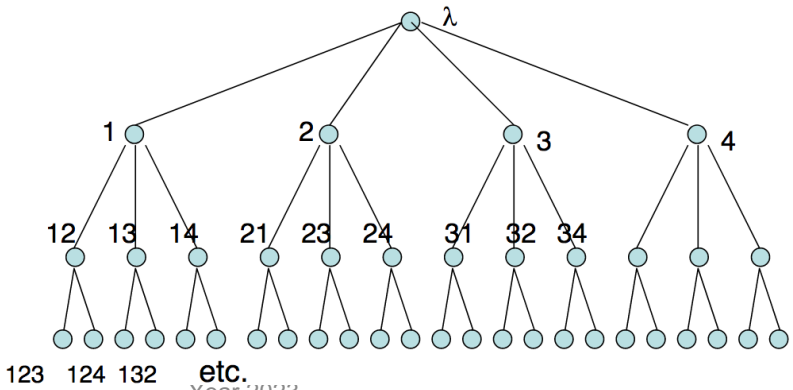
1 Got	2 Got	4 Got
(1, 2, y, 4)	(1, 2, x, 4)	(1, 2, x, 4)
(a, b, c, d)	(e, f, g, h)	(1, 2, y, 4)
(1, 2, z, 4)	(1, 2, z, 4)	(i, j, k, l)

(c)

Generalization: Exponential Information Gathering

EIG Tree

- $T_{n,f}$ for n processes and f failures:
- f+2 levels
- Each path from root to leaf corresponds to a string of f+1 distinct processes names
- Example $T_{4,1}$



Generalization: Exponential Information Gathering

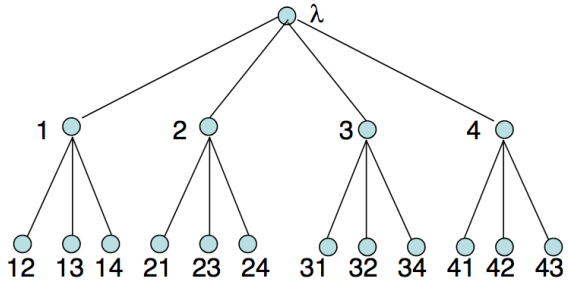
- Each process i uses the same EIG tree, $T_{n,f}$.
- Decorates nodes of the tree with values V, level by level.
- Initially: Decorate root with i's input value.
- Round r ≥1:
  - •Send all level r-1 decorations for nodes whose labels don't include i, to everyone.
    - •Including yourself---simulate locally.
  - •Use received messages to decorate level r nodes---to determine label, append sender's id at the end.
  - •If no message received, use ⊥.
- The decoration for node $(i_1,i_2,i_3,\ldots,i_k)$ in i's tree is the value v such that

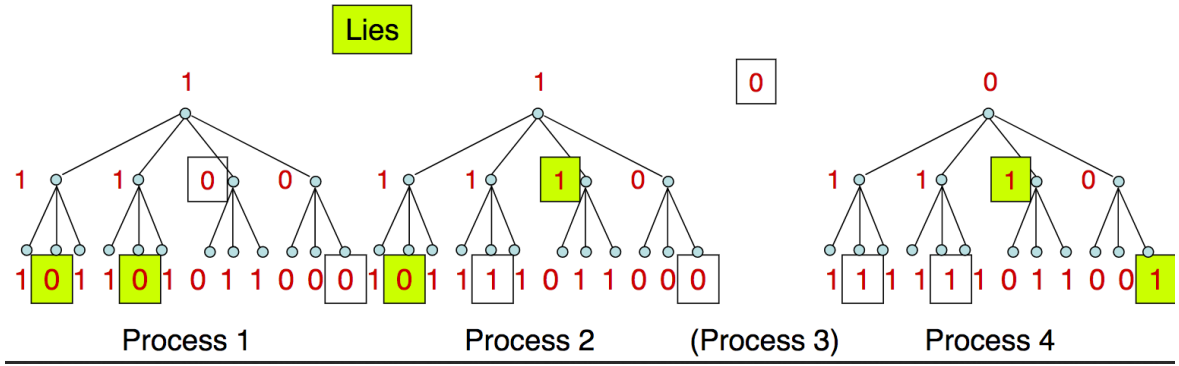
$(i_k$ told i) that $(i_{k-1}$told $i_k)$ that ...that $(i_1$ told $i_2)$ that i_1's initial value was v.

- Decision rule for stopping case:
  - •Trivial
  - •Let W = set of all values decorating the local EIG tree.
  - •If |W| = 1 decide that value, else default v0.

Example EIG n=4, f=1

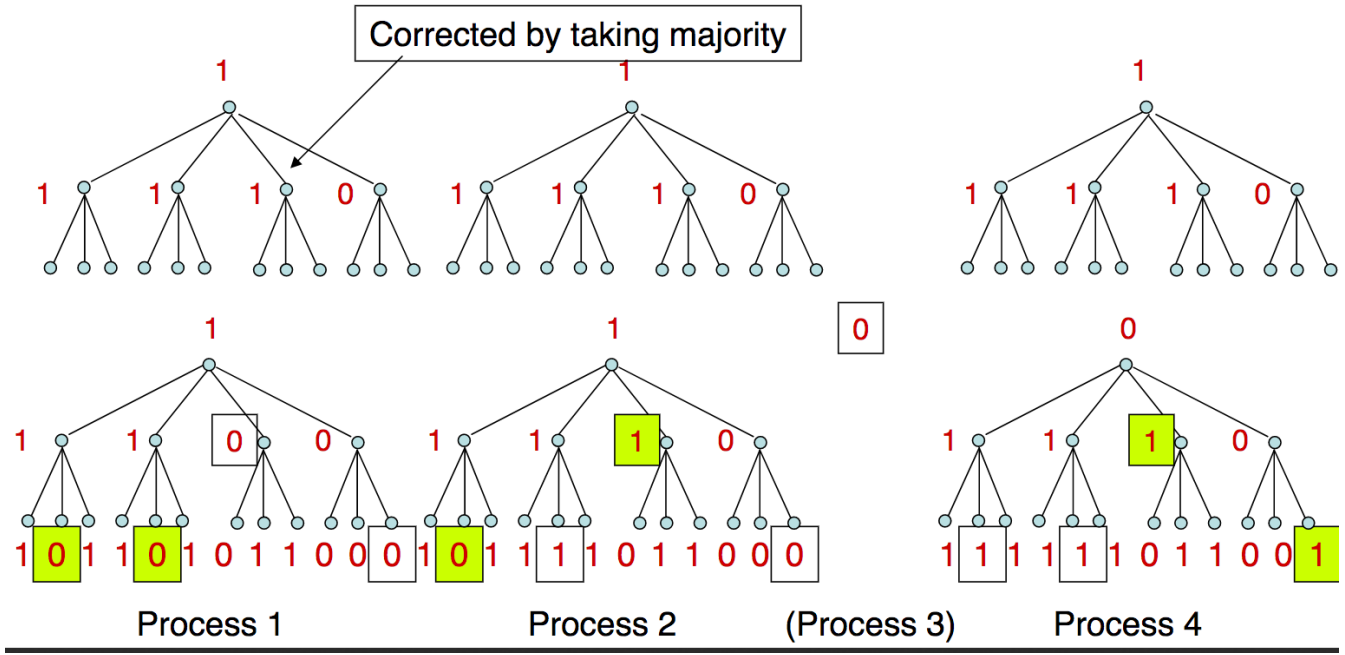
- $T_{4,1}$:
- Consider a possible execution in which p3 is faulty.
- Initial values 1 1 0 0
- Round 1
- Round 2





Example EIG n=4, f=1 (con't)

- Now calculate newvals, bottom-up, choosing majority values, $v_0 = 0$ if no majority.



Complexity of Byzantine Tolerant Consensus?

Complexity of Byzantine Tolerant Consensus (~~EIG~~)?

Message complexity. There are f+1 rounds where f<n.
The number of messages is at most $O(n^2)$ in each round.
The total number of messages is thus $O((f+1)\ n^2)$.

Communication complexity. Exponential in the number of failures f, $O(b\ n^{f+1})$ bits.

Time complexity. There are f+1 rounds, in each of these rounds, messages can be exchange in parallel. This leads to a time complexity of O(f+1) message delays.

Message losses

Message loss

Cause

- Networks are in general unreliable
- Messages can be lost (never been delivered even if sent)

- Examples:
  - A server receives too many requests simultaneously so it cannot treat all
  - A router drops the message because its queue is full

Message losses may impact the computation of a distributed system

Message loss

Coordinated Attack Problem

- Constraints of the problem
  - •Two armies, each led by a general on separate mountains surrounding a battlefield (distributed system)
  - •Can only communicate via messengers (message passing)
  - •Messengers can be killed before reaching destination (message losses)
- Goal: they want to coordinate an attack
  - •If they attack at different times, they both die
  - •If they attack at the same time, they win

Message loss

Coordinated Attack Problem (con't)

- There is no protocols to make sure they will win!

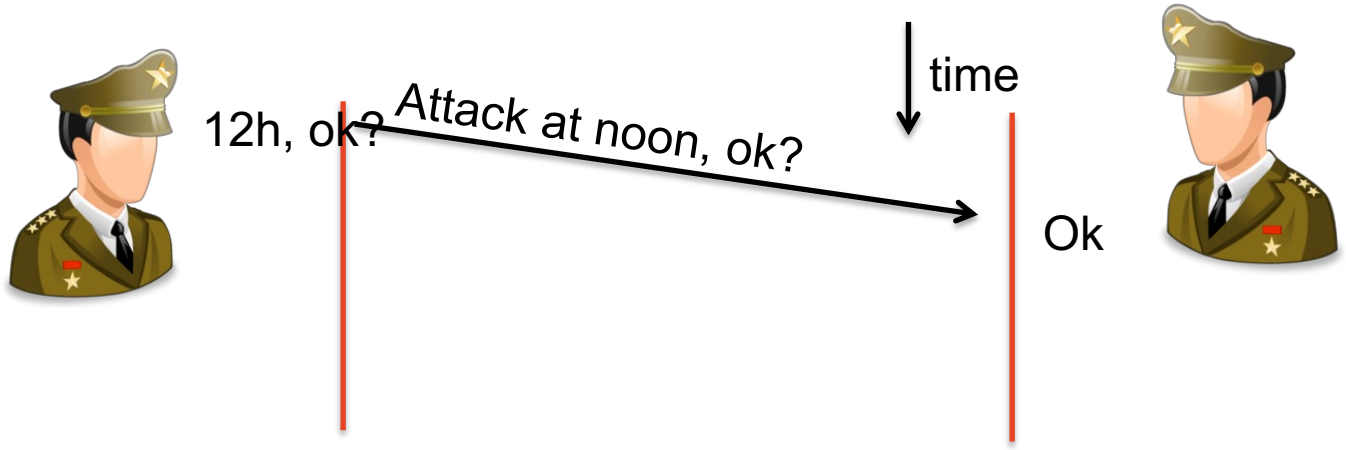
12h!

time

Message loss

Coordinated Attack Problem (con't)

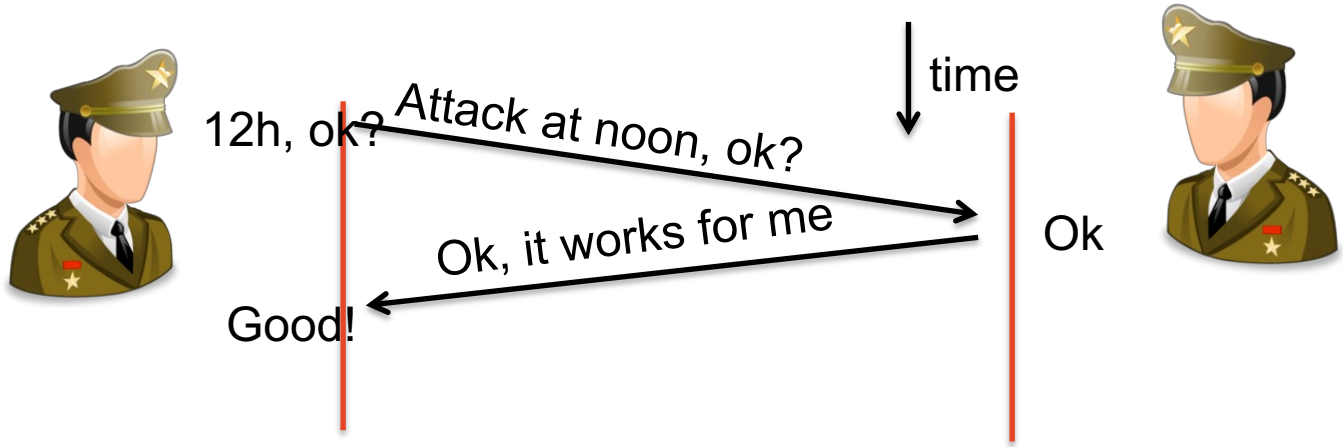
- There is no protocols to make sure they will win!



Message loss

Coordinated Attack Problem (con't)

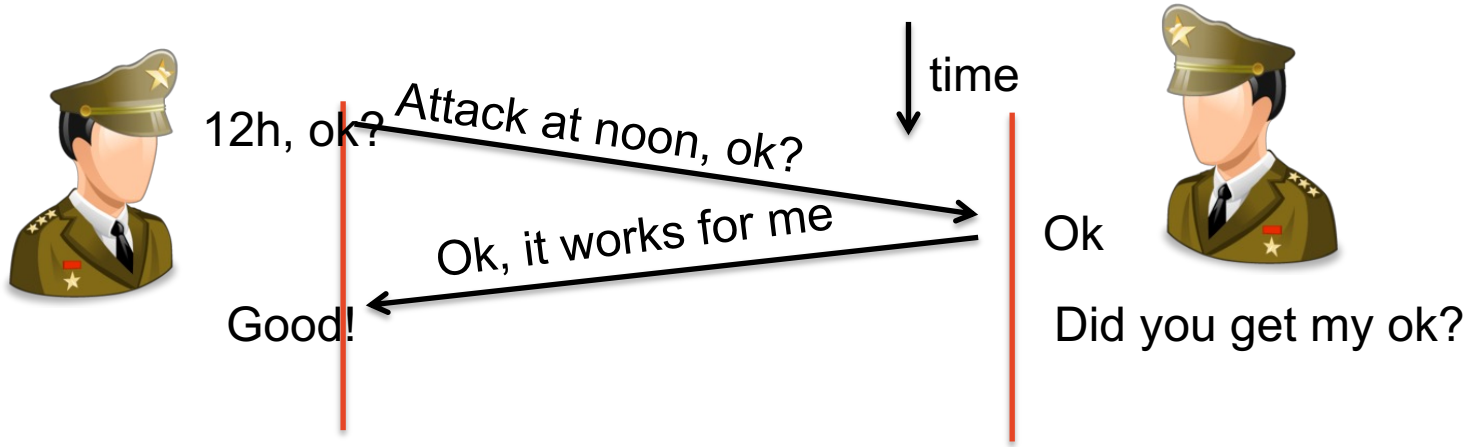
- There is no protocols to make sure they will win!



Message loss

Coordinated Attack Problem (con't)

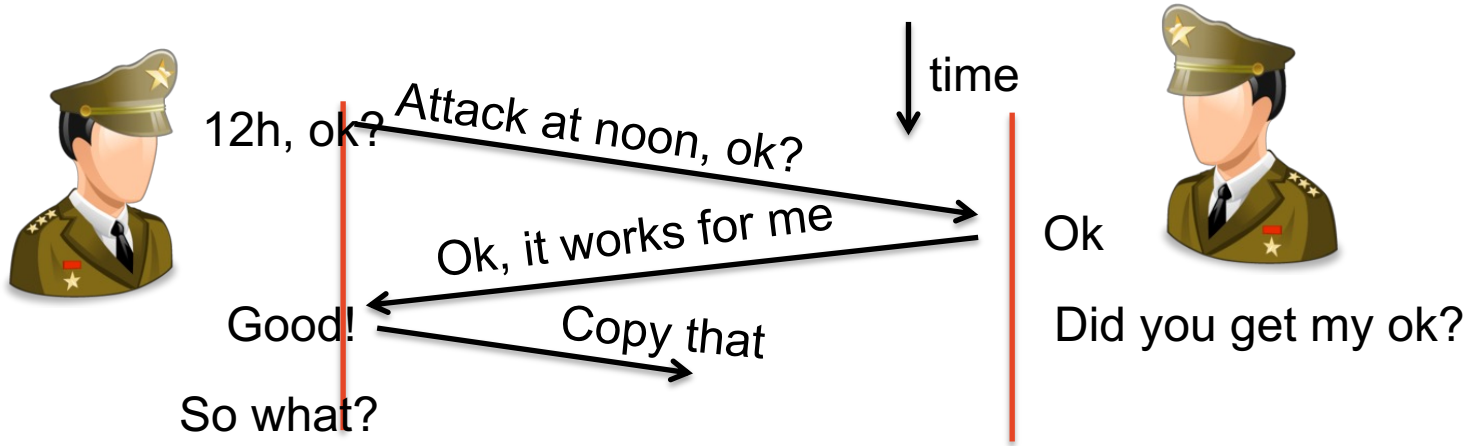
- There is no protocols to make sure they will win!



Message loss

Coordinated Attack Problem (con't)

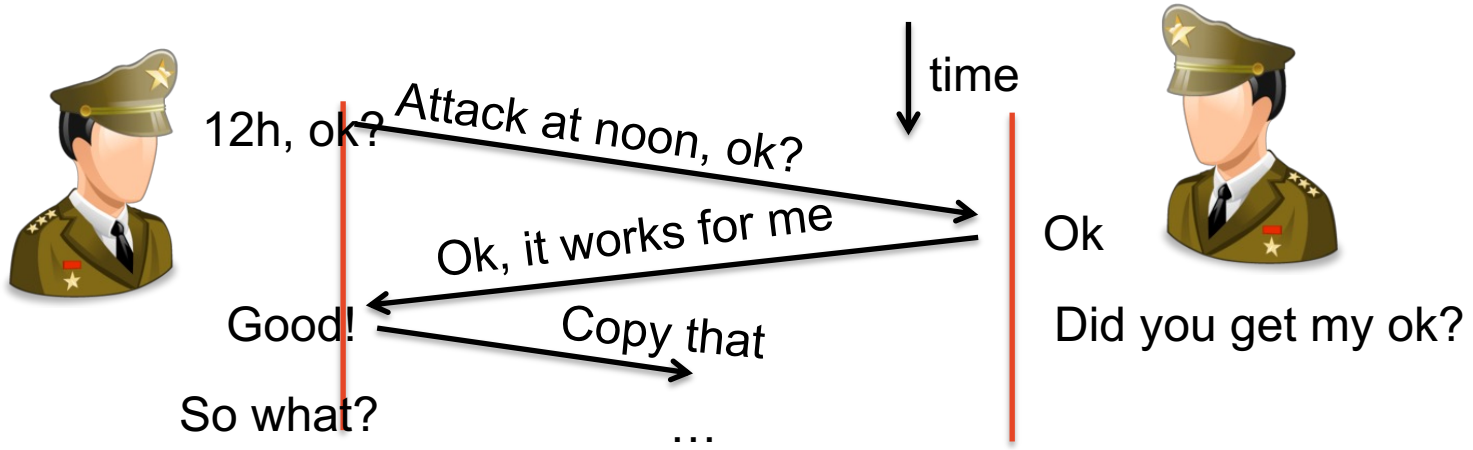
- There is no protocols to make sure they will win!



