Decentralization and Distributed Consensus

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
May 18, 2022

Recap Lecture 3

- *Data Structures:*
  - Hash pointers: pointers & hashing dereferenced values
  - Blockchains: hash pointer linked lists
  - Merkle Trees: hash pointer based binary trees
- *Digital Signatures*
  - Digital signature schemes
  - Generating keys, signing, verifying
  - The unforgeability game
  - Elliptic curve digital signature algorithm
- *Identities*
  - Public keys
  - Properties
- *Simple Online Cash:*
  - Centralized Coin I
  - Centralized Coin II: preventing double spending

Recap Lecture 3

- *Data Structures:*
  - Hash pointers: pointers & hashing dereferenced values
  - Blockchains: hash pointer linked lists
  - Merkle Trees: hash pointer based binary trees
- *Digital Signatures*
  - Digital signature schemes
  - Generating keys, signing, verifying
  - The unforgeability game
  - Elliptic curve digital signature algorithm
- *Identities*
  - Public keys
  - Properties
- *Simple Online Cash:*
  - Centralized Coin I
  - Centralized Coin II: preventing double spending

Recap Lecture 3

- *Data Structures:*
  - Hash pointers: pointers & hashing dereferenced values
  - Blockchains: hash pointer linked lists
  - Merkle Trees: hash pointer based binary trees
- *Digital Signatures*
  - Digital signature schemes
  - Generating keys, signing, verifying
  - The unforgeability game
  - Elliptic curve digital signature algorithm
- *Identities*
  - Public keys
  - Properties
- *Simple Online Cash:*
  - Centralized Coin I
  - Centralized Coin II: preventing double spending

Recap Lecture 3

- *Data Structures:*
  - Hash pointers: pointers & hashing dereferenced values
  - Blockchains: hash pointer linked lists
  - Merkle Trees: hash pointer based binary trees
- *Digital Signatures*
  - Digital signature schemes
  - Generating keys, signing, verifying
  - The unforgeability game
  - Elliptic curve digital signature algorithm
- *Identities*
  - Public keys
  - Properties
- *Simple Online Cash:*
  - Centralized Coin I
  - Centralized Coin II: preventing double spending

Centralization versus Decentralization

Distributed Consensus

Blockchain Consensus Issues

Blockchain Consensus Algorithm

Overview

Introduction

Overview

Introduction

UNIVERSITÄT BIELEFELD

Overview

Introduction

Overview

Introduction

- *Decentralization*
  - Introduction
  - Decentralization in Bitcoin
- *Distributed Consensus*
  - Motivation
  - Challenges
  - Impossibility Results
  - General Algorithm
- *Blockchain Consensus: Issues*
  - Concurrency
  - No Global Time
  - Node Failure
- *Blockchain Consensus: Algorithm*
  - Input Values
  - Elect, Vote and Decide
  - Disturbing Attacks

Centralization versus Decentralization

Distributed Consensus

Blockchain Consensus Issues

Blockchain Consensus Algorithm

Decentralization: Introduction

Centralization versus Decentralization

- Two competing paradigms:
  - Web pages:
    - (D) Internet (HTTP)
    - (C) ”walled gardens”: AOL and CompuServe
  - Messaging:
    - (D) Email: Simple Mail Transfer Protocol (SMTP)
    - (C) Facebook, LinkedIn
  - Social Networking:
    - (D) Efforts by hobbyists, free developers etc.
    - (C) Facebook, LinkedIn
  - Clash of paradigms also in telephony, television, radio, film
- Not all or nothing though:
  - Email servers in SMTP based messaging
  - Bitcoin services: Exchanges or wallet platforms

UNIVERSITÄT
BIELEFELD

Decentralization in Bitcoin

Emerging Questions

1. Who maintains the ledger of transactions?
2. Who has authority over which transactions are valid?
3. Who creates new bitcoins?
4. Who determines how to change rules in the system?
5. How do bitcoins achieve value outside the system?

