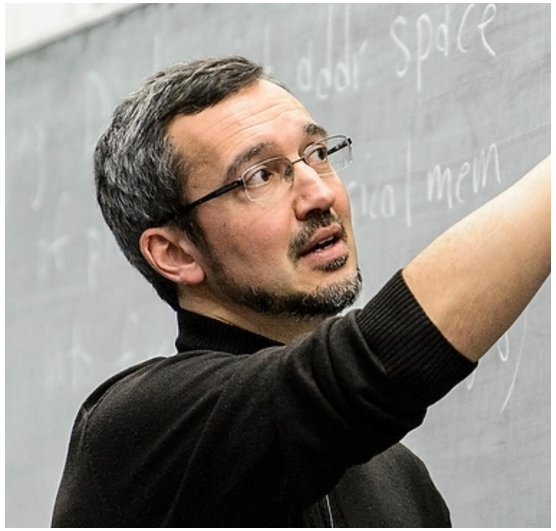
INTRODUCTION TO BLOCKCHAIN TECHNOLOGIES

Kai Mast

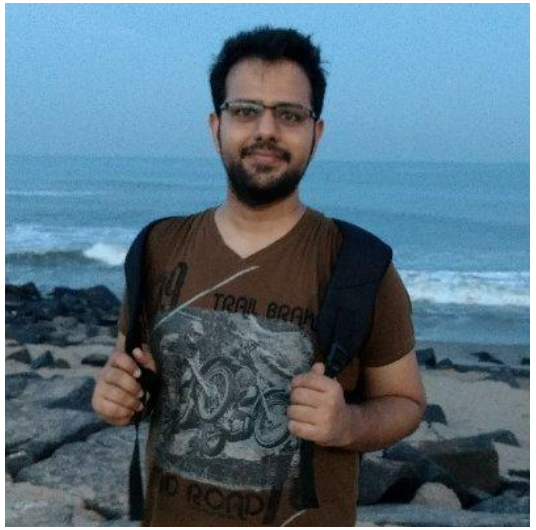
Spring 2023

COURSE STAFF

Instructor (me):
Kai Mast



Supervisory Instructor:
Remzi Arpaci-Dusseau



Teaching Assistant:
Sambhav Satija

WHO AM I?

Research Areas: Cloud Computing, Databases, Blockchains

Name Pronunciation:
Kai *(like the "Ky" in "Kyle")* Mast *(like the American English word "must")*

Position: Postdoctoral Researcher + Lecturer

LEARNING OUTCOMES

- Be able to build decentralized applications

- Understand the design of complex distributed systems
  - Learn more about concurrency and networking
  - Understand how protocols are designed
  - Be able to read whitepapers

- Be able to decide when (not to) use blockchains
  - Know the technical limitations and open problems

- Know the inner workings of blockchain nodes
  - What actually happens when smart contract code executes?

PREREQUISITES

- **No prior knowledge** of blockchains or smart contracts is assumed
- **No prior knowledge** of distributed systems is assumed
  - This class also serves as your introduction to distributed systems
- Students should have a basic understanding of computer systems, e.g.,
  - Virtualization: Processes and threads
  - Concurrency: Locks and conditions variables

LECTURES AND DISCUSSIONS

- One lecture a week ~90 minutes
- Followed by a discussion or tutorial ~60 minutes
  - Might not happen every week!
- There may be optional review sessions and discussions

"MINI" PROJECTS

- Shorter than the final project, but still significant work
- Four mini projects planned
  - Each roughly 5% of the final grade
- Mini-projects have to be done by yourself
- No formal slip day policy
  - We will give extensions on a case-by-case basis

FINAL PROJECT

Work on a group project of your choosing, for example:

- Build a decentralized app of your choosing
- Modifying the code of an existing blockchain system
- Investigate a complex decentralized app
- Review a topic that we did not cover in class

Tentative timeline:

- *Before mid-March:* Discuss potential topics with instructor
- *Mid-March:* Project proposals due (can be short)
- *Beginning of April:* Decision on topic
- *Third week of April:* Early progress report (e.g., literature review)
- *End of semester:* Project report due

GRADE COMPOSITION

- 45% Exams
  - There will be one midterm and a final
  - Each exam is worth 22.5%
- 5% Homework Quizzes
  - Mostly to help you review course material
- 20% “Mini” Projects
- 30% Final Project