Message loss

Coordinated Attack Problem (con't)

- There is no protocols to make sure they will win!



Theorem

There is no algorithm that ensures that the generals attack simultaneously

Proof:

1. Consider the algorithm that sends the fewest messages
2. It still works if last message lost
3. So just don't send it (messengers' union happy!)
4. But now we have a shorter protocol!
5. Contradicting #1

Fundamental limitation: We need an unbounded number of messages, otherwise it is possible that no attack takes place!

Message loss

Analogy in networking

- Constraints of the problem
  - Two remote entities of a distributed system
  - Can only communicate through messages
  - The network is unreliable: messages can be dropped
- Goal: they want to make sure to do something simultaneously

This is impossible, even if all messages go through

Asynchronous consensus

Theorem

There is no algorithm that solves consensus in an asynchronous system with even one failure

Proof:

There is no way to distinguish between a delayed message and a failure
Termination requires this distinction

Workaround: some randomized consensus algorithms terminate in expected number of message delays.

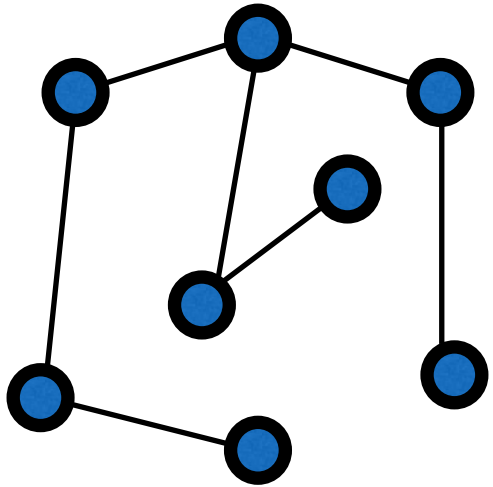
[FLP85] Fischer M. J., Lynch N. A., and Paterson M. S., Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374-382 (1985).

Security

Balance attacks

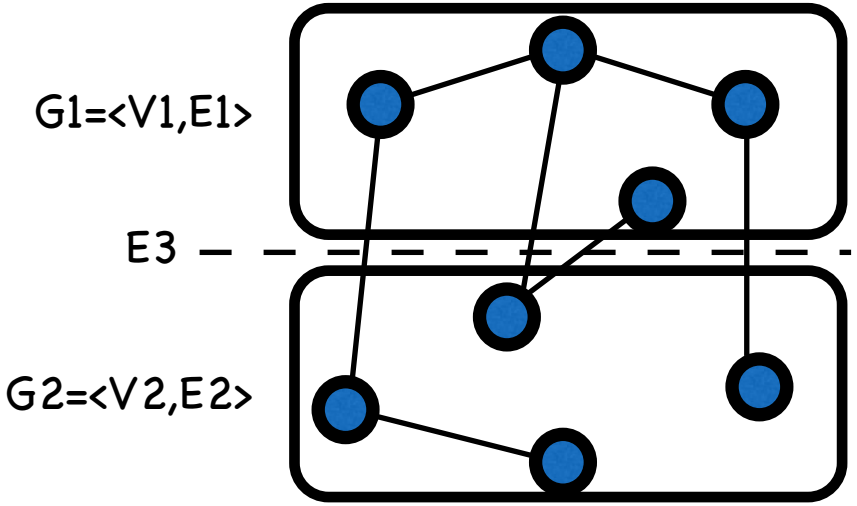
The Balance Attack

Consider a communication graph G=<V,E>



The Balance Attack (con't)

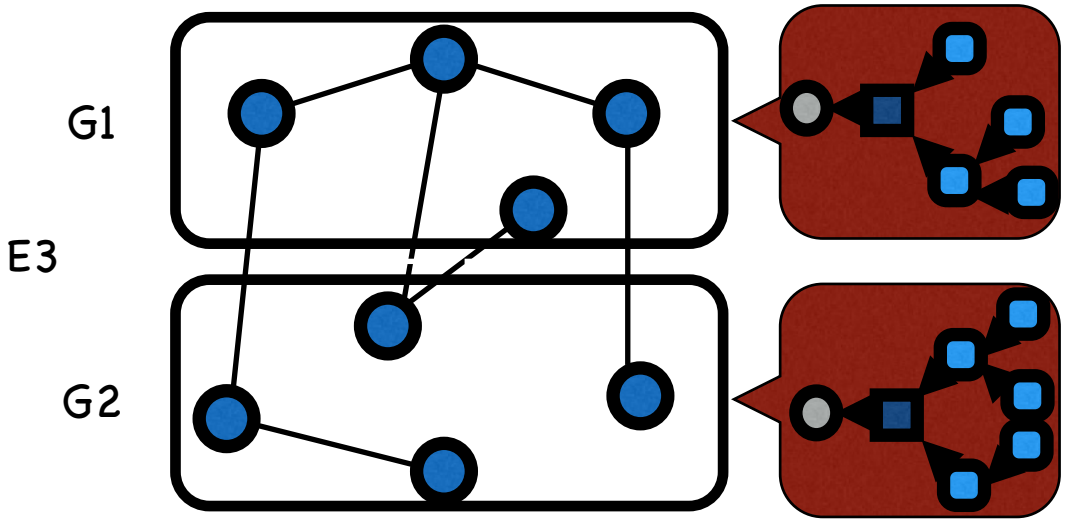
Consider a communication graph G=<V,E>



Let G1 and G2 be two subgraphs of G with the same mining power separated by E3 such that E = E1 ∪ E2 ∪ E3

The Balance Attack (con't)

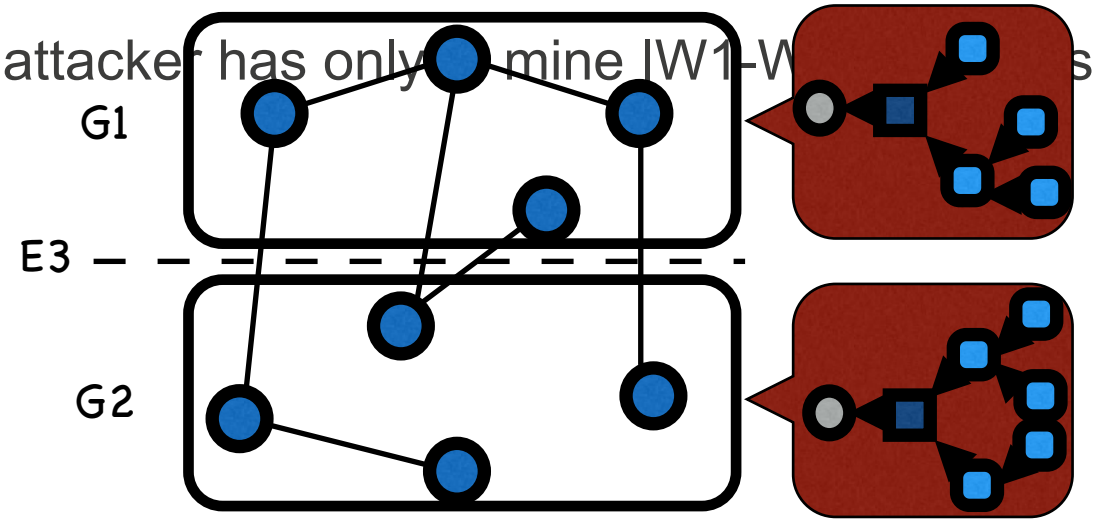
Let an attacker issue t_A in G1 and delay links E3



Then the two subgraphs end up with branches of similar lengths or weights W1 and W2

The Balance Attack (con't)

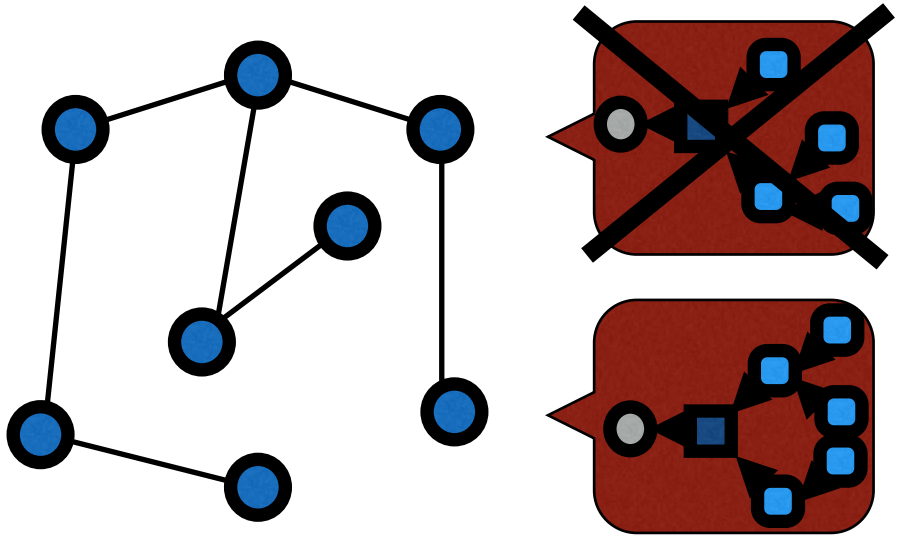
The attacker has only mine |W1-W... in G2



The weight of a branch in G2 exceeds G1's branches

The Balance Attack (con't)

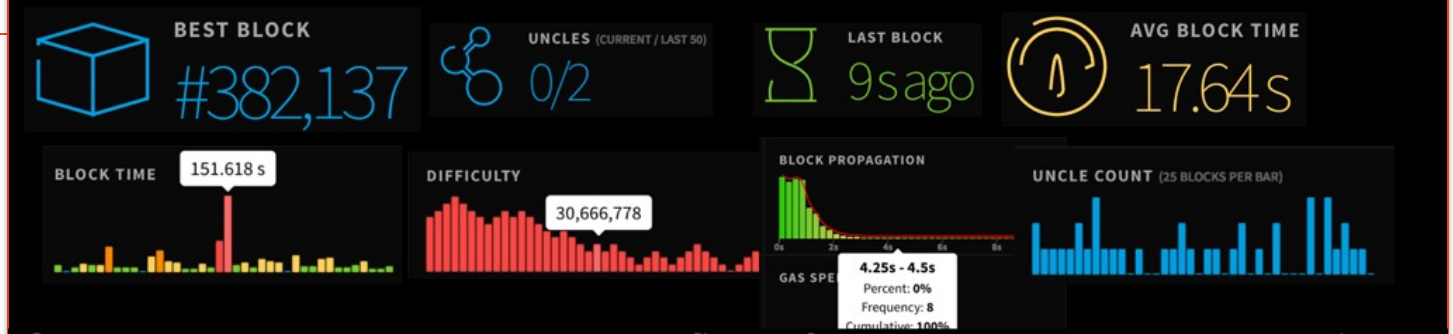
When the attacker stops delaying the messages



t_A is discarded, allowing double spending with t_A'

R3 Ethereum Tesnet

R3 Ethereum Testnet



r3	Geth/v1.4.5-stable/linux/go1.5.1	11 ms	2.8 MH/s	0	0	#376,359	b770e607...c0cc60fa	6,341,516,825,347
bnkasa	Geth/v1.4.5-stable/linux/go1.5.1	4 ms	1.3 MH/s	1	0	#375,704	80a4a9fe...b68d66b8	6,317,896,659,713
cba	Geth/v1.4.5-stable/linux/go1.5.1	7 ms	1.3 MH/s	2	0	#376,776	4b17aae1...e0b7da4f	6,358,160,274,447
bbva	Geth/v1.4.5-stable/linux/go1.5.1	2 ms	1.3 MH/s	1	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
bony	Geth/v1.4.5-stable/linux/go1.5.1	4 ms	1.2 MH/s	1	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
cs	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.2 MH/s	1	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
baml	Geth/v1.4.5-stable/linux/go1.5.1	9 ms	1.2 MH/s	1	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
ubs	Geth/v1.4.5-stable/linux/go1.5.1	2 ms	1.2 MH/s	1	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
ntrs	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.2 MH/s	0	0	#375,570	3fd23e35...a3c94717	6,313,773,575,292
nordea	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.2 MH/s	5	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
ms	Geth/v1.4.5-stable/linux/go1.5.1	4 ms	1.2 MH/s	2	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
cibc	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.2 MH/s	1	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
intesa	Geth/v1.4.5-stable/linux/go1.5.1	2 ms	1.2 MH/s	2	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
r3lab - utils	Geth/v1.4.5-stable/linux/go1.5.1	1 ms	1.1 MH/s	8	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
r3lab	Geth/v1.4.5-stable/linux/go1.5.1	11 ms	344.6 KH/s	3	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
rbc		2 ms		9	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043
danske		2 ms		1	0	#382,166	e0ec2129...e23676b1	6,531,043,823,043

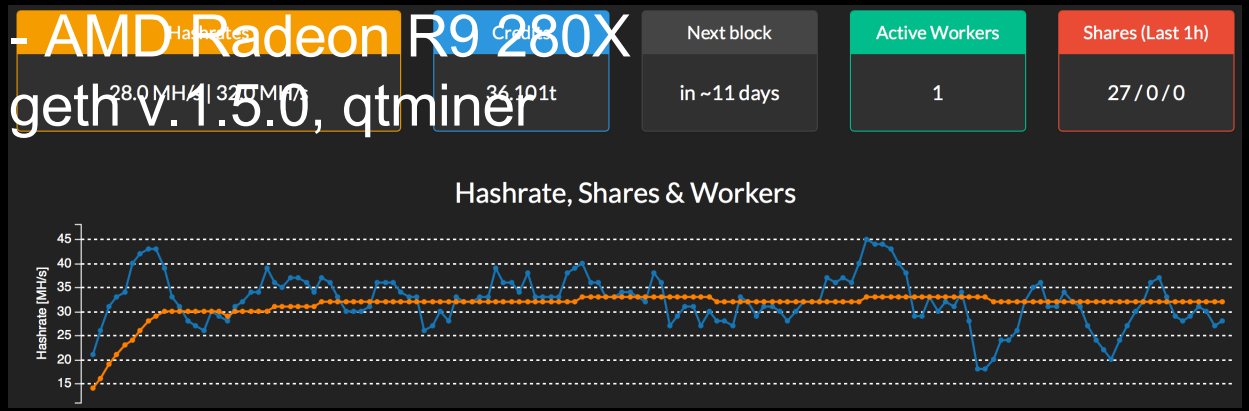
R3 Ethereum Testnet

r3	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	2.4 MH/s	0	0	#376,358	032235b4...7a879e73	6,341,489,326,541
bnkasa	Geth/v1.4.5-stable/linux/go1.5.1	4 ms	1.3 MH/s	0	0	#375,704	80a4a9fe...b68d66b8	6,317,896,659,713
cba	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.3 MH/s	1	0	#376,776	4b17aae1...e0b7da4f	6,358,160,274,447
bony	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.3 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
bbva	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.3 MH/s	2	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
cs	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
baml	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
ntrs	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.2 MH/s	0	0	#375,569	34316d78...30084964	6,313,742,215,265
ubs	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
nordea	Geth/v1.4.5-stable/linux/go1.5.1	5 ms	1.2 MH/s	5	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
ms	Geth/v1.4.5-stable/linux/go1.5.1	7 ms	1.2 MH/s	2	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
intesa	Geth/v1.4.5-stable/linux/go1.5.1	4 ms	1.2 MH/s	2	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
cibc	Geth/v1.4.5-stable/linux/go1.5.1	8 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
r3lab - utils	Geth/v1.4.5-stable/linux/go1.5.1	1 ms	1.1 MH/s	8	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
r3lab	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	344 KH/s	3	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
rbc		6 ms		9	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
danske		2 ms		1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
bmo		2 ms		1	0	#376,776	4b17aae1...e0b7da4f	6,358,160,274,447
stt		6 ms		1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
macq		1 ms		0	0	#319,864	9076a734...bc5c0180	3,845,547,329,094
nab		3 ms		0	0	#355,464	b7ec6fde...6db75642	5,453,765,160,440
nomura		8 ms		5	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228

20MH/s

My miner

4-core Intel Core i5-6500, 3.2GHz
Ubuntu 14.04
2 GPUs:
- AMD Radeon R9 290X
- AMD Radeon R9 280X
geth v.1.5.0, qtminer



Analysis of the Balance Attack

Analysis

As there is no other strategy than brute force to solve the crypto-puzzle, each subgraph performs, during delay τ, a series of n independent Bernoulli trials returning:

- 1 with probability p = (1-ρ)t/(2d) and
- 0 otherwise

The numbers X1 and X2 of blocks have a binomial distribution with mean:

$$\mu_c = np = \frac{(1-\rho)t\tau}{2d}.$$

…where is the ρt is the mining power of the attacker.