- We will answer the first three questions

Decentralization in Bitcoin: Remarks

- Bitcoin's decentralization is a combination of
  - technical methods and
  - incentive engineering
- Bitcoin's peer-to-peer network is close to purely decentralized:
  - Everybody can install the client and run a node
  - As of May 10, 2022: 15691 nodes (estimates vary heavily!)
- *Bitcoin mining*
  - is open to everyone,
  - but requires substantial capital cost (hardware etc).
  - Undesirable because of concentration of power?
- *Software:*
  - Updates can imply rule changes
  - Many different interoperable versions
  - Majority runs reference implementation
  - Reference developers trusted by community
    ☞ have a lot of power

Distributed consensus provides answers

Centralization versus Decentralization

Distributed Consensus

Blockchain Consensus Issues

Blockchain Consensus Algorithm

Distributed Consensus: Motivation

- Imagine to have the simple coin of type II without the central authority (”Scrooge”)
- Participants (nodes) need to reach consensus on decisions taken by Scrooge
- This means consensus on:
  - Which transactions to agree on
  - How to group tranactions into blocks
  - How to order blocks
  - When to create new coins
- A *distributed consensus protocol* for Bitcoin has to figure this out

Distributed Consensus: Motivation

Definition [Distributed Consensus Protocol]:

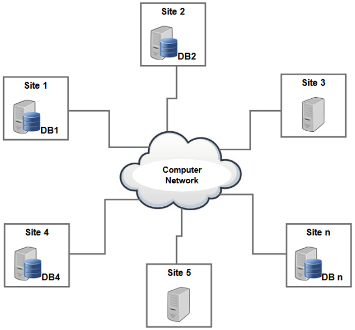
Situation:

- There are n nodes each of which suggests an input value
- The n nodes need to reach agreement about which value to select
- *Attention:* Some of the n nodes are faulty or malicious

A *distributed consensus protocol* ensures that

- Upon termination, all *honest* nodes are in agreement on the value
- The value stems from an honest node

Distributed Consensus: Popular Example



DB servers hold different data

From Ezekiel et al., 2019

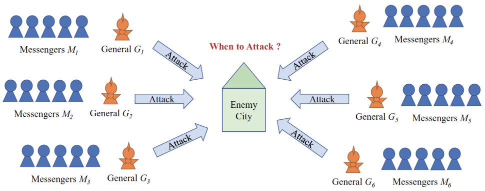
Distributed Databases

- Distributed consensus protocols have been studied extensively for *distributed databases*
- There, data is distributed across (e.g. thousands of) servers
- Nodes must be sychronized in terms of database contents

Distributed Consensus: Challenges

1. *Concurrency:*
   - Nodes act independently
   - So, multiple events occur simultaneously
2. *Lack of Global Clock:*
   - Latencies etc. prevent canonical ordering of events
   - Time and order are fundamental obstacles in distributed systems
3. *Node Failure:*
   - *Crash-Fail:* Nodes stop working
   - *Omission:* Messages are dropped (not received)
   - *Byzantine:* Nodes act arbitrarily or maliciously
4. *Message Passing:*
   - Messages are passed on asynchronously
   - Messages can be duplicated

CONSENSUS: IMPOSSIBILITY RESULTS I

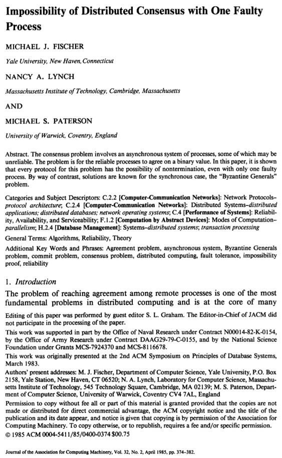


Generals need to agree on when to attack by exchanging messengers

From Kuo et al., 2019

- Byzantine army separated into divisions, each lead by a general
- Generals communicate by messenger when to launch attack
- *Goal:* Loyal generals should agree on good time without traitorous generals preventing this
- **Byzantine Generals Problem:** If one third or more generals are traitors, consensus cannot be reached

Consensus: Impossibility Results II



Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

Abstract. The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. In this paper, it is shown that every protocol for this problem has the possibility of nontermination, even with only one faulty process. By way of contrast, solutions are known for the synchronous case, the "Byzantine Generals" problem.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols–*protocol architecture*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems–*distributed applications; distributed databases; network operating systems*; C.4 [**Performance of Systems**]: Reliability, Availability, and Serviceability; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation–*parallelism*; H.2.4 [**Database Management**]: Systems–*distributed systems*; *transaction processing*