NON-GOALS

- A deep dive into the economics of blockchains
  - We will talk about transaction fees, block rewards, but not much more
- Advanced Cryptography
  - We will treat cryptography more or less like a black box
  - Class will not cover systems like ZCash
- Web design / GUI Development for Decentralized Applications
  - We will focus on the “backend”

THIS CLASS: A TOP-DOWN APPROACH

Start with a coarse-grained view and work towards a detailed understanding

Part 0 (Today!): Basic Concepts and Terminology
- Don't worry, we will revisit a lot of these over the course of the semester

Part 1: Applications
- Learn about smart contracts and how they work

Part 2: Networks
- Learn how nodes communicate with each other

Part 3: Nodes
- Learn the internal working of blockchain nodes

FEEDBACK WELCOME

This may be the first lecture that

- Focuses on the implementation of blockchains system
- Is not a paper reading class
- And is not a cryptography class

We are still developing course material

- Feedback and suggestions welcome

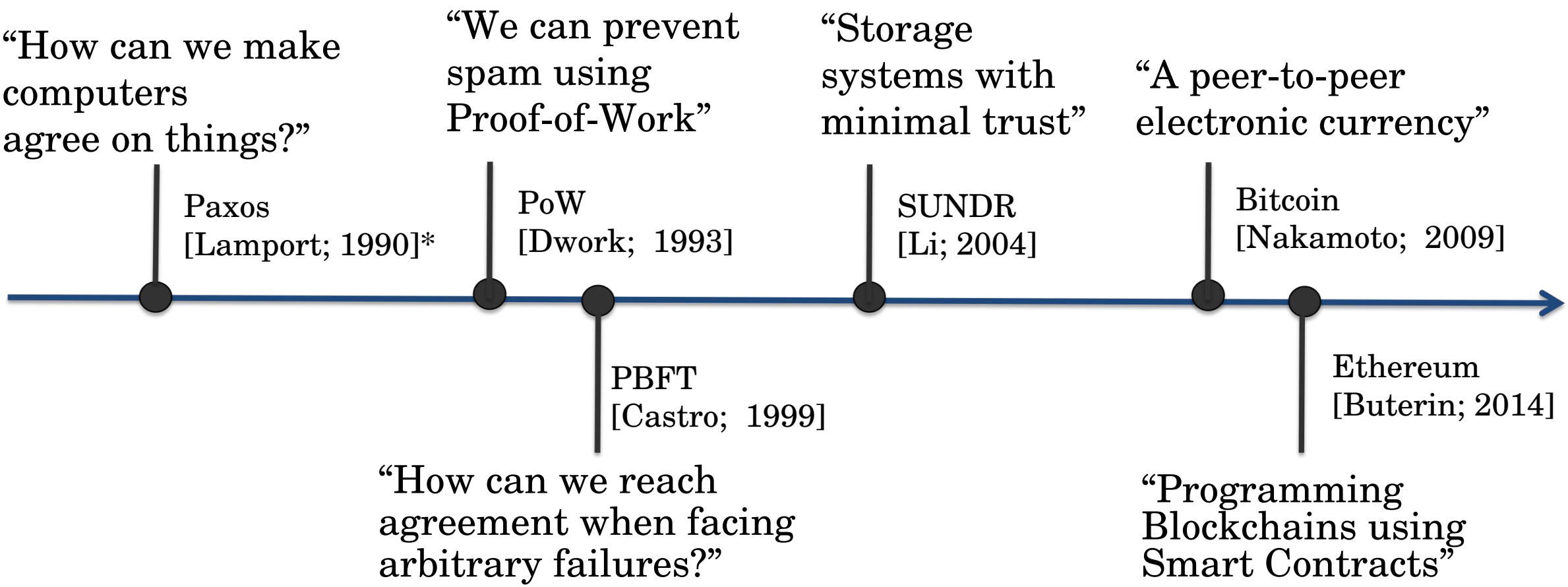
We want this to be a fun and safe learning environment

- Let us know if we can improve accessibility somehow

DISCUSSION

- What is a blockchain?
- When do we need/want a blockchain?
- Limitations of blockchain technologies?