Analysis

We can upper-bound the deviation of the number of blocks mined Xi (i∈{1,2}) from their mean using Chernoff bounds [Motwani and P. Raghavan 1995] by

$$\Pr[|X_i - \mu_c| < \delta\mu_c] \quad > \quad 1 - 2e^{-\frac{\delta^2}{3}\mu_c}.$$

Let Δ=|X1-X2| be the difference of the number of blocks mined in each subgraph

Analysis

Observe that the probability that these random variables are within a $\pm\delta\mu_c$ is lower than the probability that their difference Δ is upper-b

$$\Pr[\Delta < 2\delta\mu_c] > \left(1 - 2e^{-\frac{\delta^2}{3}\mu_c}\right)^2.$$

and Bernoulli inequality gives us:

$$\Pr[\Delta < 2\delta\mu_c] > 1 - 4e^{-\frac{\delta^2}{3}\mu_c}.$$

Analysis

By choosing $\delta = 2\rho/(1-\rho)$ we obtain that the expectation of the number of blocks mined by the malicious miner μ_m is strictly greater than A with probability

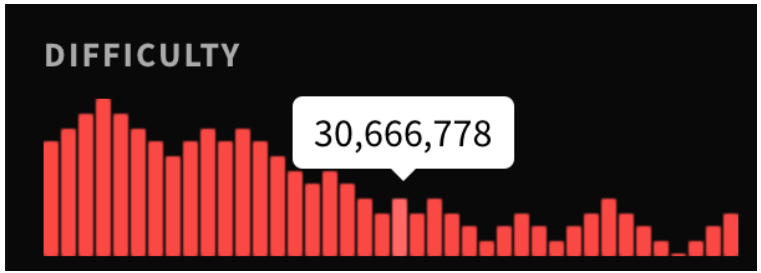
$$\Pr[\Delta < \mu_m] \quad > \quad 1 - 4e^{-\frac{4\rho^2 \mu_c}{3(1-\rho)^2}}.$$

With communication delay $\tau \geq \frac{(1-\rho)6d\log(\frac{4}{\epsilon})}{4\rho^2 t}$

$$\Pr[\Delta < \mu_m] \quad > \quad 1 - \epsilon.$$

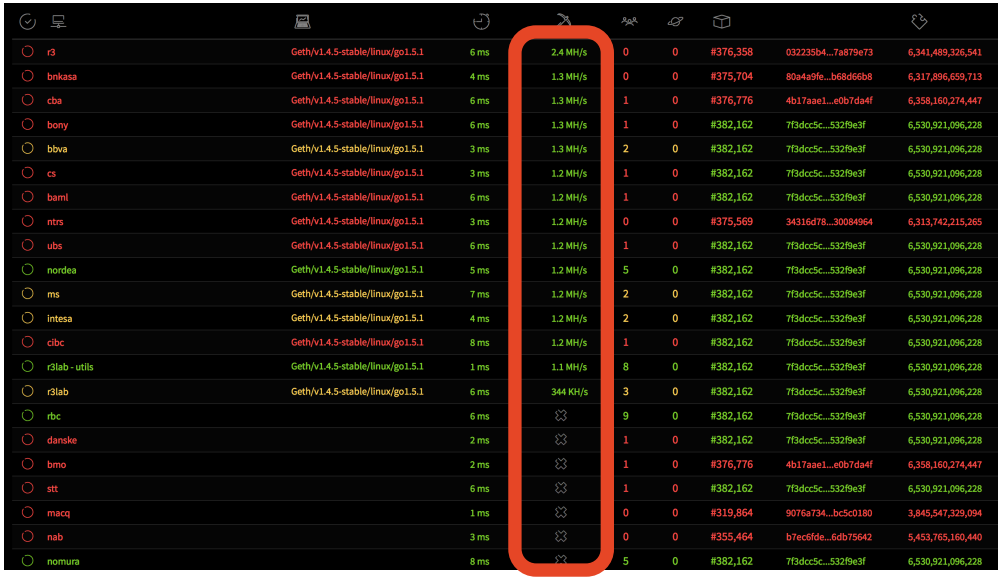
Example with Banks

Consider an Ethereum/GHOST private chain with difficulty 30MH…



Example with Banks

Consider an Ethereum/GHOST private chain with difficulty 30MH and a mining power of 20MH/s.



r3	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	2.4 MH/s	0	0	#376,358	032235b4...7a879e73	6,341,489,326,541
bnkasa	Geth/v1.4.5-stable/linux/go1.5.1	4 ms	1.3 MH/s	0	0	#375,704	80a4a9fe...b68d66b8	6,317,896,659,713
cba	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.3 MH/s	1	0	#376,776	4b17aae1...e0b7da4f	6,358,160,274,447
bony	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.3 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
bbva	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.3 MH/s	2	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
cs	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
baml	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
ntrs	Geth/v1.4.5-stable/linux/go1.5.1	3 ms	1.2 MH/s	0	0	#375,569	34316d78...30084964	6,313,742,215,265
ubs	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
nordea	Geth/v1.4.5-stable/linux/go1.5.1	5 ms	1.2 MH/s	5	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
ms	Geth/v1.4.5-stable/linux/go1.5.1	7 ms	1.2 MH/s	2	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
intesa	Geth/v1.4.5-stable/linux/go1.5.1	4 ms	1.2 MH/s	2	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
cibc	Geth/v1.4.5-stable/linux/go1.5.1	8 ms	1.2 MH/s	1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
r3lab - utils	Geth/v1.4.5-stable/linux/go1.5.1	1 ms	1.1 MH/s	8	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
r3lab	Geth/v1.4.5-stable/linux/go1.5.1	6 ms	344 KH/s	3	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
rbc		6 ms		9	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
danske		2 ms		1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
bmo		2 ms		1	0	#376,776	4b17aae1...e0b7da4f	6,358,160,274,447
stt		6 ms		1	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228
macq		1 ms		0	0	#319,864	9076a734...bc5c0180	3,845,547,329,094
nab		3 ms		0	0	#355,464	b7ec6fde...6db75642	5,453,765,160,440
nomura		8 ms		5	0	#382,162	7f3dcc5c...532f9e3f	6,530,921,096,228

~20MH/s

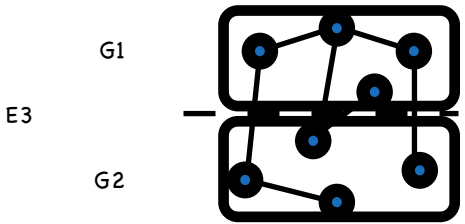
Example with Banks

Consider an Ethereum/GHOST private chain with difficulty 30MH and a mining power of 20MH/s. Assume the attacker has 12% of the mining power, i.e., 2.4MH/s, so he can mine, in expectation, 96 blocks during 20 minutes…

12%

Example with Banks

Consider an Ethereum/GHOST private chain with difficulty 30MH and a mining power of 20MH/s. Assume the attacker has 12% of the mining power, i.e., 2.4MH/s, so he can mine, in expectation, 96 blocks during 20 minutes and that he can select E3 such that each subgraph has a mining power of 8.8MH/s.

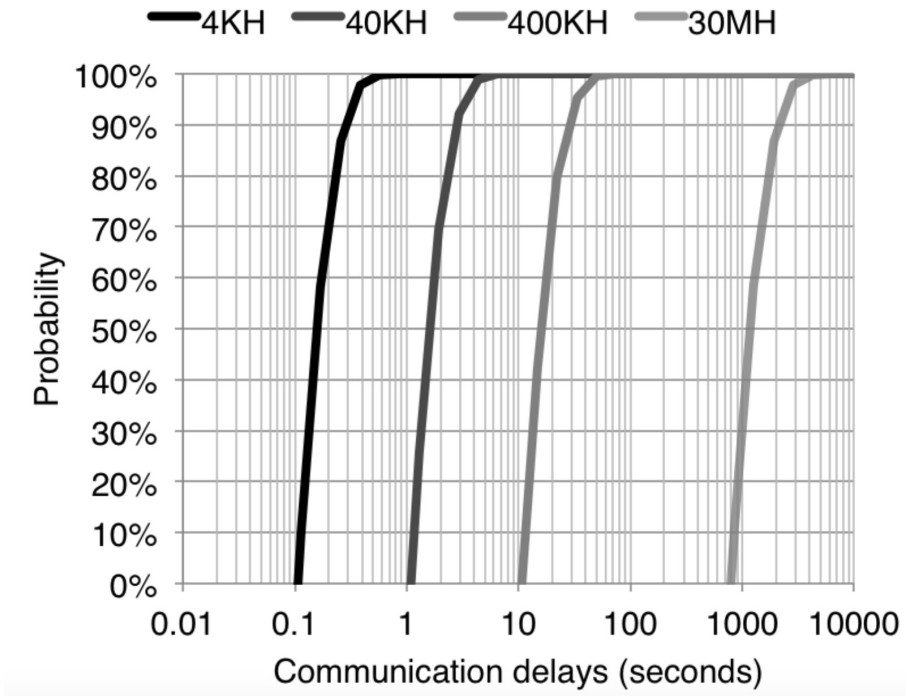


Example with Banks

Consider an Ethereum/GHOST private chain with difficulty 30MH and a mining power of 20MH/s. Assume the attacker has 12% of the mining power, i.e., 2.4MH/s, so he can mine, in expectation, 96 blocks during 20 minutes and that he can select E3 such that each subgraph has a mining power of 8.8MH/s.
Let us select $\delta = 95/(2\mu_c) \simeq 0.13$ for the difference Δ to be lower than 96, which happens with probability

$$P[\Delta < 96] > 52\%$$

Probability of Attack Success per Difficulties



Number of nodes	50
Number of miners	15
Total mining power (MH/s)	20
Mining power of the most powerful miner (MH/s)	2.4
Difficulty (MH)	30

[NG17] Natoli C., Gramoli V. The Balance or Why Proof-of-Work is Ill-Suited for Consortium Blockchains. *DSN 2017*.

Synchrony

“This is not at all novel. This basically just points out that proof of work has a synchrony assumption.”

Vitalik Buterin

Synchrony

“This is not at all novel. This basically just points out that proof of work has a synchrony assumption.”

Vitalik Buter

Feasibility of the Balance Attack

Network attacks

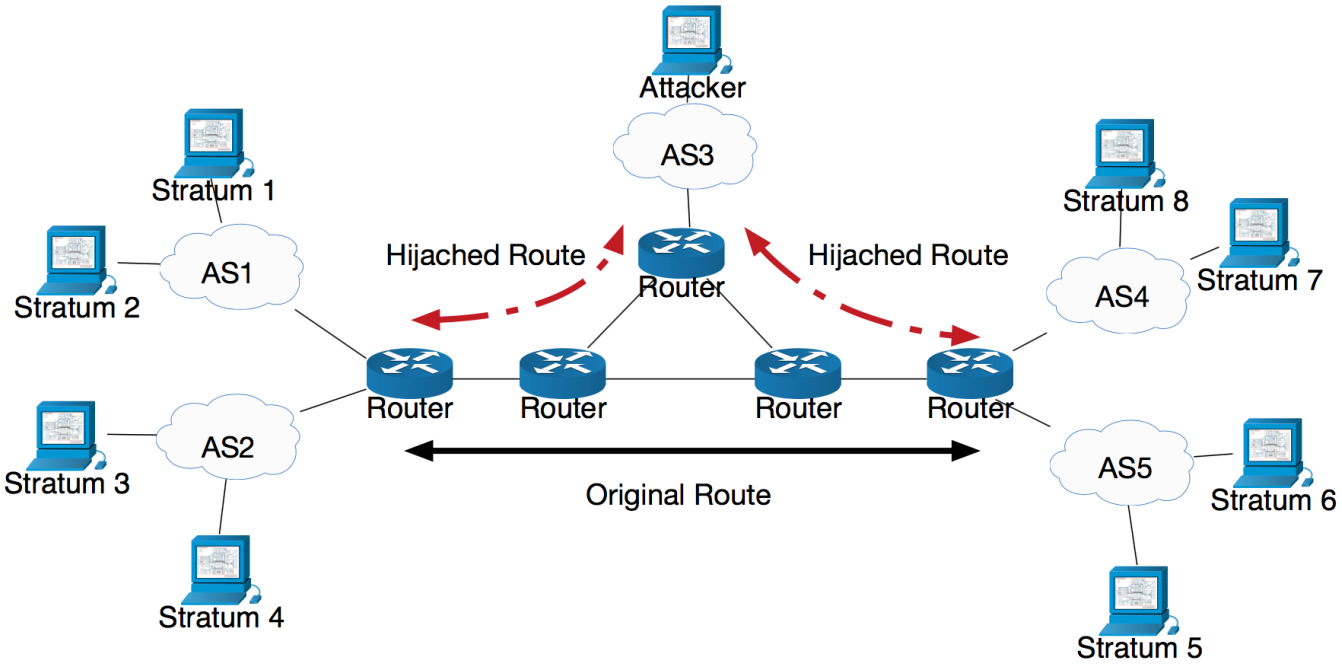
The Balance Attack requires network delays:

- Man-in-the-middle attack
- DoS attack
- BGP hijacking attack
- ARP spoofing attack, …

Ethereum can run as:

- a public chain
- a consortium blockchain
- a private blockchain

BGP Hijacking Attack



Ethereum mining pools

Power (%)	Pool Name	Stratum Servers	Location	ASN	AS Owner
27.02	f2pool	eth.f2pool.com	Hangzhou, China	37963	Alibaba (China) Technology Co., Ltd.
23.76	Ethermine	us1.ethermine.org	Montreal, Canada	16276	OVH SAS
		us2.ethermine.org	California, US	63949	Linode, LLC
		eu1.ethermine.org	France	16276	OVH SAS
		eu2.ethermine.org	France	16276	OVH SAS
		asia1.ethermine.org	Singapore	16276	OVH SAS
9.73	miningpoolhub	us-east.ethash-hub.miningpoolhub.com	Georgia, US	63949	Linode, LLC
		europe.ethash-hub.miningpoolhub.com	Hesse, Germany	63949	Linode, LLC
		asia.ethash-hub.miningpoolhub.com	Tokyo, Japan	63949	Linode, LLC
9.7	Nanopool	eth-eu1.nanopool.org	France	16276	OVH SAS
		eth-eu2.nanopool.org	France or Italy	16276	OVH SAS
		eth-asia1.nanopool.org	Singapore	16276	OVH SAS
		eth-us-east1.nanopool.org	Montreal, Canada	16276	OVH SAS
		eth-us-west1.nanopool.org	California, US	20473	Choopa, LLC
9.12	ethfans	guangdong-pool.ethfans.org	Fujian, China	4134	No.31,Jin-rong Street
		huabei-pool.ethfans.org	Fujian, China	4134	No.31,Jin-rong Street
6.24	DwarfPool	eth-eu.dwarfpool.com	France	16276	OVH SAS
		eth-us.dwarfpool.com	Montreal, Canada	16276	OVH SAS
		eth-us2.dwarfpool.com	Las Vegas, US	53667	FranTech Solutions
		eth-ru.dwarfpool.com	France	16276	OVH SAS
		eth-asia.dwarfpool.com	Taiwan	59253	Leaseweb Asia Pacific pte. ltd.
		eth-cn.dwarfpool.com	Shanghai, China	37963	Alibaba (China) Technology Co., Ltd.
		eth-cn2.dwarfpool.com	Beijing, China	37963	Alibaba (China) Technology Co., Ltd.
		eth-sg.dwarfpool.com	Singapore	59253	Leaseweb Asia Pacific pte. ltd.
		eth-au.dwarfpool.com	Melbourne, Australia	38880	Micron21 Melbourne Australia Datacentre
		eth-ru2.dwarfpool.com	Moscow, Russia	42632	MnogoByte LLC
		eth-hk.dwarfpool.com	Hong Kong	45102	Alibaba (China) Technology Co., Ltd.
		eth-br.dwarfpool.com	Sao Paulo, Brazil	262287	Maxihost Hospedagem de Sites Ltda
		eth-ar.dwarfpool.com	Rosario, Argentina	27823	Dattatec.com
4.45	BW	ether.bw.com	Wuhan, China	58563	CHINANET Hubei province network
3.34	Ethpool	us1.ethpool.org	Montreal, Canada	16276	OVH SAS
		us2.ethpool.org	Montreal, Canada	16276	OVH SAS
		eu1.ethpool.org	France	16276	OVH SAS
		asia1.ethpool.org	Singapore	16276	OVH SAS
1.83	Coinotron	coinotron.com	Poland	51290	HOSTEAM-AS
0.88	Poolgpu	eth.poolgpu.com	Hangzhou, China	37963	Alibaba (China) Technology Co., Ltd.
96.07	Total				

Ethereum pools power is sufficient to double spend

Mining power of members in an adversary subgroup (%)		Mining power of members in a victim subgroup (%)	Difference between two subgroups	Attack Success rate (%)
Adversary	The rest of subgroup			
27.02	23.76, 6.24, 3.34, 0.88	9.73, 9.7, 9.12, 4.45, 1.83	26.41	76.7
23.76	27.02, 6.24, 1.83, 0.88	9.73, 9.7, 9.12, 4.45, 3.34	23.39	63.3
9.73	27.02, 9.12, 4.45, 1.83, 0.88	23.76, 9.7, 6.24, 3.34	9.99	56.7
9.7	27.02, 9.12, 4.45, 1.83, 0.88	23.76, 9.73, 6.24, 3.34	9.93	43.3
9.12	27.02, 9.7, 4.45, 1.83, 0.88	23.76, 9.73, 6.24, 3.34	9.93	43.3
6.24	27.02, 9.7, 4.45, 3.34	23.76, 9.73, 9.12, 1.83, 0.88	5.43	40
4.45	27.02, 9.7, 6.24, 1.83, 0.88	23.76, 9.73, 9.12, 3.34	4.17	40
3.34	27.02, 9.7, 6.24, 1.83, 0.88	23.76, 9.73, 9.12, 4.45	1.95	43.3
1.83	27.02, 9.7, 6.24, 3.34, 0.88	23.76, 9.73, 9.12, 4.45	1.95	33.3
0.88	27.02, 9.7, 6.24, 3.34, 1.83	23.76, 9.73, 9.12, 4.45	1.95	26.7

Attack success with the mining power of the 10 largest mining pools of Ethereum at the end of July 2017

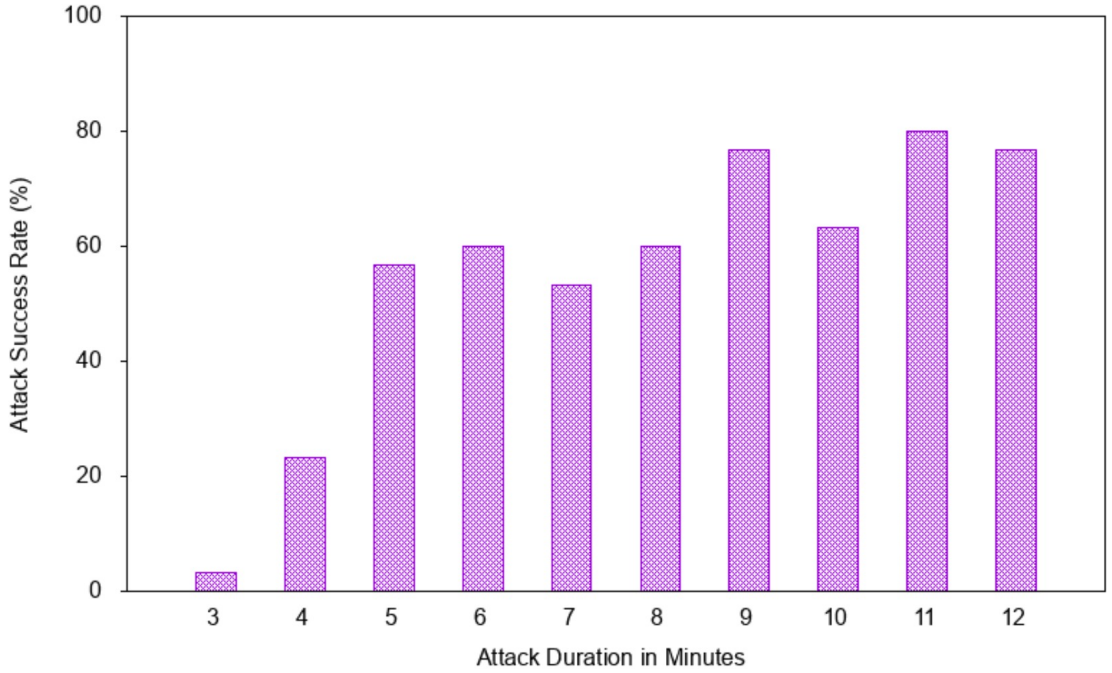
On the difficulty of partitioning mining pools

Mining pools often operate multiple stratum servers
These servers are in same regions and different ASes

Disconnected miners simply try to reconnect to the next stratum server

This makes it hard to partition the network as the miner need to target multiple ASes

Feasibility of the attack on consortium Ethereum



[EGJ18] Ekparinya P., Gramoli V., Jourjon G. Impact of Man-in-the-Middle Attacks on Ethereum. *IEEE SRDS 2018*.