General Terms: Algorithms, Reliability, Theory

Additional Key Words and Phrases: Agreement problem, asynchronous system, Byzantine Generals problem, commit problem, consensus problem, distributed computing, fault tolerance, impossibility proof, reliability

1. *Introduction*

The problem of reaching agreement among remote processes is one of the most fundamental problems in distributed computing and is at the core of many

Editing of this paper was performed by guest editor S. L. Graham. The Editor-in-Chief of JACM did not participate in the processing of the paper.

This work was supported in part by the Office of Naval Research under Contract N00014-82-K-0154, by the Office of Army Research under Contract DAAG29-79-C-0155, and by the National Science Foundation under Grants MCS-7924370 and MCS-8116678.

This work was originally presented at the 2nd ACM Symposium on Principles of Database Systems, March 1983.

Authors' present addresses: M. J. Fischer, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520; N. A. Lynch, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139; M. S. Paterson, Department of Computer Science, University of Warwick, Coventry CV4 7AL, England

Journal of the Association for Computing Machinery, Vol. 32, No. 2, April 1985, pp. 374–382.

Fischer, Lynch, Paterson, 1985

Fischer-Lynch-Paterson Impossibility

Consensus is impossible in a system

- that is *asychronous*
- involves *n* processes
- having to agree on a binary value
- where messages can be delayed, but will be delivered eventually
- has at least one faulty node
  ☞ even only node suffices

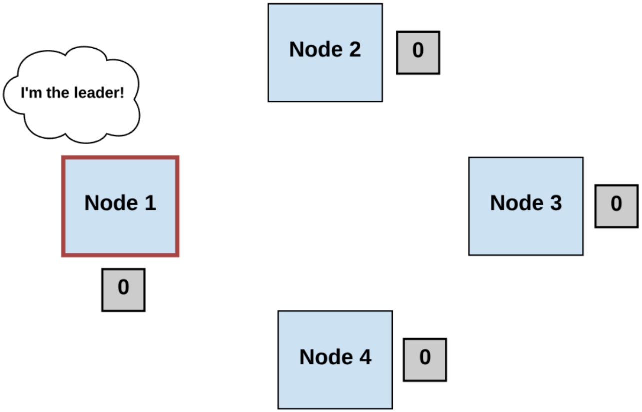
Rescue: Bitcoin is not a distributed database system

General Consensus Algorithm

General Consensus Algorithm

1. *Elect:*
   - Nodes select a *leader*
   - The leader proposes the next value
2. *Vote:*
   - Nodes validate the suggested value
3. *Decide:*
   - If validated successfully, the value is accepted
   - Otherwise, value is rejected; process is started over