A BRIEF HISTORY OF BLOCKCHAINS



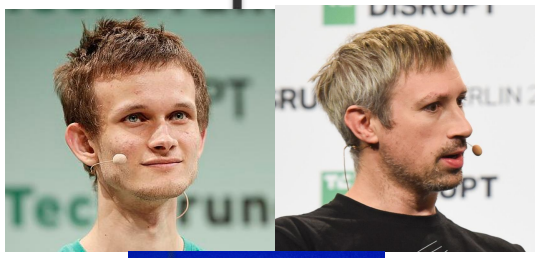
*Paxos was not actually published until 1998

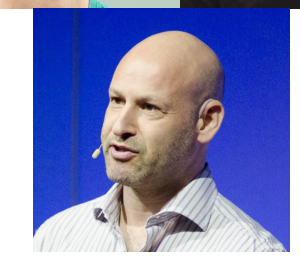
A LESS ACCURATE HISTORY OF BLOCKCHAINS

"The Before Times" (pre 2009)

Bitcoin [2009]

Ethereum [2014]

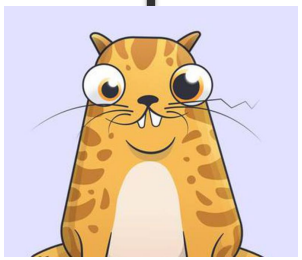




First large-scale exploit of a smart contract bug

DAO Hack [2016]

CryptoKitties [2017]