Quiz

Question 1

How to make sure that a transaction is committed in Ethereum or Bitcoin?

Scalability

Leader-based Consensus

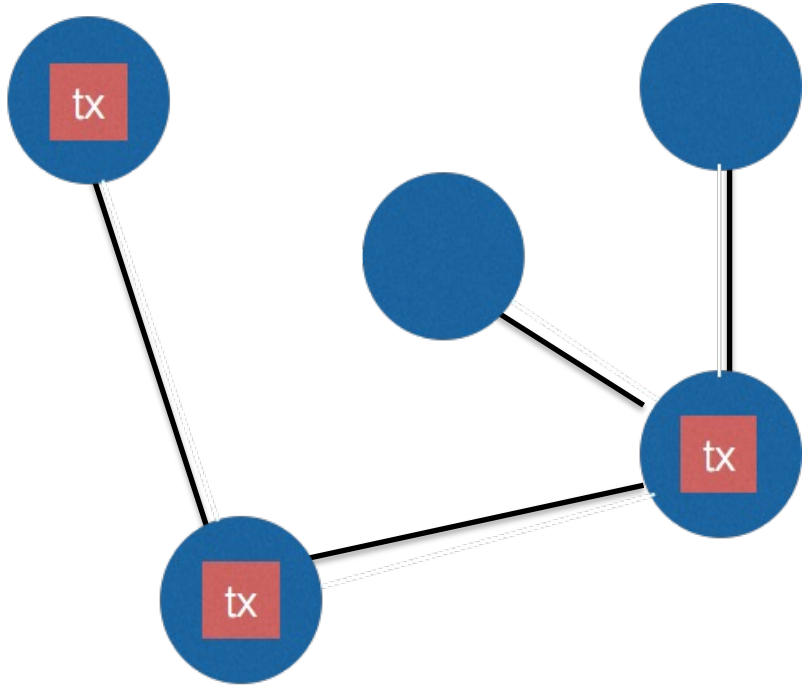
Leader-based consensus in blockchains

The process that plays the role of the leader changes over time.

It's hard for a client (e.g. wallet) to know which process is the leader to send a request to.

Easier to:

- send transaction requests to the processes that will run the consensus
- send their request to the processes that are the closest geographically



Leader vs. leaderless consensus algorithms

Consider a consensus algorithm that decides a *set of transactions* of B bits

We want to compare the performance of:

- A leader-based solution
- A leaderless solution

In a leader-based algorithm, the set of transactions is proposed by the leader

In a leaderless algorithm, the set of transactions is proposed by all the participating processes

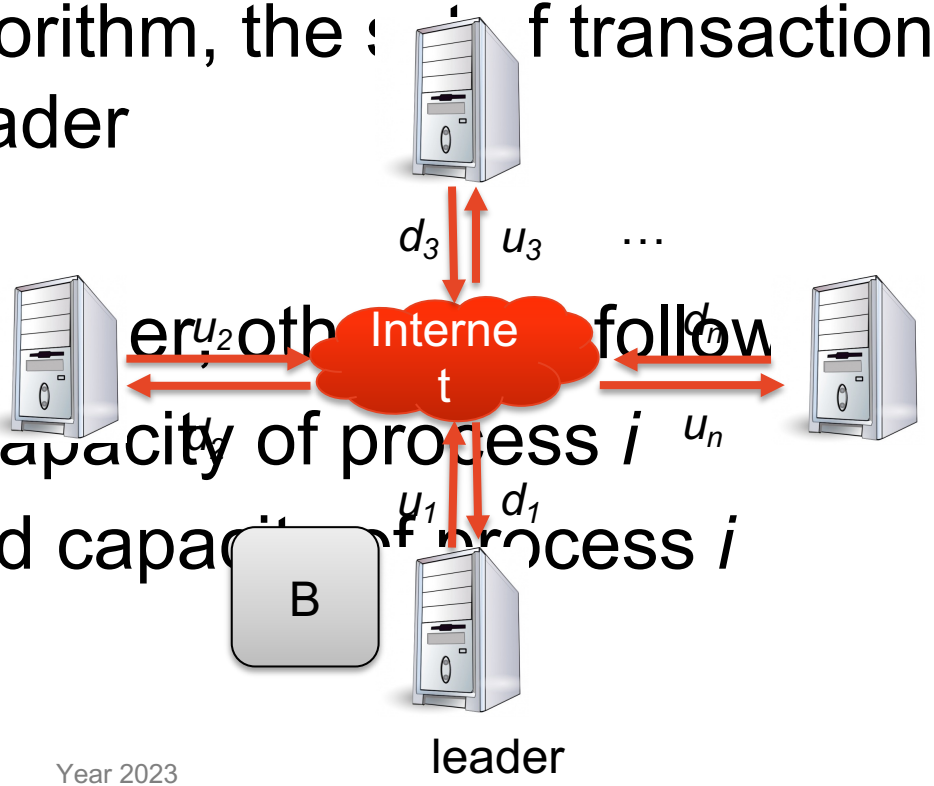
Block exchange time in leader-based algorithms

In a leader-based algorithm, the s f transactions is proposed by the leader

Let process 1 be the er; oth follow

Let u_i be the upload capacity of process i

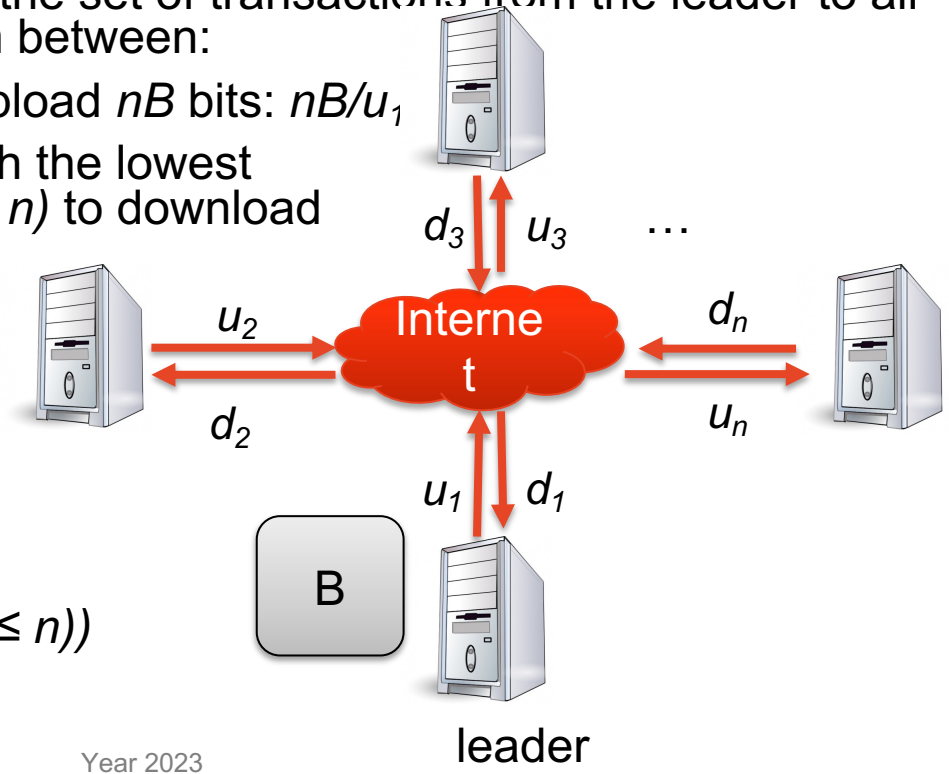
Let d_i be the download capac process i



Block exchange time in leader-based algorithms (~~con't~~)

The time P it takes to exchange the set of transactions from the leader to all other processes is the maximum between:

- The time for the leader to upload nB bits: nB/u_1
- The time for the follower with the lowest download rate $min(d_i: 1 \le i \le n)$ to download B bits: $B/min(d_i: 1 \le i \le n))$



$\Rightarrow P = max(nB/u_1, B/min(d_i: 1 \le i \le n))$

$= O(nB)$

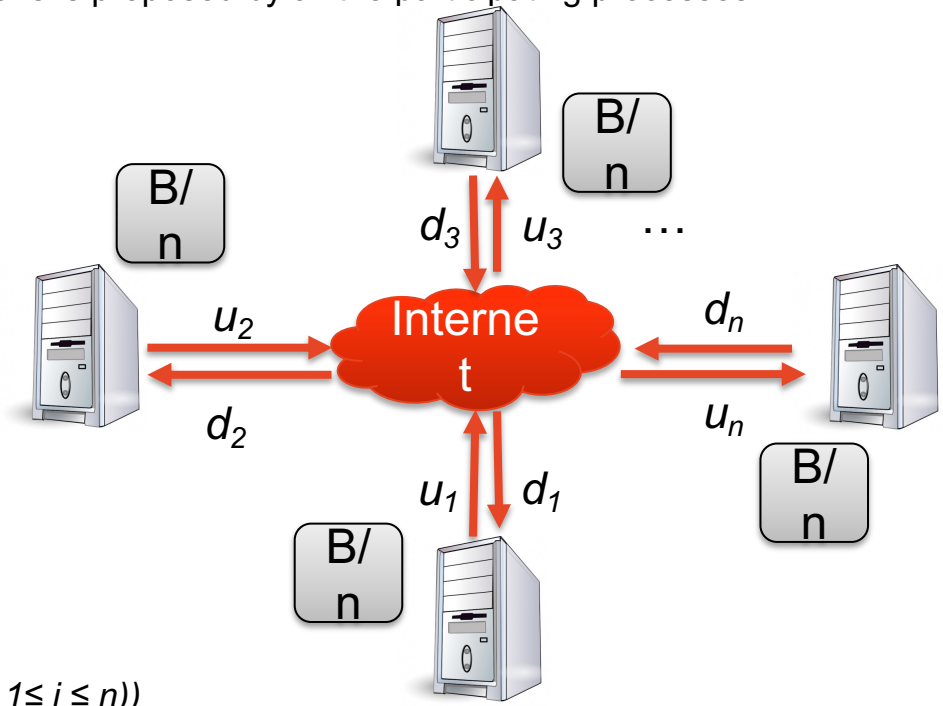
Block exchange time in leaderless algorithms

In a leaderless algorithm, the set of transactions is proposed by all the participating processes

Consider that each process has a portion *B/n* of the total set of transactions

The time it takes to exchange these sets of transactions is the max of:

- The time for the slowest process to upload *B/n* bits
- The time for the slowest process to download *B* bits



$\Rightarrow P = max((B/n)/min(u_i : 1 \leq i \leq n), (B/min(d_i : 1 \leq i \leq n))$

$= O(B)$

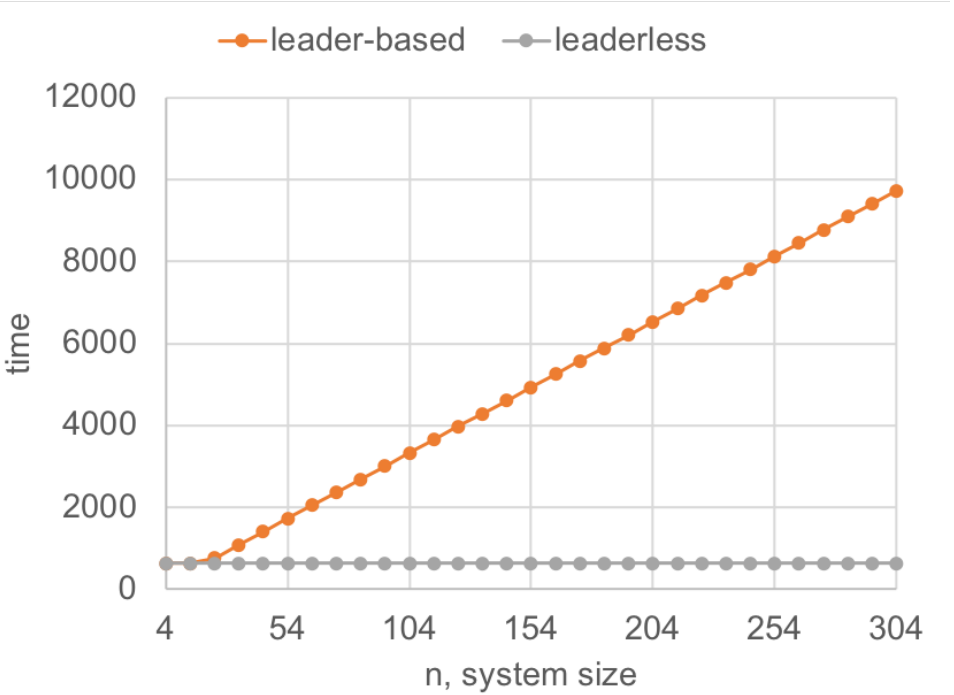
Leader vs. leaderless block exchange time

Consider that the leader has 10 times higher upload rate than other processes

And that every process download 2 times faster than it uploads

The time it takes for all processes to be aware of the set of transactions is the same on a small network

When the number of processes reaches 21, then the leaderless algorithm starts being faster and faster



The Set Consensus Problem

Traditional Consensus does not Scale

Two reasons why traditional consensus does not scale:

1. The leader tries to impose its block, disallowing other nodes from proposing blocks
2. The time to propagate the block to the followers is limited by:
   - the upload capacity of the leader and
   - the lowest download capacity of the followers

Defining Blockchain Consensus

Instead of proposing blocks, processes propose sets of transactions that will be included in the block:

1. All processes propose a set of transactions
2. They exchange these sets among each other
3. They decide a block that:
   - Is not necessarily what any process has proposed
   - Can result from the union of several proposed sets of transactions

Blockchain Consensus

A set of transactions is *non-conflicting* if it does not contain conflicting transactions.

Assuming that each correct node proposes a proposal s, the Set Byzantine Consensus (SBC) problem is for each of them to decide on a set in such a way that the following properties are satisfied:

- SBC-Termination: every correct node eventually decides a set of transactions;
- SBC-Agreement: no two correct nodes decide on different sets of transactions;
- SBC-Validity: a decided set of transactions is a non-conflicting set of valid transactions taken from the union of the proposed sets; and if all nodes are correct and propose a common valid non-conflicting set of transactions, then this subset is the decided set.

Democratic Byzantine Fault Tolerance

Democratic Byzantine Fault Tolerance (DBFT)

- est: local current estimate of the value to decide
- r: local round number, initially 0
- bin-values[]: array of binary values for all rounds
- b, auxiliary binary value
- $values_i$, auxiliary set of values
- The algorithm uses two message types
  EST[r]() used at round r to BV-bcast est.
  AUX[r]() used at round r to broadcast bin-values[r]

DBFT (con't)

BV-broadcast [JACM'15]

- BV-obligation: if t+1 correct BV-bcast v, then v is eventually added to the set bin-values of all correct processes
- BV-justification: if pi is correct and v in bin-values$_i$, then v was BV-bcast by a correct process
- BV-uniformity: if v is added to bin-values$_i$ of correct p_i, then eventually v will be in bin-values$_j$ for all correct p_j
- BV-termination: eventually bin-values of correct p_i is not empty

Binary Consensus

```
1. est ← v
2. r ← 0
3. while (true) {
4.   r ← r+1
5.   BV-bcast EST[r](est)
6.   wait until (bin-values[r] ≠ ∅)
7.   bcast AUX[r](bin-values[r])
8.   wait until (messages AUX[r](bval_p1), …, AUX[r](bval_p(n-t)) received
9.     from (n-t) distinct processes p(x), 1 ≤ x≤n-t, and their content is such that:
10.    ∃ a non-empty set values_i such that
11.                (i) values_i ⊆ bin-values[r]_i and
12.                    (ii) values_i = U b_val_x
13.  b ← r mod 2
14.  if (values_i = {v}) {
15.    est ← v
16.    if (v=b) then decide(v) and exit (if not done yet)
17.  } else {
18.    est ← b
19.  }
20.}
```

Binary Consensus (con't)

```
1. est ← v
2. r ← 0
3. while (true) {
4.   r ← r+1
5.   BV-bcast EST[r](est)
6.   wait until (bin-values[r] ≠ ∅)
7.   bcast AUX[r](bin-values[r])
8.   wait until (messages AUX[r](bval_p1), …, AUX[r](bval_p(
9.     from (n-t) distinct processes p(x), 1 ≤ x≤n-t, and …  is such that:
10.    ∃ a non-empty set values_i such that
11.                (i) values_i ⊆ bin-values[r]_i and
12.                    (ii) values_i = U b_val_x
13.  b ← r mod 2
14.  if (values_i = {v}) {
15.    est ← v
16.    if (v=b) then decide(v) and exit (if not done yet)
17.  } else {
18.    est ← b
19.  }
20.}
```

Filter out values proposed only by Byzantine processes

Agreement proof

Lemma 1: If at the beginning of a round, all correct processes have the same estimate, they never change their estimate thereafter.

Proof sketch. Assume all correct have the same estimate v at round r. BV-Obligation and BV-Justification => bin_values[r] = {v}. Hence, at every correct we have values = {v}, so that est becomes v.

Agreement proof (con't)

Lemma 2: Let pi and pj be correct. If their values are singletons, then they are the same.

Proof sketch. If a correct process has values = {v} then it received AUX[r]{v} from n-t distinct processes and t+1 correct. For two correct processes to have w and v as this singleton, it would mean they received these distinct values from n-t processes each. As (n-t)+(t+1) > n, one correct process must have sent the same value to both, and we have v=w.

Agreement proof (con't)

Theorem [Agreement]: No two correct processes decide different values.