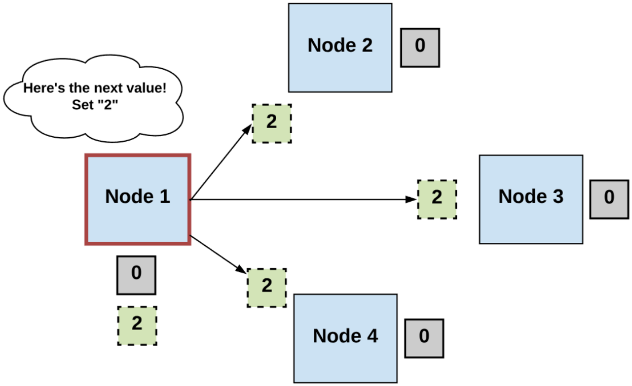
CONSENSUS ALGORITHM I



Leader is elected

From `preethikasireddy.com`

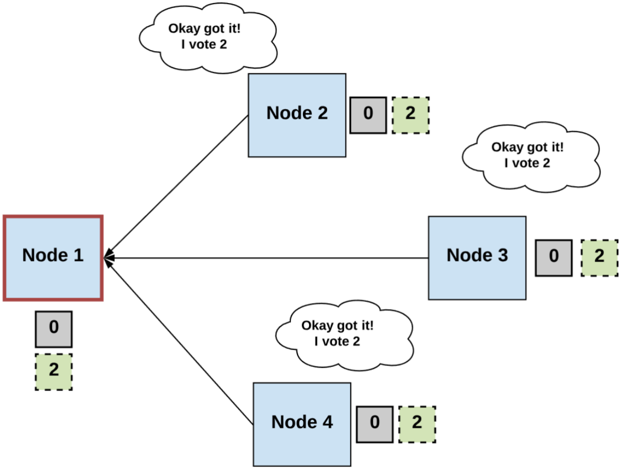
Consensus Algorithm II



Leader suggests value / block

From `preethikasireddy.com`

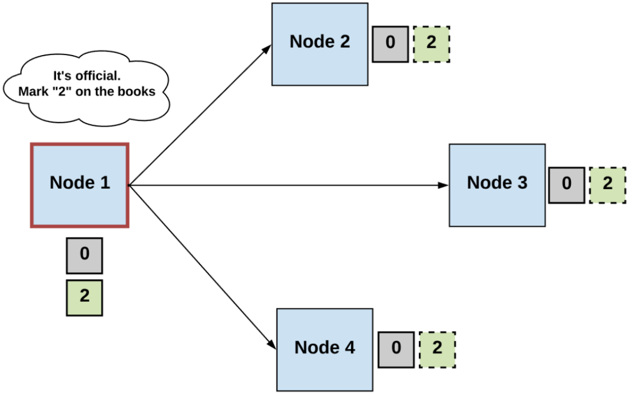
Consensus Algorithm III



Nodes approve of suggested value / block

From `preethikasireddy.com`

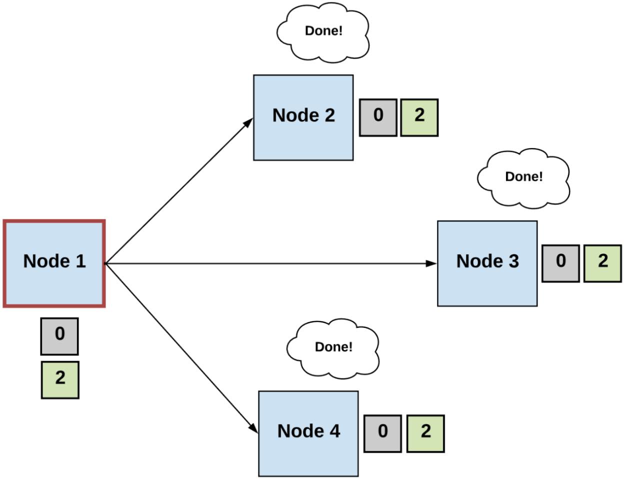
Consensus Algorithm IV



Consensus: value to be recorded / block to be appended to blockchain

From `preethikasireddy.com`

Consensus Algorithm V



Value recorded / blockchain updated

From preethikasireddy.com

Centralization versus Decentralization

Distributed Consensus

Blockchain Consensus Issues

Blockchain Consensus Algorithm

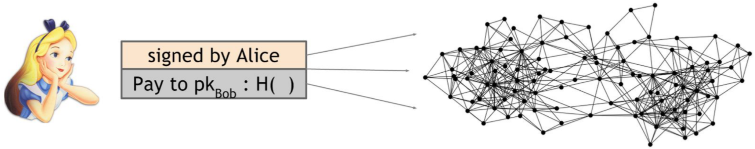
Consensus in Bitcoin

Questions

- Which issues do we have to deal with?
- What are the input values?
- How to select a leader?
- How to vote?
- How to decide?

Bitcoin Consensus Issues

Bitcoin Consensus: Concurrency



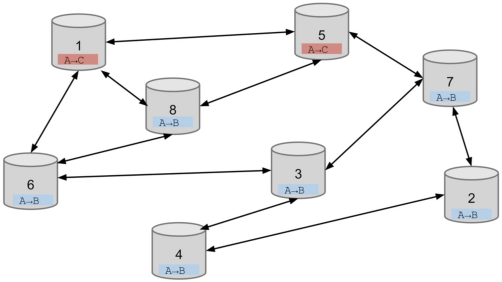
Alice communicates her intended transaction