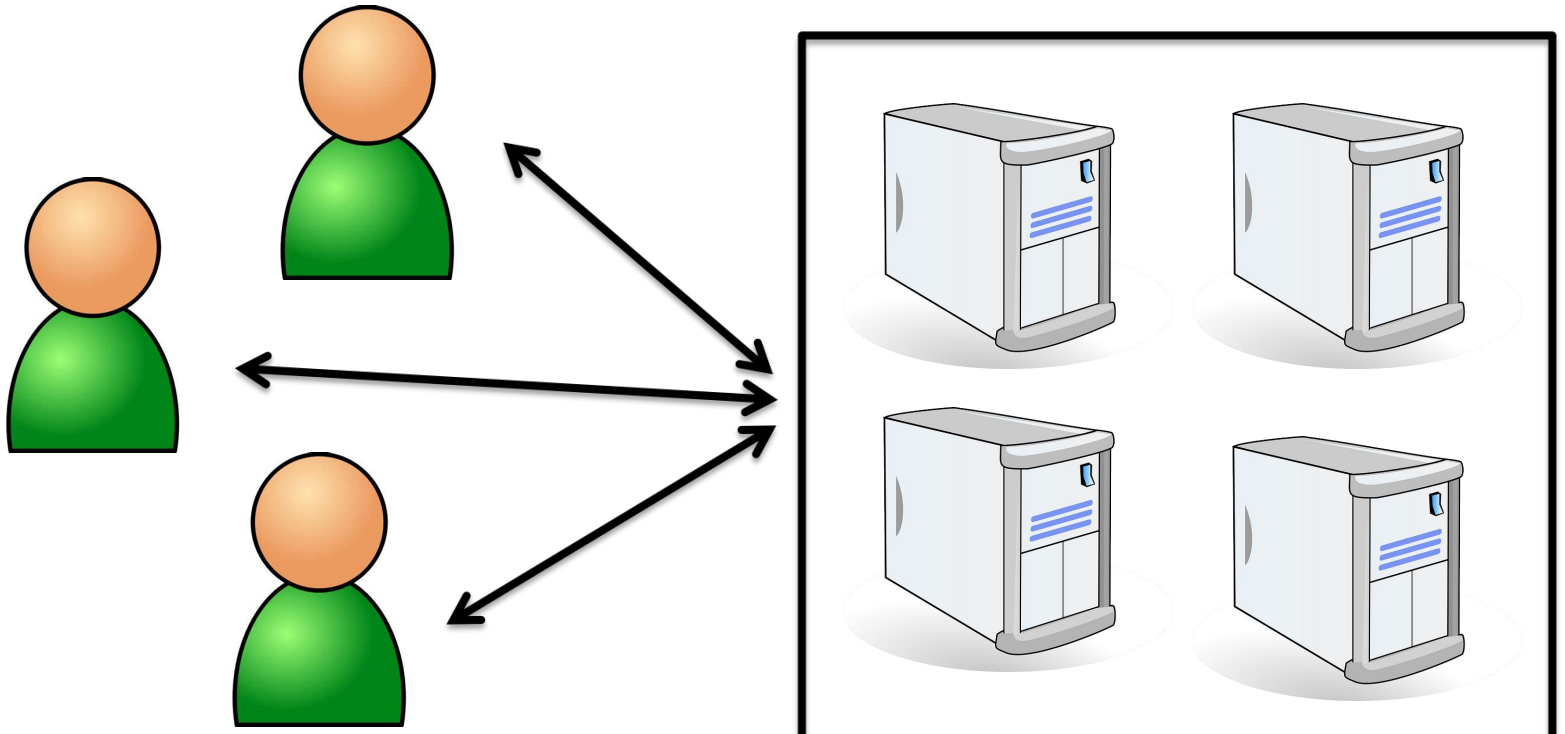


First DApp to push the Ethereum Network to its limits

Ethereum moves away from mining

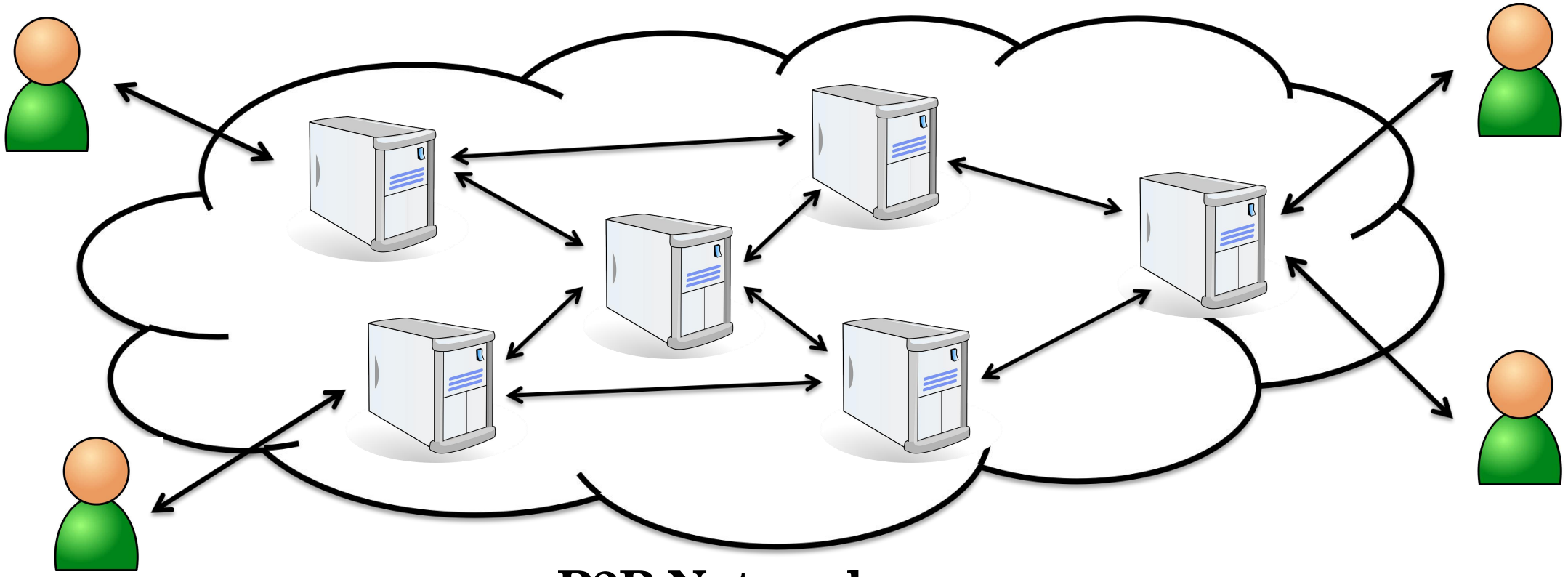
"The Merge" [2022]

THE PROBLEM: CENTRALIZED SYSTEMS



- All servers (or **nodes**) are controlled by the same entity (e.g., Google)
- All nodes trust each other