Proof sketch. Let r be the 1st round where a correct process decides, hence values[r] = {v=r mod 2}. If another correct process decides w in the same round, then v=w by Lemma 2. Let pj be a correct that does not decide in round r. By Lemma 2, $values_j \neq \{w\}$ so $values_j = \{0,1\}$. Thus $est_j = v$. All correct have thus the same estimate in round r+1, and stick to it (Lemma 1).

Safe version of DBFT

```
BC, binary consensus instance


mv-propose(v) { // similar to [PODC'94]
  RB-bcast VAL(v) // reliable broadcast [Bra'87]
  repeat if (∃ k : proposals[k]≠⊥ ∧ BC[k].binpropose() not invoked)
    BC[k].binpropose(1)
  until(∃ l: bin-decisions[l]=1)
  for each k such that BC[k].binpropose() not invoked:
    BC[k].binpropose(0)
  wait until (∧1≤x≤n bin-decisions[x]≠⊥)
  j = min{x : bin-decisions[j] = 1}
  wait until proposals[j] ≠⊥
  decide(U∀j:bin-decisions[j]=1proposals[j])
}

when val(v) is RB-delivered from pj do // reliable broadcast delivery
  if valid(v) then
    proposals_i[j] <- v;
    BV-deliver b-val[1](1) to BC[j]


when BC[k].binpropose() returns b do bin-decisions[k] <- b
```

Spawn multiple binary cons. instances concurrently

Use binary decisions as a bitmask

Redbelly Blockchain

The Transaction Verification Cost

We saw how to leverage bandwidth for the consensus solution to scale

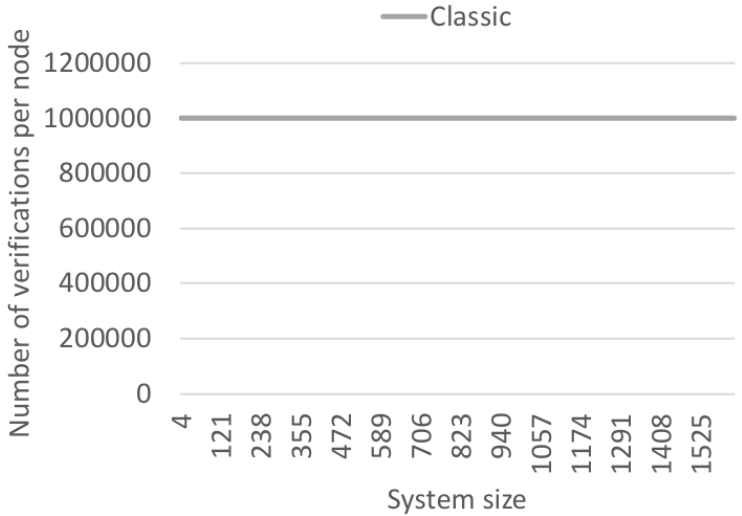
The Set Byzantine Consensus problem is well-suited for blockchains but requires a verification of signatures to check that a transaction is valid

This verification typically uses a public-key cryptosystem, that is CPU intensive and costly.

We will see how to lower the cost of verification to build a scalable blockchain

CPU waste

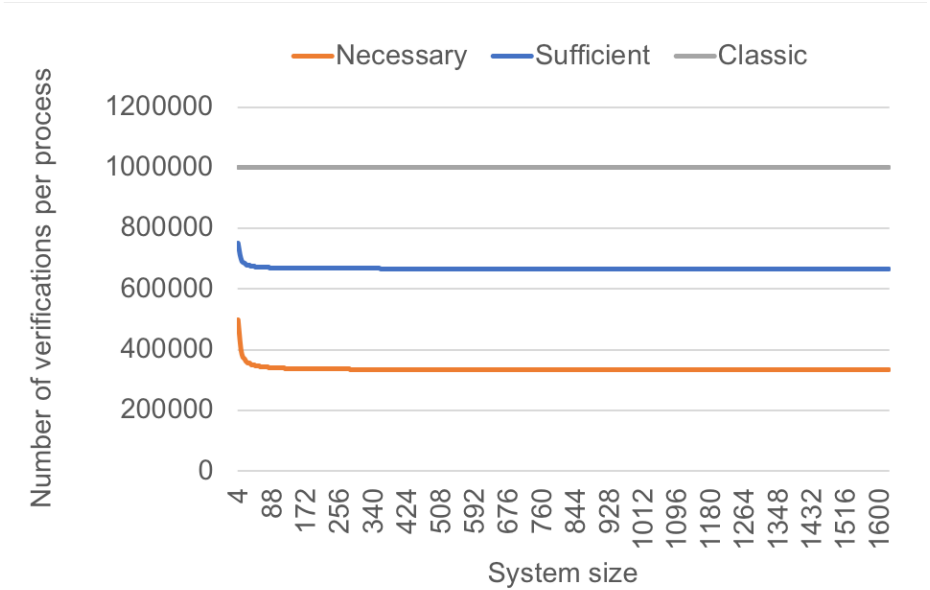
In traditional blockchains, all miners verify all transactions, which limits scalability



This means that as the number of miners grow, the number of verification grows proportionally

How Sharding can help Scaling

For example, if only k nodes out of the n nodes verify the same set of transactions, then another set of k nodes can verify other transactions in parallel, hence saving time



The more the system grows, the more CPU resources we bring to the system and the faster the system verifies all transactions. This is called *verification sharding*.

Red Belly Blockchain

It is a community blockchain [VG18]

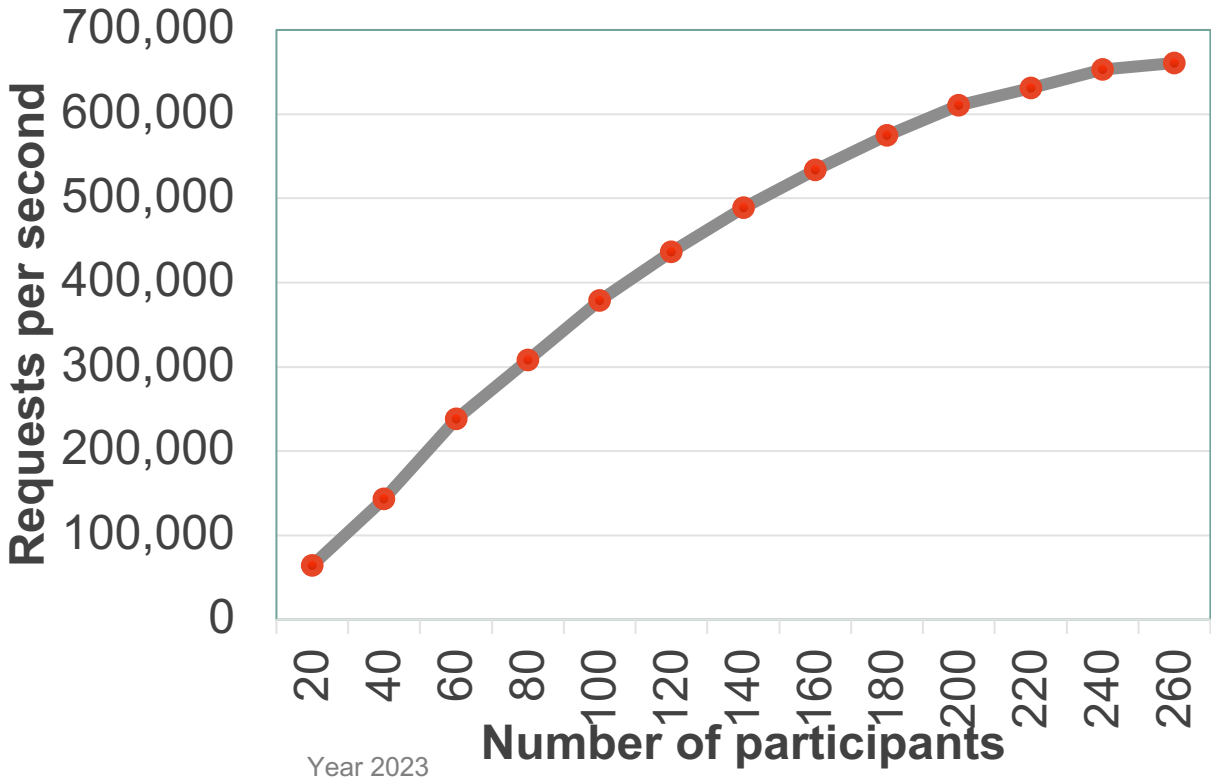
- It differs from a consortium blockchain by not pre-determining the participants but by having a set of participants that changes at runtime.
- It differs from a public blockchain by not incentivizing all participants to mine the same block

As more nodes join the set of participants, the performance increases.

Red Belly Blockchain

Performance scales with the amount of participating resources [Gra17]

Amazon EC2 instances
One availability zone in US
System size from 4 to 260
18 HT cores per node
60 GB memory
2 Gbps
Verification of ECDSA sig.
t = 6 failures
TCP+SSL

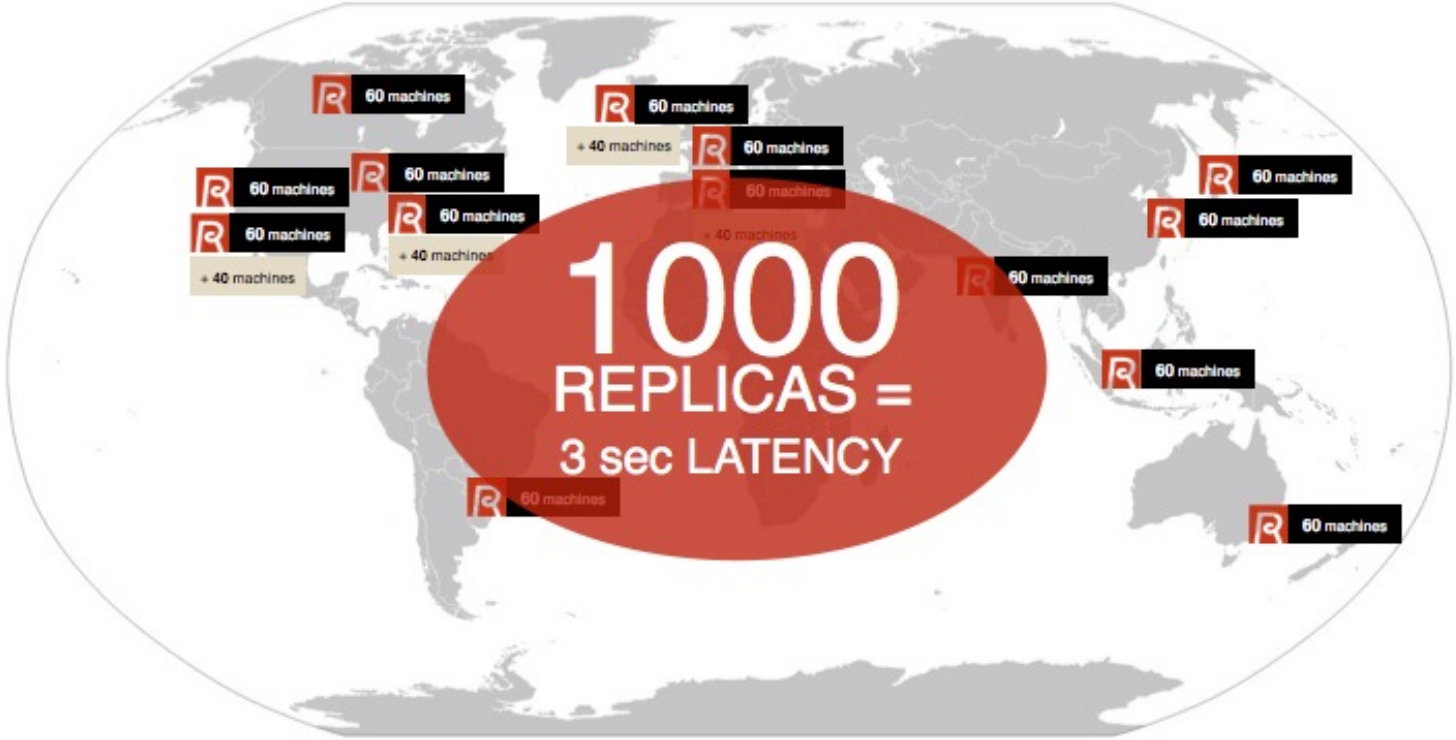


Red Belly Blockchain

Scalable to 1000 machines from 11 countries in 4 continents

Red Belly Blockchain

Fast: transactions executed within 3 seconds



Conclusion

The Red Belly Blockchain is secure and scalable.

It solves the Set Byzantine Consensus problem using the Democratic BFT algorithm.

Because it does not use any leader, it leverages the bandwidth

It also shards the verification of signatures so that every participant does not need to verify every transaction.

Theorem

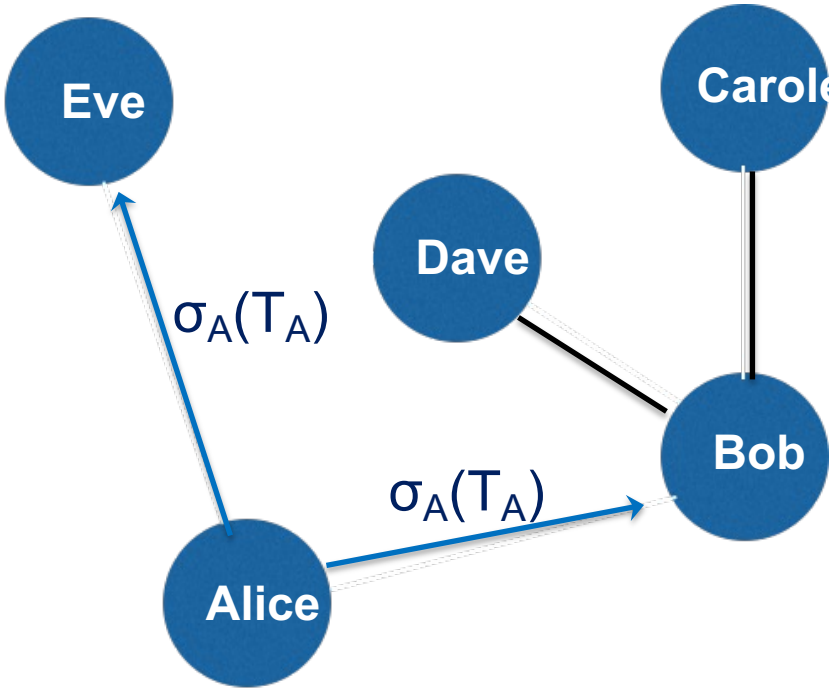
There is no solution to the Byzantine consensus if f=1 and n=3.

Proof.

- Consider execution E1 where p1 and p3 are correct and p2 Byzantine. Process p1 proposes 1 to p2 and p3, and p3 reports back to p2 but p3 says that p1 proposed 0. As p1 and p3 are correct, it follows that we need to decide 1 (due to validity).
- Consider a similar execution E2 but where p1 is Byzantine and sends different values to p2 and p3 that are correct. We need to decide 1 (due to agreement).
- Consider E3 where p1 sends 0 and is correct while p3 reports 1 back to p2. We need to decide 0 (due to validity).
- At the end of the execution, p2 cannot distinguish E3 from E2 yet it should decide differently.