From `bitcoinbook.cs.princeton.edu`

Broadcasting Transactions

- Alice broadcasts her transaction to the peer-to-peer network
- *Simultaneously:* Many others broadcast their transactions
  ☞ *Concurrency!*
- *Note:* Bob needs his identity pk_{Bob}, but no node to receive funds

Bitcoin Consensus: No Global Time



From bitcoinbook.cs.princeton.edu

- Node 4 broadcasts transaction $A \to B$
- Node 1 broadcasts transaction $A \to C$
  ☞ *Potential node failure!*
- Network latencies: nodes receive transactions at different times
  ☞ *No global time!*

Initial disagreement on validity of transactions

Bitcoin Consensus: Node Failure

All sorts of node failure apply:

- *Crash-Fail:* Nodes may crash, so stop working
- *Omission:* Transactions are not received
- *Byzantine:* Nodes may run attacks:
    - Nodes may try to steal coins
    - Nodes may try to double spend coins
    - Nodes may try to prevent transactions
    - ...

Centralization versus Decentralization

Distributed Consensus

Blockchain Consensus Issues

Blockchain Consensus Algorithm

Bitcoin: Input Values

General Situation

- *At any given time:*
  - All nodes have a ledger, that is a sequence of blocks of transactions
  - Various nodes have broadcast their transactions
  - Each node has collection of transactions that it approved of
  - *Question:* Which transactions to put into the next block?

Blocks = Input Values

- Node's input value is collection of self-approved transactions
  - The consensus protocol is run
  - Consensus is reached on collection of transactions to choose
- The selected collection gets the next block
  - Valid transactions not included just wait (no problem!)

Consensus in Bitcoin: First Try

Suggested Procedure

- At regular intervals (e.g. every 10 minutes), each node presents its pool of transactions as potential next block
- The consensus protocol is run:
  - *Elect:* A leader is determined; the leader proposes his block
  - *Vote:* Everyone verifies transactions in leader's block
  - *Decide:* If successfully verified, block is appended to blockchain
- *Essentially,* this is how Bitcoin works
- *However,* the issues mentioned earlier (concurrency, node failure, lack of global time etc.) require special precautions
- *Advantage:* Impossbility results appear to be theoretical obstacles

Distributed Consensus in Bitcoin

- Bitcoin consensus works in practice, but not in theory
    - Theory that explains success yet to be developed
- Bitcoin uses *incentives*
    - Bitcoin distributes rewards for generating non-malicious blocks
    - Reward is assigned only after some period of time
    - Only after some sufficient amount of time, honesty of block can be safely confirmed
    - Works in the particular setting of a currency system
- Bicoin uses *randomness*
    - Leaders are determined randomly
    - Random process such that malicious nodes less likely to be selected

Bitcoin Consensus Algorithm

Algorithm [Bitcoin Consensus]:

1. *Generating Input:*
   - New transactions are broadcast to all nodes
   - Each node collects new transactions into a block
2. *Elect:* A *random* node is selected to propose its block
3. *Vote:* Each node votes for block if found to
   - Nodes accept block if valid
     - Coins in transactions are unspent
     - Signatures successfully verified
   - Nodes vote for acceptance/rejection of block:
     - *Acceptance:* including its hash in the next block they propose
     - *Rejection:* hash of previous block included in next block
4. *Decide:* Block gets confirmed if accepted by majority of nodes
   - Initially, blockchain may branch; later, branches differ in length
   - Confirm only blocks in the longest branch

ELECT: SYBIL ATTACK I

- *Situation:* Each node is picked with equal probability
  - E.g. by running random number generator
- *Recall:* Nodes are equivalent with public identities
- *Sybil Attack:* Malicious participants can create large numbers of nodes by creating identities

  ☞ increases probability that one of their nodes is picked

Elect: Sybil Attack II

- *Sybil Attack:* Malicious participants can create large numbers of nodes by creating identities

  ☞ increases probability to pick one of their nodes

- *Solution:* Probability for a node picked proportional to value malicious nodes cannot manipulate:
  - *Proof of Work (PoW):* In proportion to compute power of a node
  - *Proof of Stake (PoS):* In proportion to amount of bitcoins owned

- *Note:* A sybil attacker requires to distribute his compute power or coins across his identities