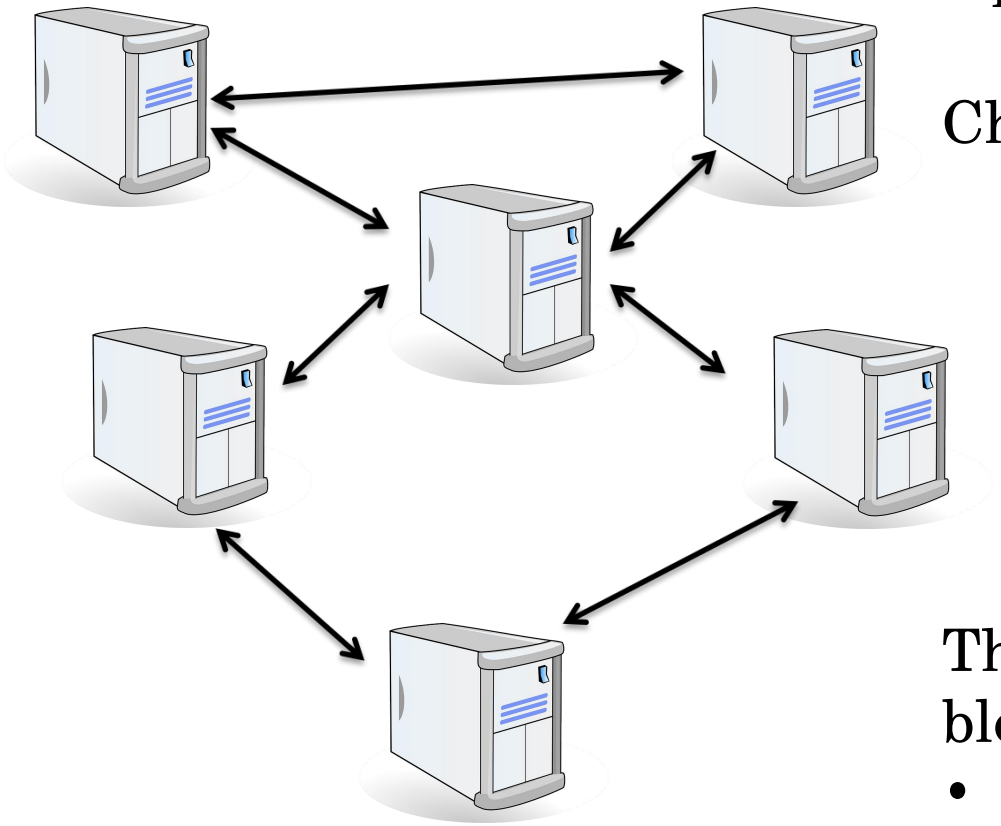
THE SOLUTION: DECENTRALIZATION



P2P Network

- Spread data and execution across a many nodes run by different entities
- Nodes are **mutually distrusting**

PEER-TO-PEER NETWORKS



Permissionless (or public) blockchains operate on a peer to peer network

Characteristics of a P2P network:

- Not all participants need to be known
- Each node is only connected to small number of other nodes

There also exist **permissioned** (or private) blockchains

- We will not talk about these in this class

~~BLOCKCHAIN~~ DECENTRALIZED LEDGER

Abstract Definition: Blockchains record transactions on a ledger

"Blockchain" is a misnomer

- Not all blockchains have **block**s
- Not all blockchains order transactions in a sequence (**chain**)

Decentralized ledger is the better term, but too wordy

- We will use the two interchangeably

BLOCKCHAIN VS. BLOCKCHAIN TECHNOLOGIES

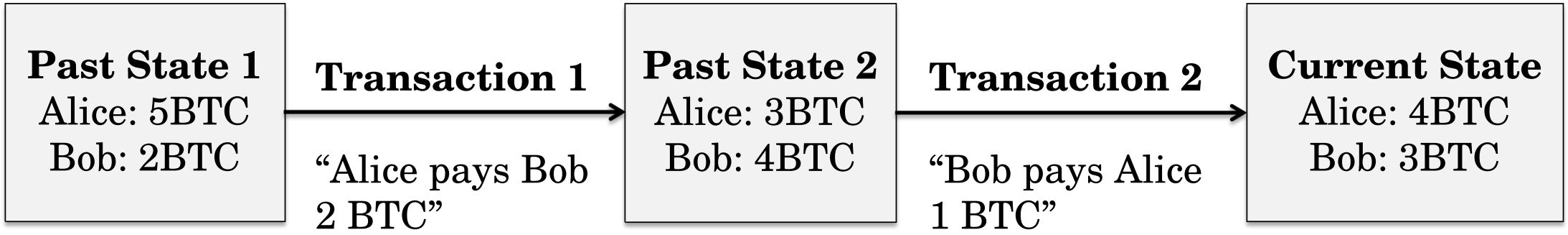
Blockchain (or Decentralized Ledger)