Smart Redbelly Blockchain

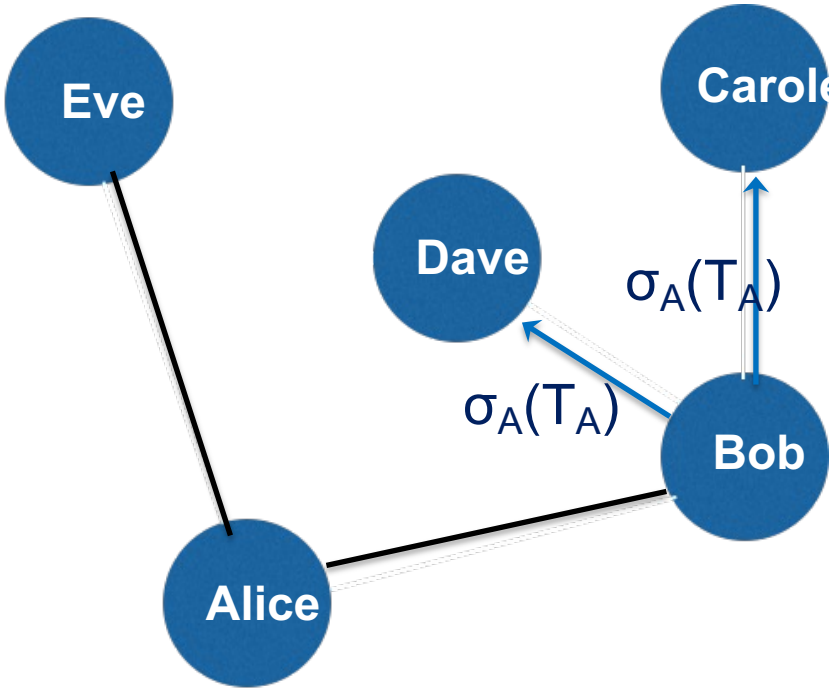
Transactions are validated twice



genesis

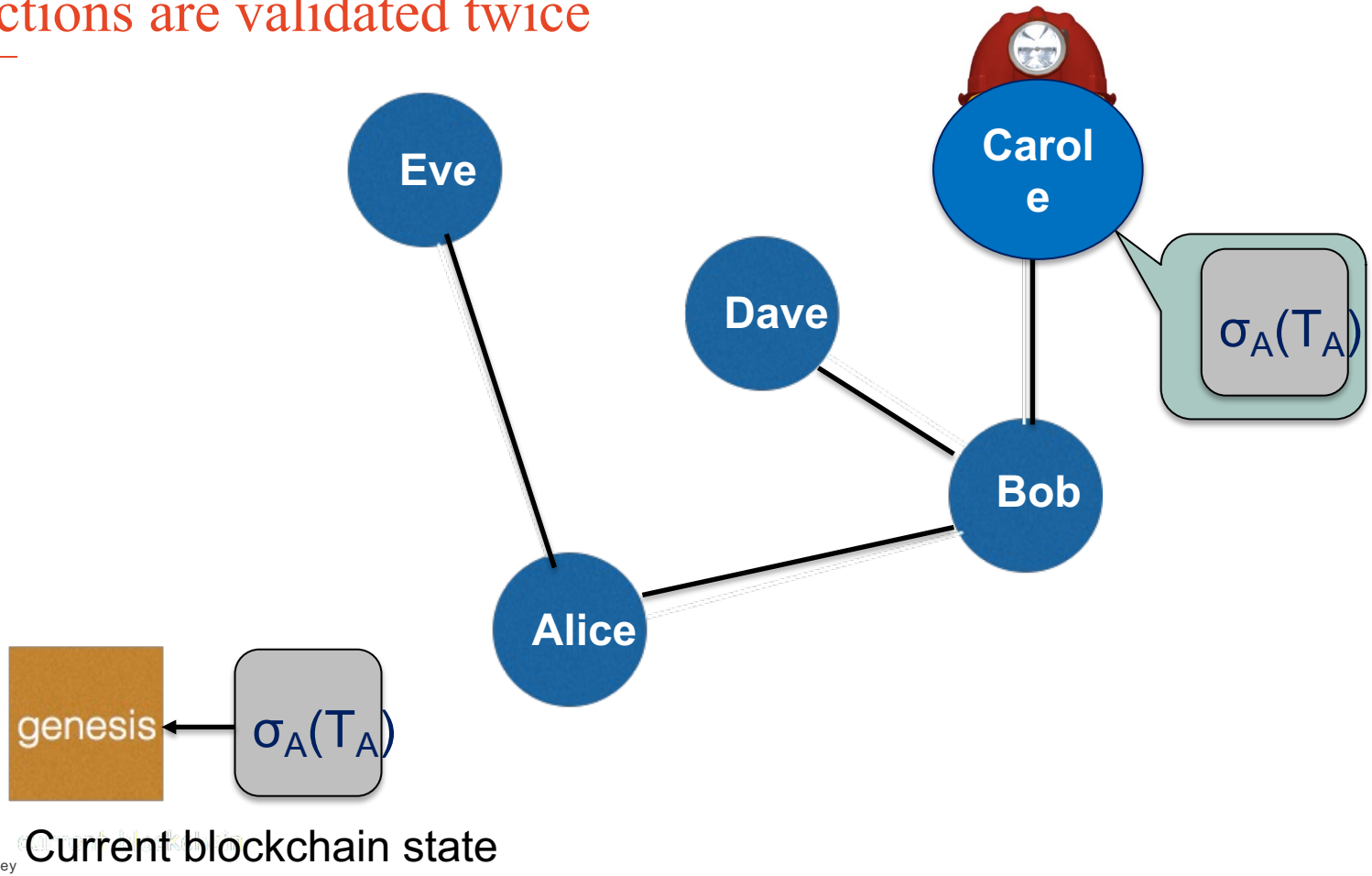
Current blockchain state

Transactions are validated twice



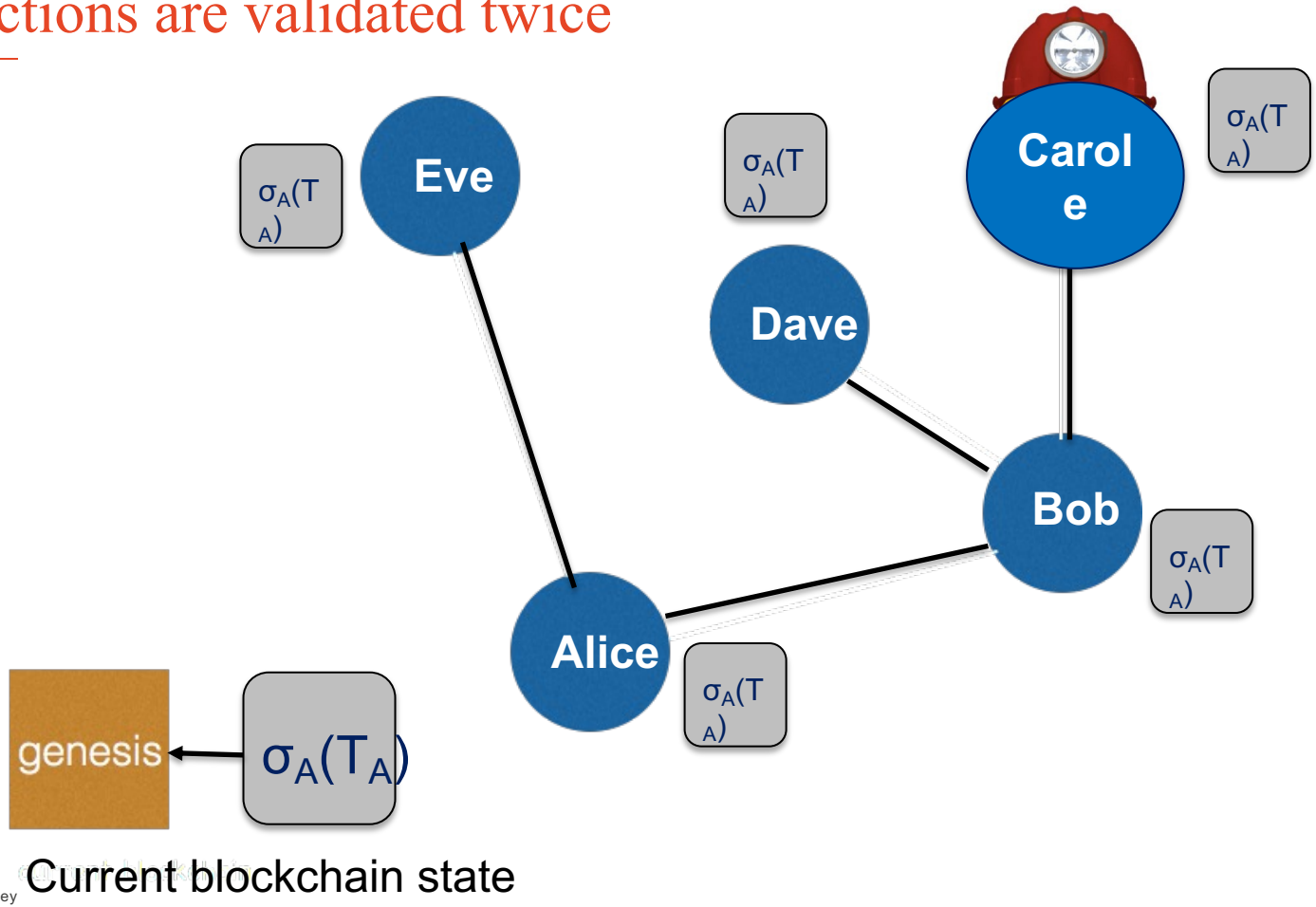
Current blockchain state

Transactions are validated twice

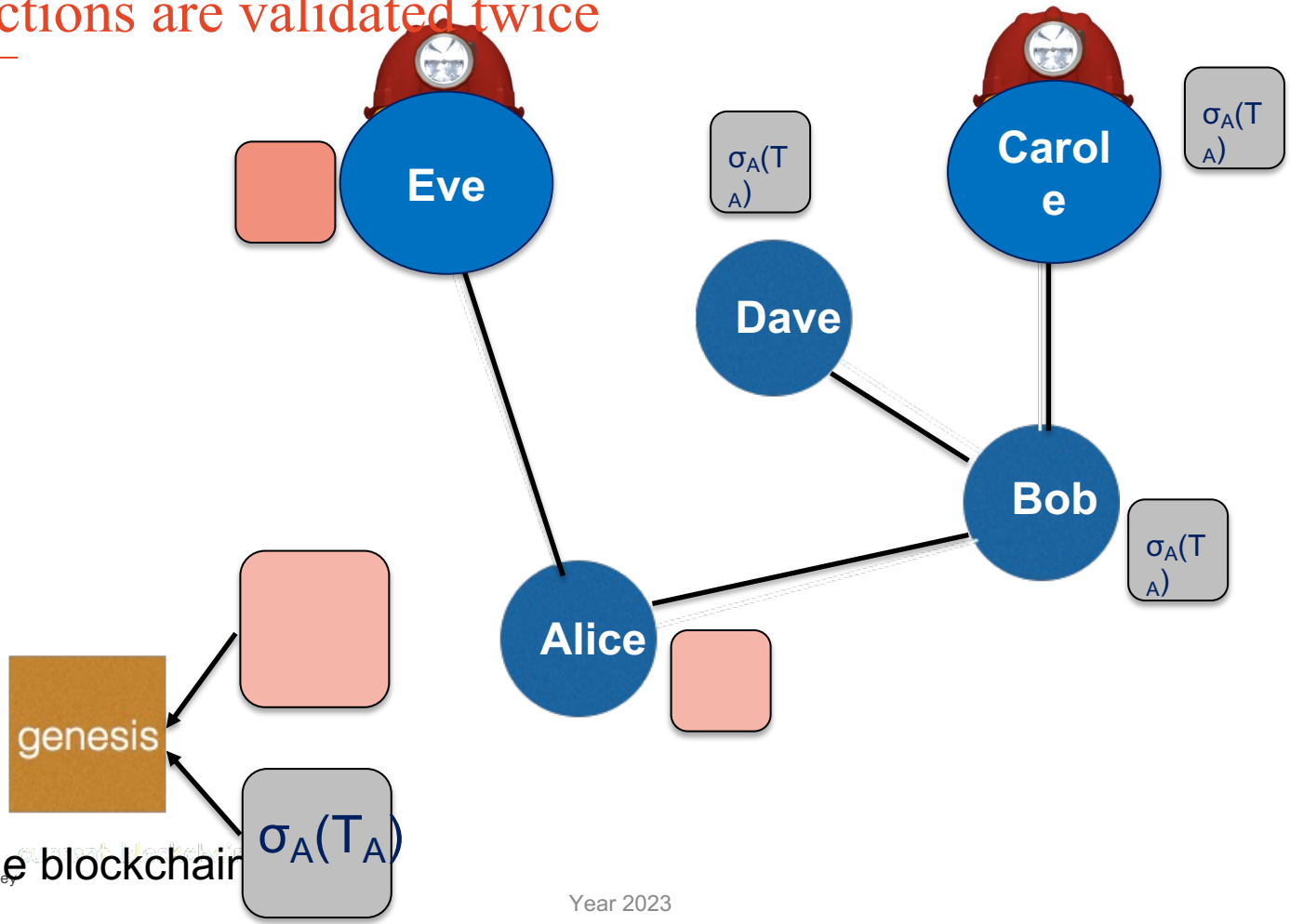


Current blockchain state

Transactions are validated twice

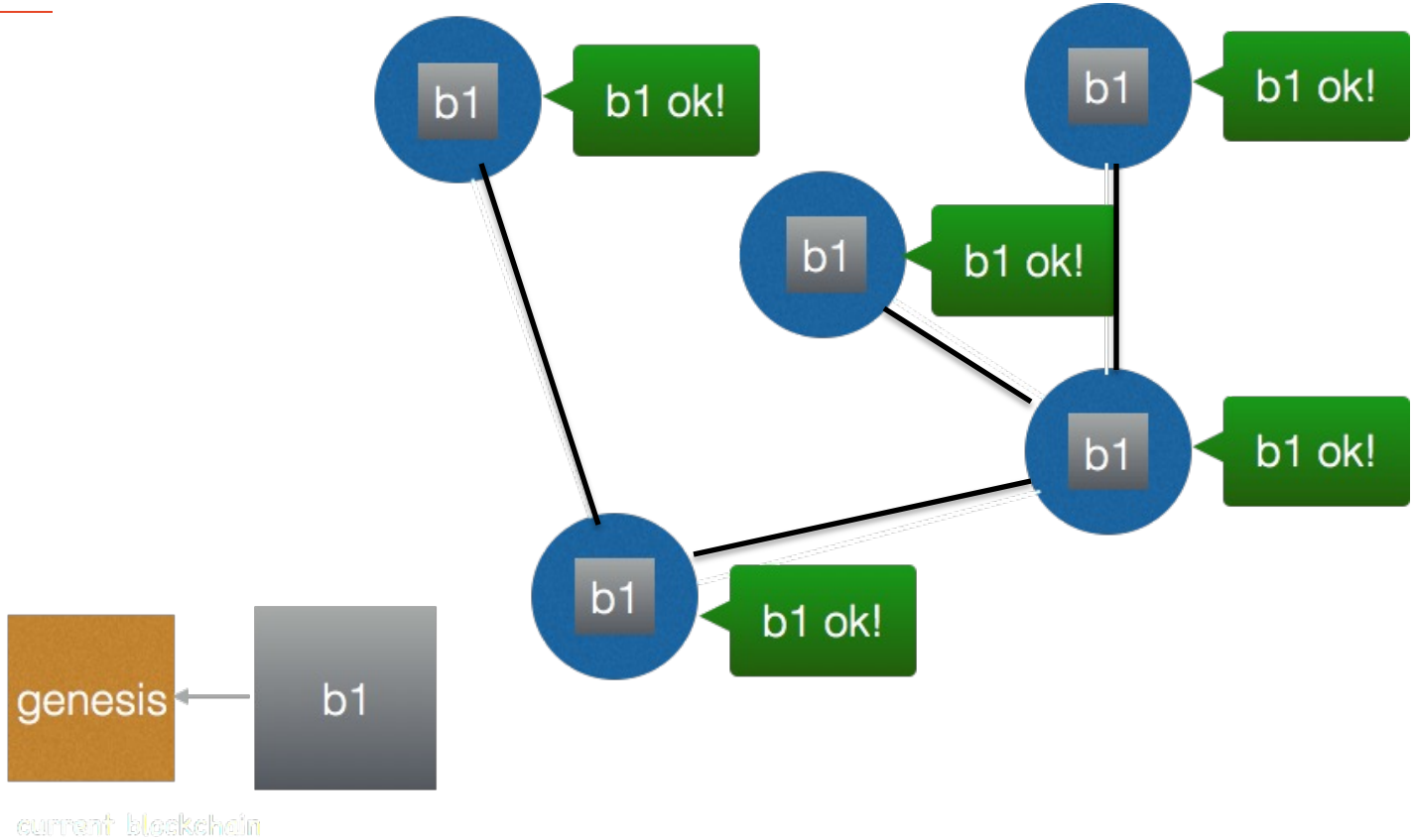


Transactions are validated twice

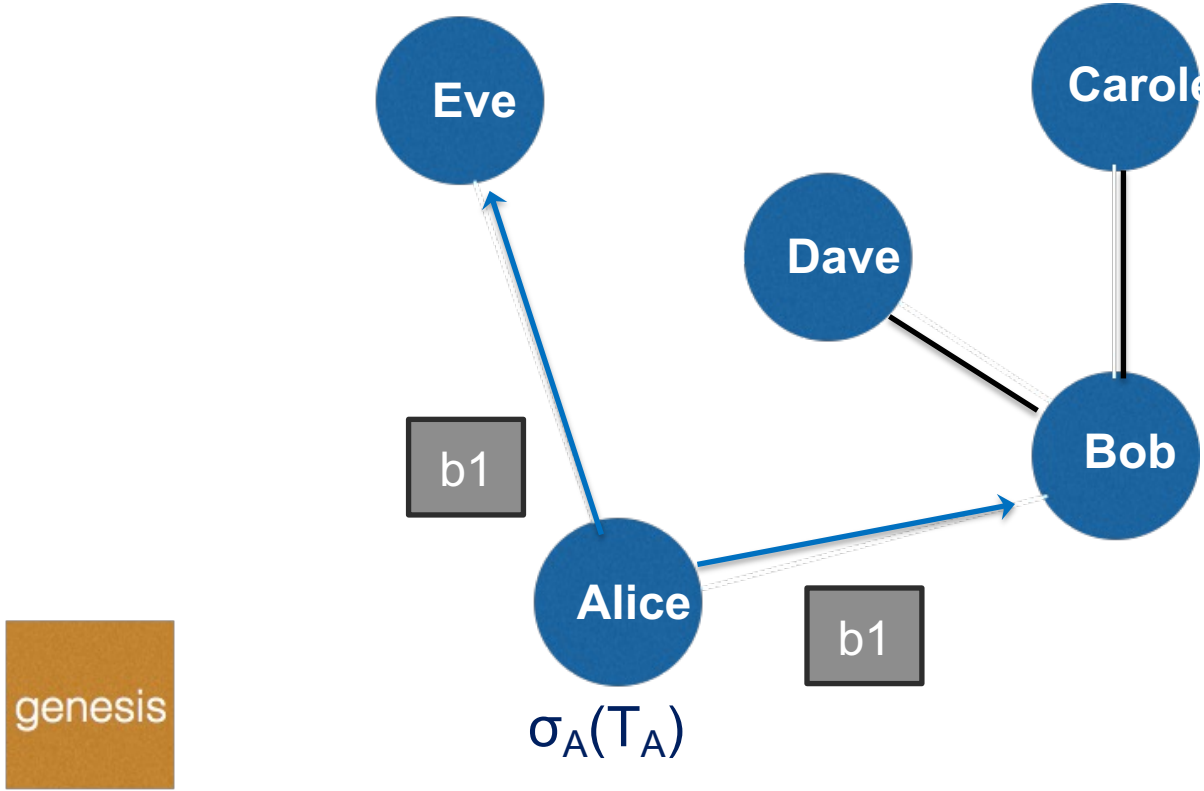


The blockchain

Transactions are validated twice

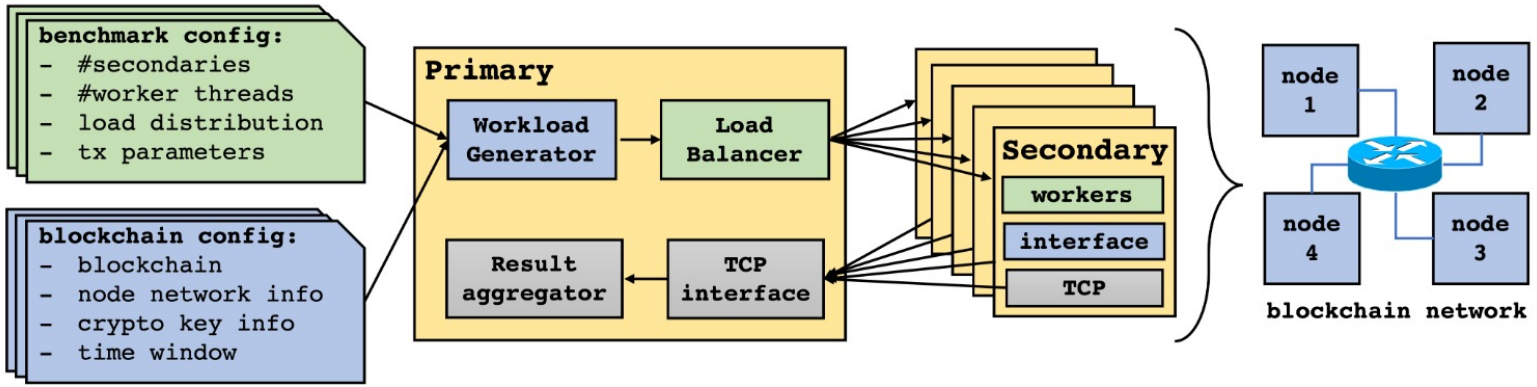


Smart Redbelly Blockchain validates less



Current blockchain state

Diablo benchmarking framework



Diablo benchmarking framework

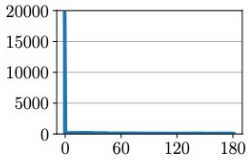
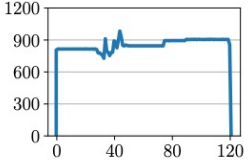
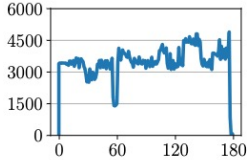
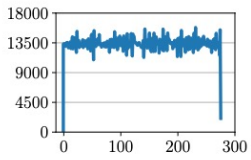
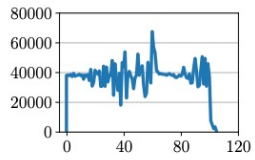
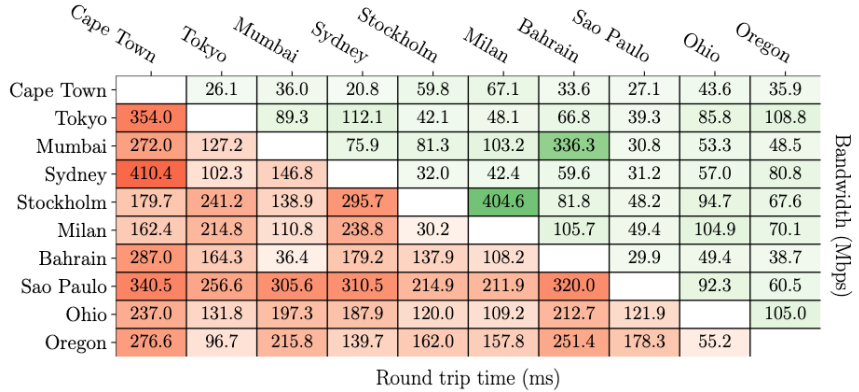
DApp	Exchange	Mobility service	Web service	Gaming	Video sharing
Workload					
Source trace	NASDAQ	Uber	FIFA	Dota 2	YouTube
Characteristics	Burst	Compute intensive	Contended	High sending rate	Very high sending rate

Table 2. Decentralized applications (DApps) used as DIABLO benchmarks and their associated workload based on real traces. Each graph shows the number of submitted transactions (y-axis) per second (x-axis).