  ☞ No increase in probability to be elected

VOTE: IMPLICIT CONSENSUS I

Issue: Real Voting

- Malicious nodes can multiply votes by creating identities
  ☞ Analogy to Sybil attack
- How to broadcast votes?
  ☞ Latencies may distort evaluation
- Who counts votes?
  ☞ Requires (temporary) authority who can manipulate process

Conclusion: Real voting is impossible

Vote: Implicit Consensus II

- *Issue:* Real voting impossible
- *Solution – Implicit Consensus:* Each node votes by including the suggested block as hash in the next block, or not
- Majority ”votes” for block:
  ☞ probability that next block builds on it is large
- Majority ”votes” against block:
  ☞ probability that block is not appended to blockchain is large

Conclusion: Works if probability of picking honest random node as leader during elections is greater than 50%

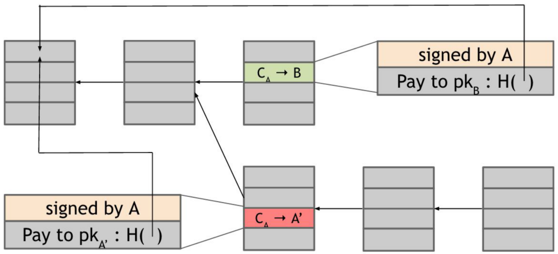
Decide: Double-Spend Attack I

- *Recall:* Initially, nodes vote for
  - extending suggested block or
  - omitting suggested block
- *Consequence:* Different votes induce branching in blockchain
- One needs to determine which branch to keep

  ☞ Remember: blockchain should be linear eventually
- *Solution:* Wait, and keep only longest branch

  ☞ This is the branch that the majority supports over the long run

DECIDE: DOUBLE-SPEND ATTACK II

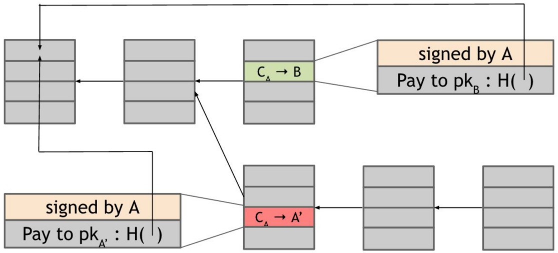


Arrows = hash pointers

From `bitcoinbook.cs.princeton.edu`

- Alice creates two transactions to spend identical coin:
  - Transaction $C_A \rightarrow B$ spends it to Bob
  - Transaction $C_A \rightarrow A'$ spends it to herself
  - A' may be an alternative public key of Alice
- No way to identify that $C_A \rightarrow A'$ is immoral

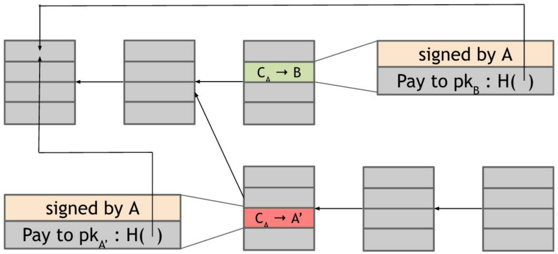
DECIDE: DOUBLE-SPEND ATTACK III



Alice (A) tries to spend same on coin on both Bob (B) and herself (A′) From `bitcoinbook.cs.princeton.edu`

- However, nodes notice the double-spending. Initially, honest nodes
  - build on the block received first
  - omit the block received later
- Ambiguity among nodes induces a branch
- *Recall:* Without branches no double spending ☞ try to get rid of it!

DECIDE: DOUBLE-SPEND ATTACK IV



Alice (A) tries to spend same on coin on both Bob (B) and herself (A′) From `bitcoinbook.cs.princeton.edu`

First Rule: Keep building on block received first

- However, latencies may confound the situation
- ☞ both branches may keep extending; double spend realizes

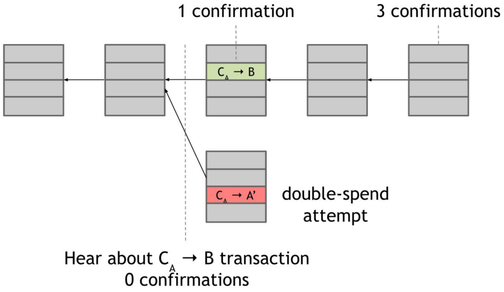
Better rule required!