- The *abstraction* provided to applications and users
- Also, the data structure that holds transactions
  - Usually a directed acyclic graph (DAG)

Blockchains Technologies (or DLTs)

- The protocol that decides how nodes operate and how they talk to each other
- We will look a few different protocols throughout the semester

TRANSACTIONS



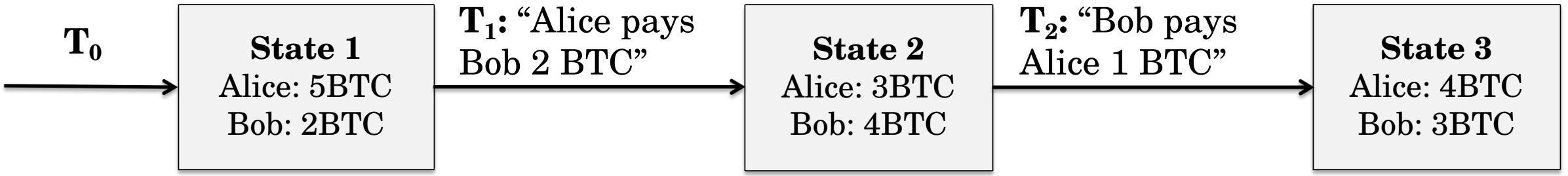
- At any point in time the ledger has some state (e.g., account balances)
- Transactions update the state of the ledger, e.g, by transferring money
- Meaning of the transaction is application and ledger specific

Main task of DLTs: Decide what transactions to accept and in which order!

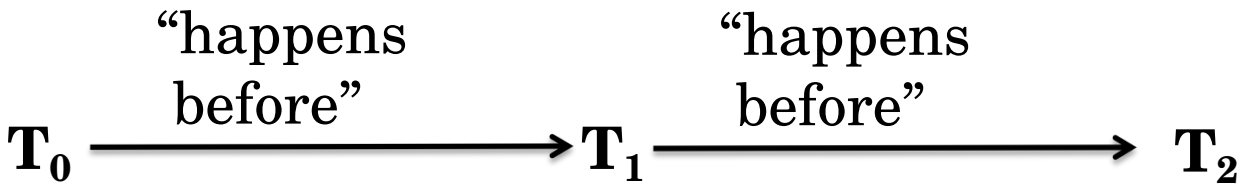
ORDERING TRANSACTIONS

Task of a DLT: Decide which transactions to accept and in what order

Sequence of States



Transaction Ledger



IMMUTABILITY

A decentralized ledger (=the set of transactions) is immutable

What does that mean?
(Hint: “Immutability” is also a slight misnomer)

- Transactions cannot be removed from the ledger
- New transactions can be added
- Newly added transactions cannot logically happen before already existing transactions

AUDITABILITY

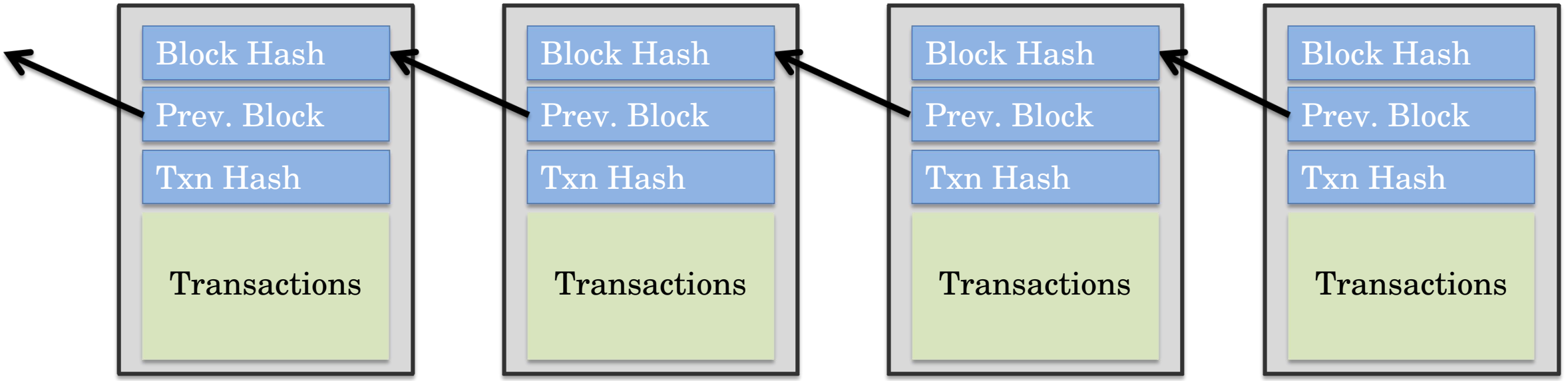
A decentralized ledger **history is public**