Diablo benchmarking framework

Configuration	Blockchain nodes			Regions
	number	#vCPUs	memory	
`datacenter`	10	36	72 GiB	Ohio
`testnet`	10	4	8 GiB	Ohio
`devnet`	10	4	8 GiB	all
`community`	200	4	8 GiB	all
`consortium`	200	8	16 GiB	all



Diablo benchmarking framework

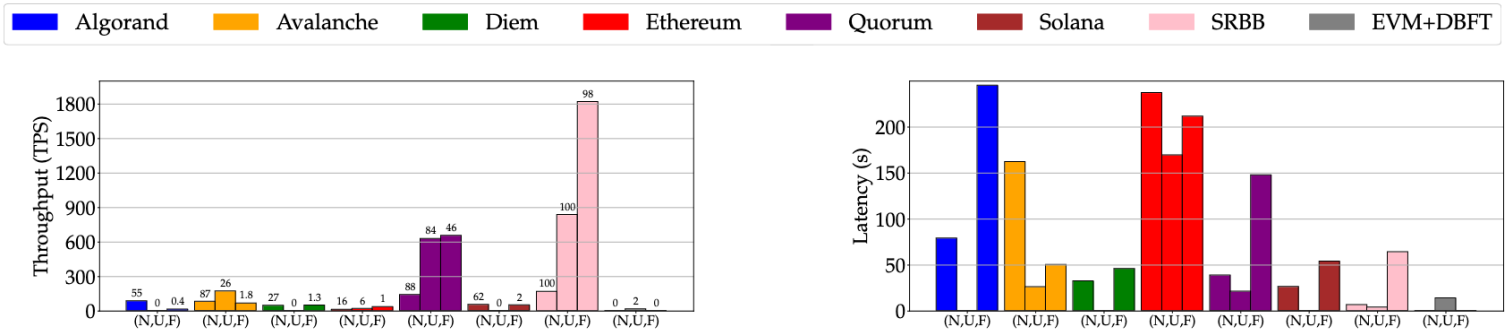


Fig. 2: Throughput (y-axis) and commit percentage (top of the bar) for NASDAQ, Uber and FIFA workloads (i.e., (N,U,F) is NASDAQ, Uber and FIFA)

Fig. 3: Latency (y-axis) for NASDAQ, Uber and FIFA workloads (i.e., (N,U,F) is NASDAQ, Uber and FIFA)

VINCENT GRAMOLI A/Prof.
THE UNIVERSITY OF SYDNEY
Rm 417, Computer Science Building | The University of Sydney | NSW | 2006
+61 2 9036 9270

https://gramoli.github.io

https://twitter.com/VincentGramoli

https://www.linkedin.com/in/vincent-gramoli-206148a/

https://scholar.google.com.au/citations?user=Fv7ZfN8AAAAJ


THE UNIVERSITY OF
SYDNEY

Backup

A guide to using our University PowerPoint template

Please read the following 3 slides, prior to building your PowerPoint presentation.

Within this presentation there are both sample presentation slides AND a full range of templates built into the master slides for your use. There is a range of traditional bullet pointed slides, title slides and chapter break slide templates, and also graphic slides to help you to make your presentation less text heavy and more visually engaging.

It is recommended that upon designing your PowerPoint presentation you:

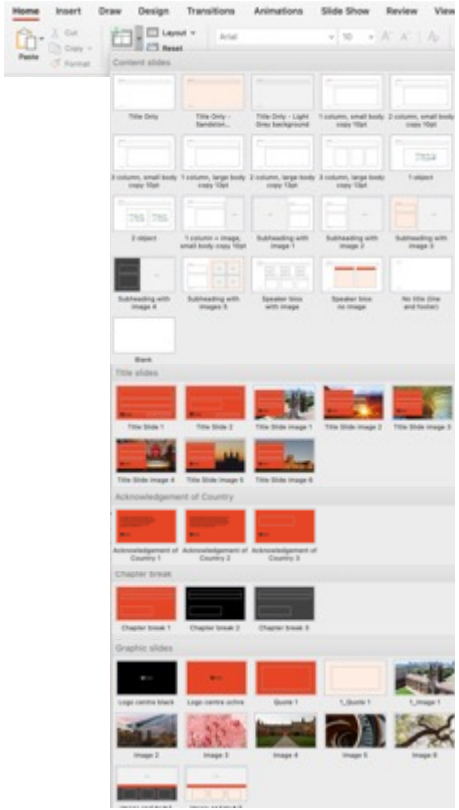
- Start by writing out your talking points.
- Utilise presenter notes. It's important not to read off the slide and speak directly to your audience throughout your PowerPoint presentation.
- Minimise the amount of text on each slide, and use images, icons and simple data/statistics to engage your audience.
- Keep your design consistent throughout. Read the notes on the following 2 slides detailing colour and font style.
- Delete the slides from the presentation that you don't need, and add slides that you do need by going to: Home > New Slide > Drop down arrow

Design and layout

In the following presentation we have included examples of presentation slides.

There are many more slide templates available. These can be viewed and added to your presentation by going to:
Home > New Slide > Drop down arrow

Templates have been grouped into categories:
- Title slides
- Acknowledgement of Country
- Chapter break slides
- Graphic slides
- Content slides



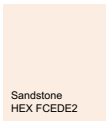
Slide transitions and animation

Our slide transition style is to keep things simple. Our preference is no transition style (ie ‘None’) as transitions can distract your audience from you and your content. Simple animation (ie ‘Appear’) can add value when used to build the story telling of your presentation, ie Step 1, Step 2 etc.

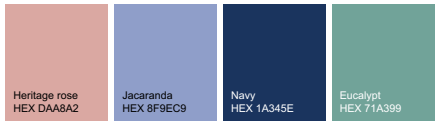
Colours

Primary colour palette

Secondary colour palette



Tertiary colour palette



These colours are preloaded into this PowerPoint colour palette.

Tertiary colours are only for use in graphs and tables. Only use the Tertiary colour palette after the Primary and Secondary colours have already been used on the same slide.

You can use tones of the colour palette.

Graphs and table

Our style is to keep graphs and tables simple with the use of the solid colours available in the colour palette, or mono tones of the Ochre colour. Do not use shading or patterns.

Fonts

The University primary fonts for use in PowerPoint are:

Arial Regular

Plus a dash of Times Italic

All body copy is Charcoal colour (not black).
Do not use any other fonts.

Bullet points

Our bullet point style is to use a ‘dash’ as a bullet point, rather than the tradition solid circle. This can be change by:

- Format > Bullet and numbering > Customise > Add in ‘-’ > Press ‘OK.

Logo

If you need to use the University logo, please copy the appropriate version from here:

Standard

THE UNIVERSITY OF SYDNEY

Reversed

THE UNIVERSITY OF SYDNEY

USER NOTE:
There are more Title Slide options available to add to your presentation by going to:
Home > New Slide > Drop down arrow

Heading sits here
1-2 lines of copy 28pt

Presentation subheading can go over 2 lines 28pt

Presented by 14pt

THE UNIVERSITY OF SYDNEY

Heading sits here 28pt 1-2 lines

Intro text 14pt, to sit here.
2–3 lines of copy. Am quo’xnkaln ium aute fuidestel eces, Cas cus aus opubli.

sydney.edu.au/xxxxx

THE UNIVERSITY OF SYDNEY

USER NOTE:

To add your own image to this Title Slide

1. Select the sky background image
2. Right click (or Control click for Mac users)
3. Select ‘Change image’
4. Navigate to the picture you want, select it, and then select Insert

There are more Title Slide image options available to add to your presentation by going to:

Home > New Slide > Drop down arrow

USER NOTE:
It is mandatory to include an ‘Acknowledgement of Country’ in your presentation. Please include this slide in your presentation.

We recognise and pay respect to the Elders and communities – past, present, and emerging – of the lands that the University of Sydney's campuses stand on. For thousands of years they have shared and exchanged knowledges across innumerable generations for the benefit of all.

THE UNIVERSITY OF SYDNEY

USER NOTE:
It is optional to include
this additional slide.

The University of Sydney's Camperdown Campus sits on the lands of the Gadigal people with campuses, teaching and research facilities on the lands of the Gamaraygal, Dharug, Wangal, Darkinyung, Burramadagal, Dharawal, Gandangara, Gamilaraay, Barkindji, Bundjalung, Wiradjuri, Ngunawal, Gureng Gureng, and Gagadju peoples.

THE UNIVERSITY OF SYDNEY

USER NOTE:
Alternatively you can write your own ‘Acknowledgement of Country’, specific to the land you are presenting from.

THE UNIVERSITY OF SYDNEY

USER NOTE:
There are more Chapter Break Slides options available to add to your presentation by going to:
Home > New Slide > Drop down arrow

Chapter break headline 28pt

1-2 lines of copy

Presented by 14pt

USER NOTE:
There are more Graphic Slide image options available to add to your presentation by going to:
Home > New Slide > Drop down arrow

USER NOTE:
There are more Content Slide options available to add to your presentation by going to:
Home > New Slide > Drop down arrow

Agenda or program
Add date of presentation

- Morning session
  - Details of morning session 1
  - Details of morning session 2
  - Details of morning session 3
- Lunch break
- Afternoon session
  - Details of afternoon session 1
  - Details of afternoon session 2
  - Details of afternoon session 3
- Close

Title and 2 column text box
Small body copy

Subheading, Arial 10pt

Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acilit latemperibus idesti non entur? Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus, ut facessitae liciam faceptat re, nos nos etus ut modicia volorrovid quatemp osaperu ptatur sam, corum que evenimolupi eos dempos is pa pedisci bla dolessimus quatur, ut quia nobis aut escit ant ut.

Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acilit latemperibus idesti non entur? Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus, ut facessitae liciam faceptat re, nos nos etus ut modicia volorrovid quatemp osaperu ptatur sam, corum que evenimolupi:

Add bullet point. Eos dempos is pa pedisci bla dolessimus quatur, ut quia nobis aut escit ant ut faccull aturepe llesequo tem et mo offic totatem cum fugit quodio volor apitibusam expla quam dis aceatio samustisim quatiistio. Namuscia venes vellaces aut omnient officab orepell oriam, eaquunt.

Add bullet point. Eos dempos is pa pedisci bla dolessimus quatur, ut quia nobis aut escit ant ut faccull aturepe llesequo tem et mo offic totatem cum fugit quodio volor apitibusam expla quam dis aceatio samustisim quatiistio. Namuscia venes vellaces aut omnient officab orepell oriam, eaquunt.

Title and 3 column text box
Small body copy

Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acilit latemperibus idesti non entur? Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus, ut facessitae liciam faceptat re, nos nos etus ut modicia volorrovid quatemp osaperu ptatur sam.

Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acilit latemperibus idesti non entur? Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus,.

Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus, ut facessitae liciam faceptat re, nos nos etus ut modicia volorrovid quatemp osaperu ptatur sam, corum que evenimolupi eos dempos is pa pedisci bla dolessimus quatur, ut quia nobis aut escit ant ut.

Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acilit latemperibus idesti non entur? Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus, ut

Research implication
Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie

Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acilit latemperibus idesti non entur? Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus.

Title and 3 column text box Large body copy

Subheading large 13pt
Body copy large 13pt.
Body copy text to go here, hereImaximus explab iunt qui assit quam ut facimpori rest alique odit, solut quidi te voluptam iduscium rerem ut que ratatur simusae, solut quidi te voluptam iduscium rerem ut que ratatur simusae.

Subheading large 13pt
Body copy large 13pt.
Body copy text to go here, hereImaximus explab iunt qui assit quam ut facimpori rest alique odit, solut quidi te voluptam iduscium rerem ut que ratatur simusae, solut quidi te voluptam iduscium rerem ut que ratatur simusae.

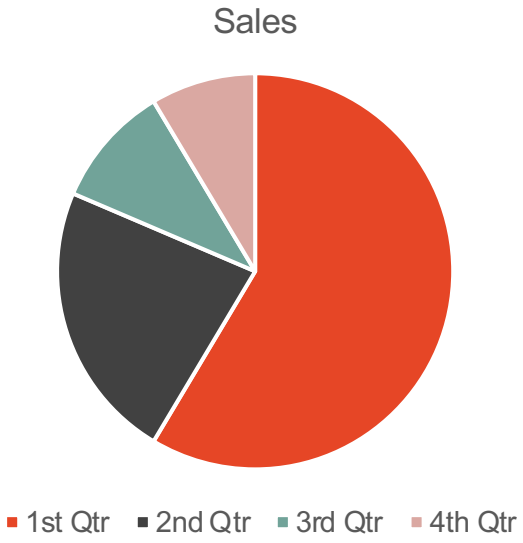
Subheading large 13pt
Body copy large 13pt.
Body copy text to go here, hereImaximus explab iunt qui assit quam ut facimpori rest alique odit, solut quidi te voluptam iduscium rerem ut que ratatur simusae, solut quidi te voluptam iduscium rerem ut que ratatur simusae.

“Insert quote here.
Unti cuptasitae velluptatur suntem as derempo rposten dignisqui omnissum eiur, earchit ab ipidici llorum, coreriae quia.”

USER NOTE:
When adding in a chart, use the default chart style ‘Style 1’.

Title and 2 objects
Full colour graph

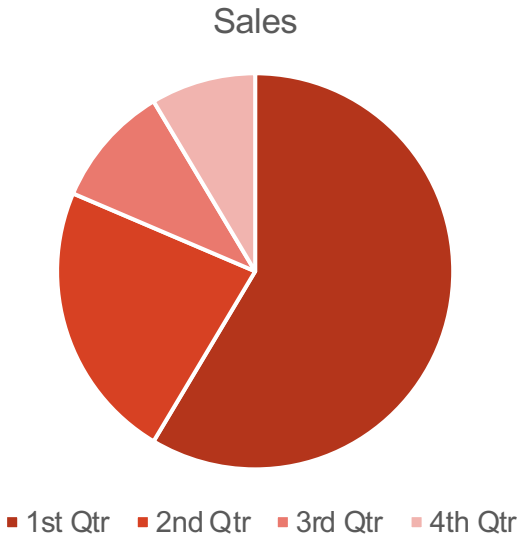
- When adding in a chart, use the default chart style ‘Style 1’.
- This will colour your graph using the University full colour themes
- Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto. As sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acili



USER NOTE:
When adding in a chart, use the default chart style ‘Style 1’. Go to ‘Change colours’ (to the left of the graph styles) and choose Monochromatic.

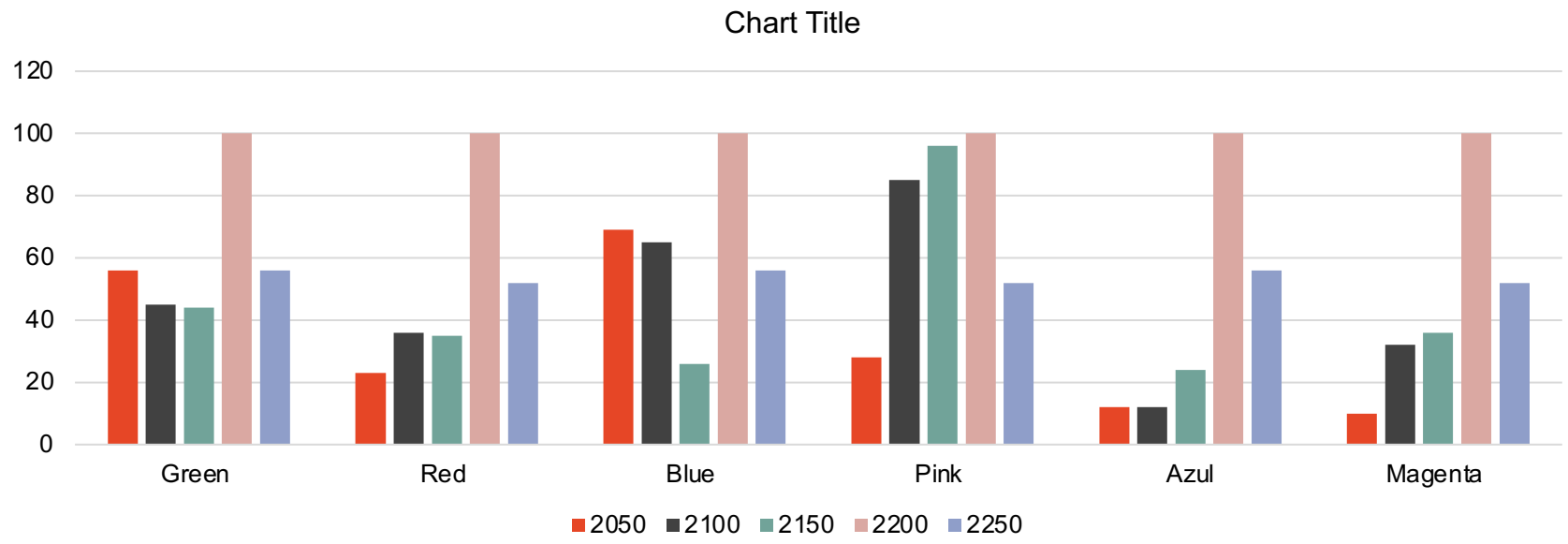
Title and 2 objects
Mono colour graph

- When adding in a chart, use the default chart style ‘Style 1’.
- This will colour your graph using the University full colour themes
- Alternatively, you can go to ‘Change colours’ (to the left of the graph styles) and choose Monochromatic.
- Small body copy 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto. As sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor atis quias acili.