UNIVERSITÄT BIELEFELD

Decide: Double-Spend Attack V

Better Rule:
Build on the block extending the longest branch

DECIDE: BLOCK CONFIRMATION



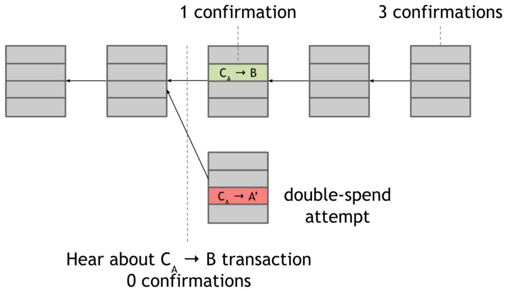
Block containing $C_A \to A'$ is an orphan block

From `bitcoinbook.cs.princeton.edu`

DEFINITION [BLOCK CONFIRMATION]: A *confirmation* of a block is a block that builds on it, including the block itself.

DEFINITION [ORPHAN BLOCK]: An *orphan block* is a block with 1 confirmation whose previous block has more than 2 confirmations

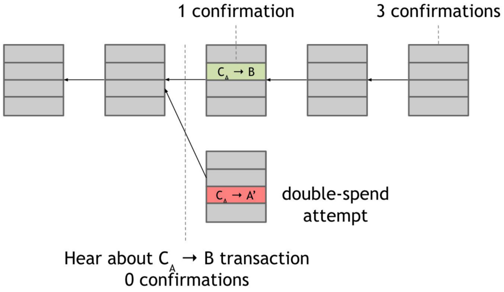
Extending Longest Block: Consequences



From `bitcoinbook.cs.princeton.edu`

- All nodes following rule: one block becomes an *orphan block*
- Removing all orphan blocks after some time prevents double spending
- Transaction to Bob in orphan block: ☞ Bob refuses service to Alice
- *Caveat:* Not all blocks may follow rule! What to implement in practice?

DECIDE: ALGORITHM



From `bitcoinbook.cs.princeton.edu`

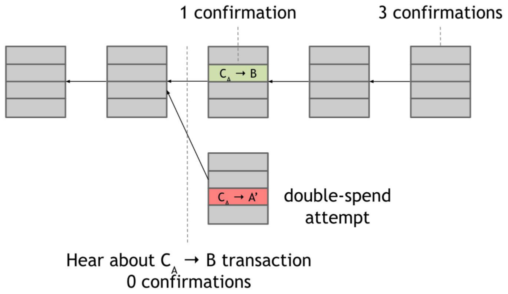
DECIDE:

1. Include blocks only when having more than *k* confirmations
2. Abandon other blocks building on earlier *for good!*

Works if majority of nodes follow rule!

UNIVERSITÄT BIELEFELD

INCENTIVIZING RULE: BLOCK REWARD



$k = 3$: only creator of block with $C_A \to B$ receives reward

From `bitcoinbook.cs.princeton.edu`

Incentive for Extending Longest Branch

- Consider paying *rewards* to block creators
- Pay only when suggested block has at least k confirmations

Motivates to extend longest branch: otherwise reward at risk!

UNIVERSITÄT BIELEFELD

Further Attacks

Stealing Bitcoins

- Stealing bitcoins from an identity one does not control?
- to forge signatures of that identity ...
- ... which in turn requires to be in possession of its secret key

Stealing bitcoins is impossible

Denial of Service

- Refusing to include particular transactions into blocks?
- If transaction is fine, honest nodes will put transaction into their blocks
- Transaction will be included in block eventually

Denial of service is impossible

MATERIALS / OUTLOOK

- See *Bitcoin and Cryptocurrency Technologies*, 2.1 – 2.3
- See `https://bitcoinbook.cs.princeton.edu/` for further resources
- See also related resources at `https://www.preethikasireddy.com`
- Next lecture: “Bitcoin: Technical Mechanics”
  - See *Bitcoin and Cryptocurrency Technologies* 3