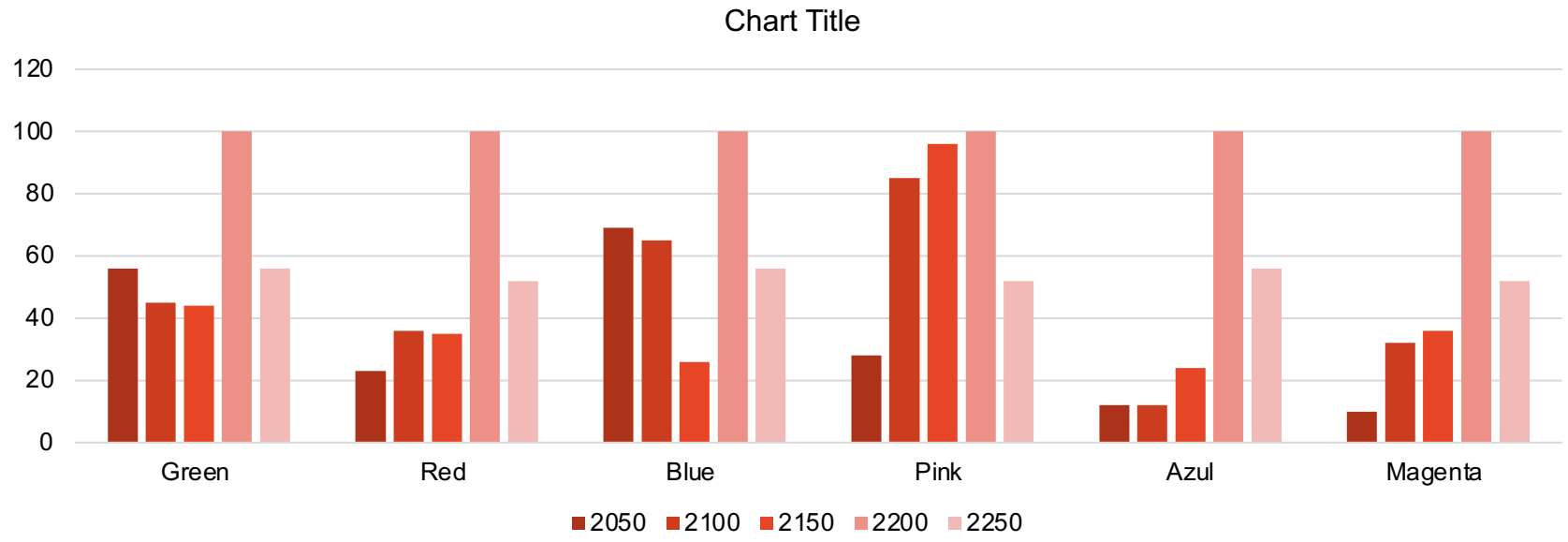
USER NOTE:
When adding in a chart, use the default chart style ‘Style 1’.

Title and 1 object
Full colour graph



USER NOTE:
When adding in a chart, use the default chart style ‘Style 1’. Go to ‘Change colours’ (to the left of the graph styles) and choose Monochromatic.

Title and 1 object
Mono colour graph



Body copy 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que num is dolumquas estrum fuga. Etfugia cusaepernam, siminto eatur sae.

Body copy 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que num is dolumquas estrum fuga. Etfugia cusaepernam, siminto eatur sae.

Body copy 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que num is dolumquas estrum fuga. Etfugia cusaepernam, siminto eatur sae.

Welcome Week (Title + 4 images)

Left to right: Welcome Week 2022, flags on Eastern Avenue, aerial view of Welcome Week activities, international students arriving at Sydney airport.

Our student-focus education is *transformational*

We will be the best Australian university for teaching and learning. Our students will attest to the transformational impact a Sydney education has on their lives, whenever and wherever they learn

Our community thrives *through diversity*

Social equity is at the heart of our past and our future. By 2032 our community will be more diverse and inclusive, helping us all to shape a positive future.

Title, text box and image
Small body copy

Body copy text to go here 10pt. Uas estrum fuga. Et fugia cusaepernam, siminto eAs sequis eatibus. Dolesti oresequae eat ressitae pra volupie ndempor

Iatis quias acilit latemperibus idesti non entur? Ritatem sincte rerum voluptincto quam qui sit, que vid qui a senderi quam, ide nobis acea sus, ut facessitae liciam faceptat re, nos nos etus ut modicia volorrovid quatemp osaperu ptatur sam, corum que evenimolupid eos dempos is pa pedisci bla dolessimus quatur, ut quia nobis aut escit ant ut faccull aturepe llesequo tem et mo offic totatem cum fugit quodio volor, apitibusam expla quam dis aceatio samustisim quatiistio. Namuscia venes vellaces aut omnient officab orepell oriam, eaquunt. Ro et aut eos atiae. Ut videlitae nonestiorrum ipsae cuptamet lantion sequae sim quis dem evel inimi, veles aut re volupta.

Inctet et, si blanturiam quassitemped quo ius.Otatem. Orestia es as eos earchil lorepudi berio bearior uptatis ma nos quis abor maionse aut est ut et, que plam in core velitio coreperat velende nis venisi alitatem.

Subheading 4
Title, text box and image
Large body copy

Body copy large 13pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que num is dolumquas estrum fuga. Et fugia cusaepernam, siminto eatur sae.

Speaker bios with image

First Last Name 10pt
Title times 10pt

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que num is dolumquas estrum fuga. Et fugia cusaepernam.

First Last Name 10pt
Title times 10pt

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que num is dolumquas estrum fuga. Et fugia cusaepernam.

First Last Name 10pt
Title times 10pt

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que num is dolumquas estrum fuga. Et fugia cusaepernam.

Icon style

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que.

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que.

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que.

Infographic data

100

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que.

95

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que.

23

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius, nume aut dis dolorro to magnitates et aut quam volorro et que.

USER NOTE:
This layout uses a custom table.
You can edit this table to keep this style.
Alternatively use the table style
‘Medium Style 1 – Accent 1” as it is most similar

Table layout option
Text can run over second line

“Large quote or key finding could go here. 13pt Times ilatic.Genihillupti is ut quatiam, sunt remquo inciis quiatistrum facea eosamet viti ommosandae. Et vid quas ventibus nosto bearum et antur? Quia sim sitae restem voluptibusa vendes debis volorest ma samusci llaccaest, untio molest, etur, veniendam architati.”

Interesting data detail, 7pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius.

Interesting data detail, 7pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius.

Interesting data detail, 7pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius.

Facebook	Twitter	Pinterest	YouTube	LinkedIn	Outcome
Text here	Text here	Text here	Date	Date	
Text here	Text here	Text here	Number views	Number views	
Text here	Text here	Text here	Text here	Text here	

Facebook	Twitter	Pinterest	YouTube	LinkedIn	Outcome
Text here	Text here	Text here	Date	Date	
Text here	Text here	Text here	Number views	Number views	
Text here	Text here	Text here	Text here	Text here	

USER NOTE:
This layout uses a custom table.
You can edit this table to keep this style.
Alternatively use the table style
‘Medium Style 1 – Accent 1” as it is most similar

Table layout option
Text can run over second line

Key highlights

Interesting data detail, 7pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius.

Interesting data detail, 7pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius.

Interesting data detail, 7pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius.

Facebook	Twitter	Pinterest	YouTube	LinkedIn	Outcome
Text here	Text here	Text here	Date	Date	
Text here	Text here	Text here	Number views	Number views	
Text here	Text here	Text here	Text here	Text here	
Text here	Text here	Text here	Text here	Text here	

“Large quote or key finding could go here. 13pt Time Ilatic. Genihillupti is ut quatiam, sunt remquo inciis quiatistrum facea eosamet viti ommosandae. Et vid quas ventibus nosto bearum et antur? Quia sim sitae restem voluptibusa vendes debis volorest ma samusci llaccaest, untio molest, etur, veniendam architati.”

USER NOTE:
This layout uses a custom table.
You can edit this table to keep this style.
Alternatively use the table style
‘Medium Style 1 – Accent 1” as it is most similar

Table layout option
Text can run over second line

Facebook	Twitter	Pinterest	YouTube	LinkedIn	Outcome
Text here	Text here	Text here	Date	Date	
Text here	Text here	Text here	Number views	Number views	
Text here	Text here	Text here	Text here	Text here	
Text here	Text here	Text here	Text here	Text here	
Text here	Text here	Text here	Text here	Text here	
Text here	Text here	Text here	Number views	Number views	
Text here	Text here	Text here	Text here	Text here	
Text here	Text here	Text here	Text here	Text here	

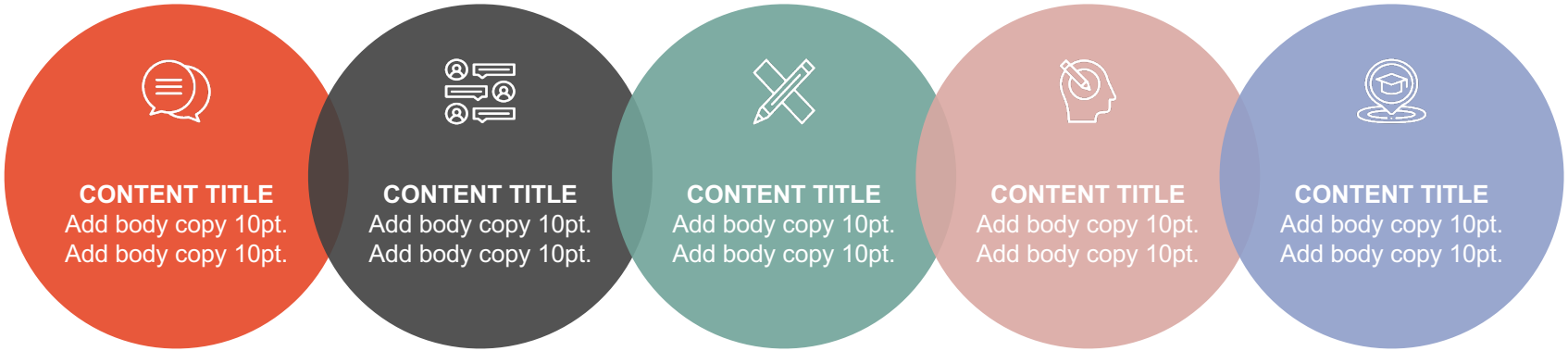
USER NOTE:
This layout uses a custom table.
You can edit this table to keep this style.

Example of timeline

Steps	January			February			March			April		
Detail 1												
Detail 2												
Detail 3												
Detail 4												
Detail 5												
Detail 6												

USER NOTE:
This layout uses a custom graphic.
You can edit this graphic to keep this style.

Flow chart style 1

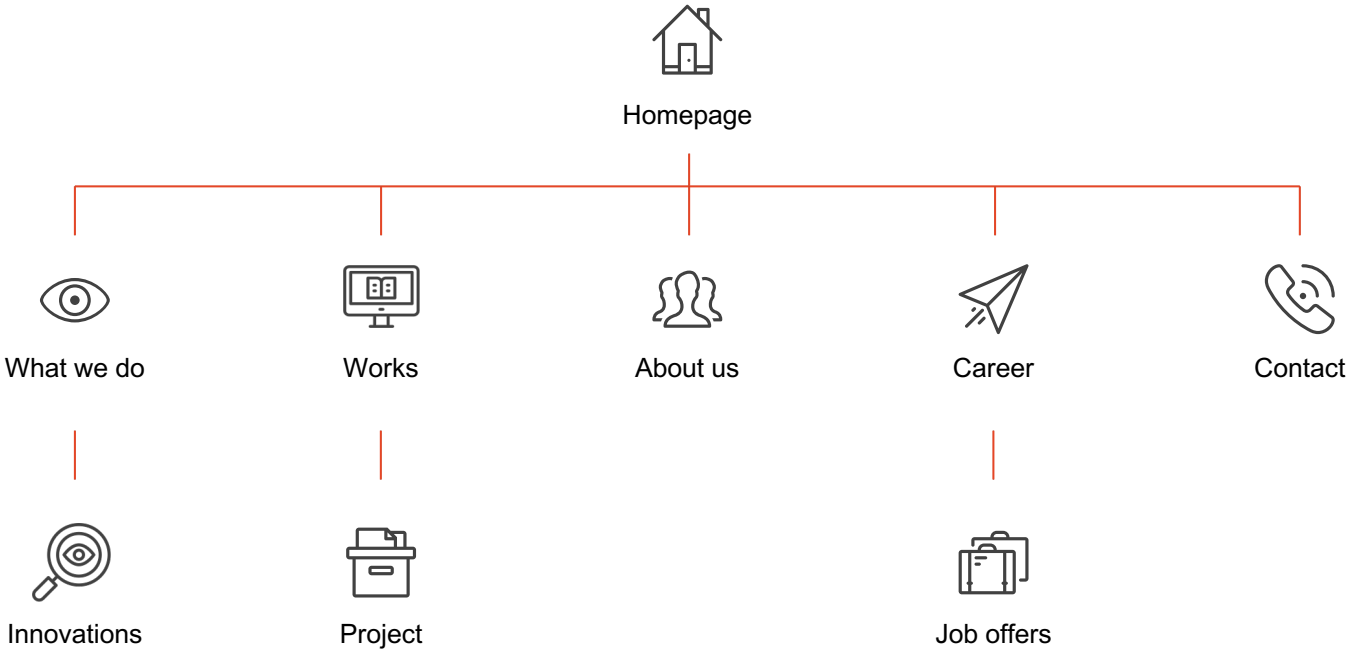


100

Body copy small 10pt. Ribusciist ut qui di re est eostorem is dit iuremolut ipis aut molupta quam, sintium quidem hicabor emperfe rfernam ius.

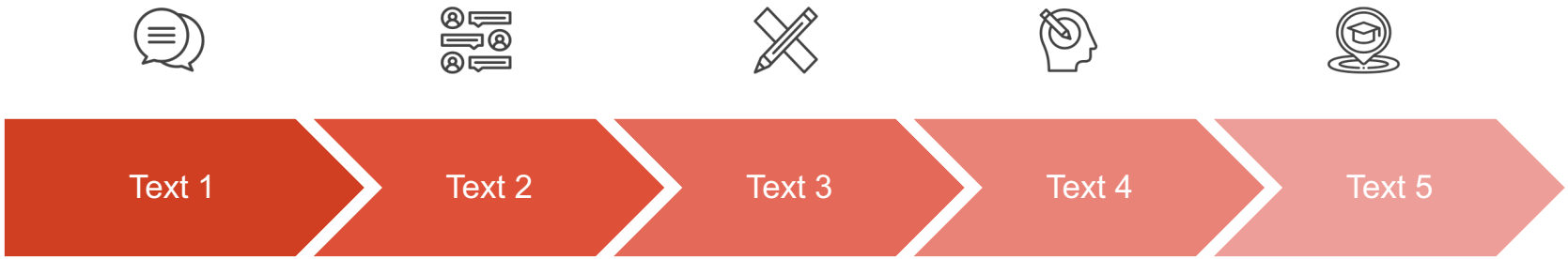
USER NOTE:
This layout uses a custom graphic.
You can edit this graphic to keep this style.

Flow chart style 2



USER NOTE:
This layout uses a custom graphic.
You can edit this graphic to keep this style.

Flow chart style 3



CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

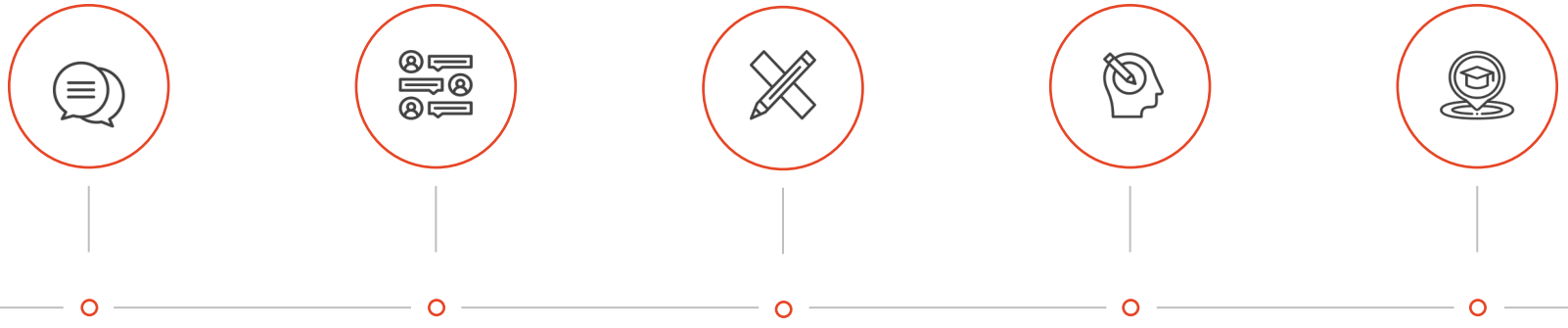
CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

USER NOTE:
This layout uses a custom graphic.
You can edit this graphic to keep this style.

Flow chart style 4



CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

CONTENT TITLE
Add body copy 10pt.
Add body copy 10pt.

USER NOTE:
Icons can be changed to white using ‘Shape fill’ for use on Ochre or Charcoal background

Icon style

Communication

Education

USER NOTE:
Icons can be changed to white using ‘Shape fill’ for use on Ochre or Charcoal background

Marketing

Research

Other icons

Our icon style is to use a thin line style of icon. Additional icons can be sourced at:
https://www.streamlinehq.com/icons/streamline-mini-line
https://thenounproject.com/pricing/#icons

USER NOTE:
Delete the social media icons below that you do not require and align the icons you need neatly

Insert team/discipline area
Name Surname Title
+61 2 1234 5678

sydney.edu.au/xxxx