- We can view and verify past states
- Allows to create a log of transactions (similar to bank account statements)
- New nodes can join by replaying transaction history

Disadvantages

- Need to store additional data
- Leakage of metadata: All of your transaction history is public
- Leakage of data: The state (data) of smart contracts is public too

BITCOIN BLOCKCHAIN

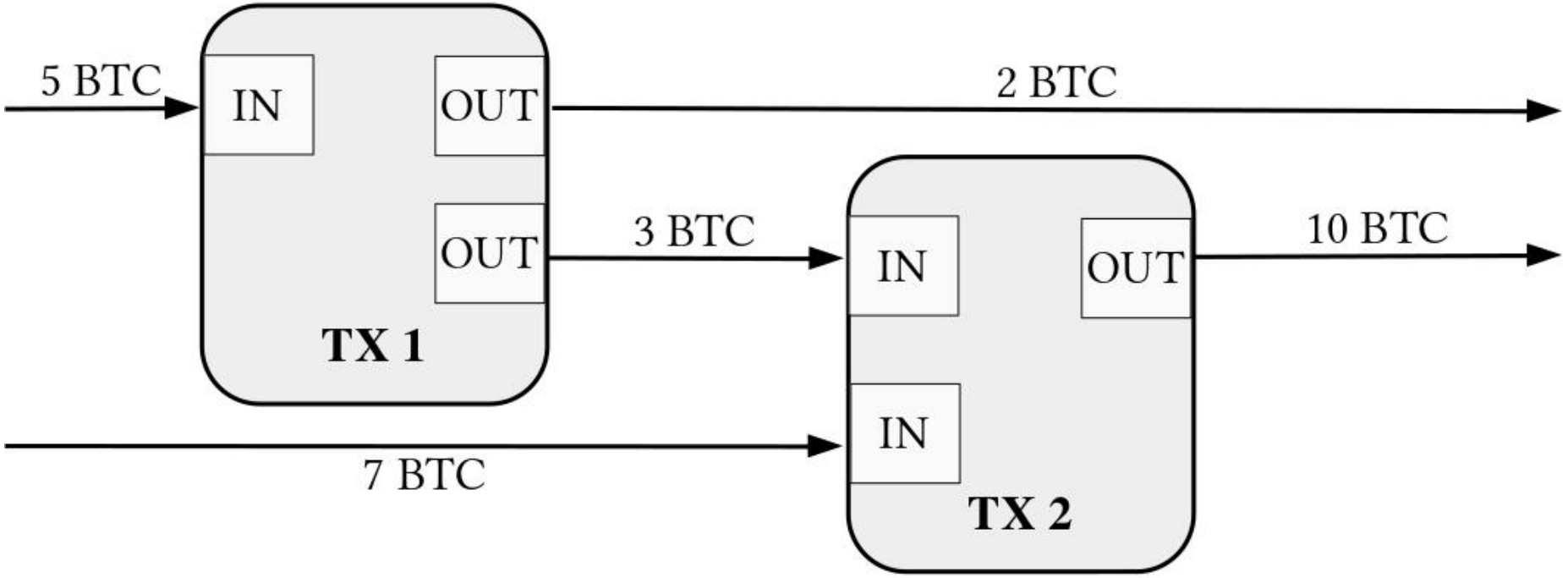


(Simplified: Some fields are omitted)

Where is the state stored?

State is stored outside of the blockchain itself
State can computed by traversing all blocks

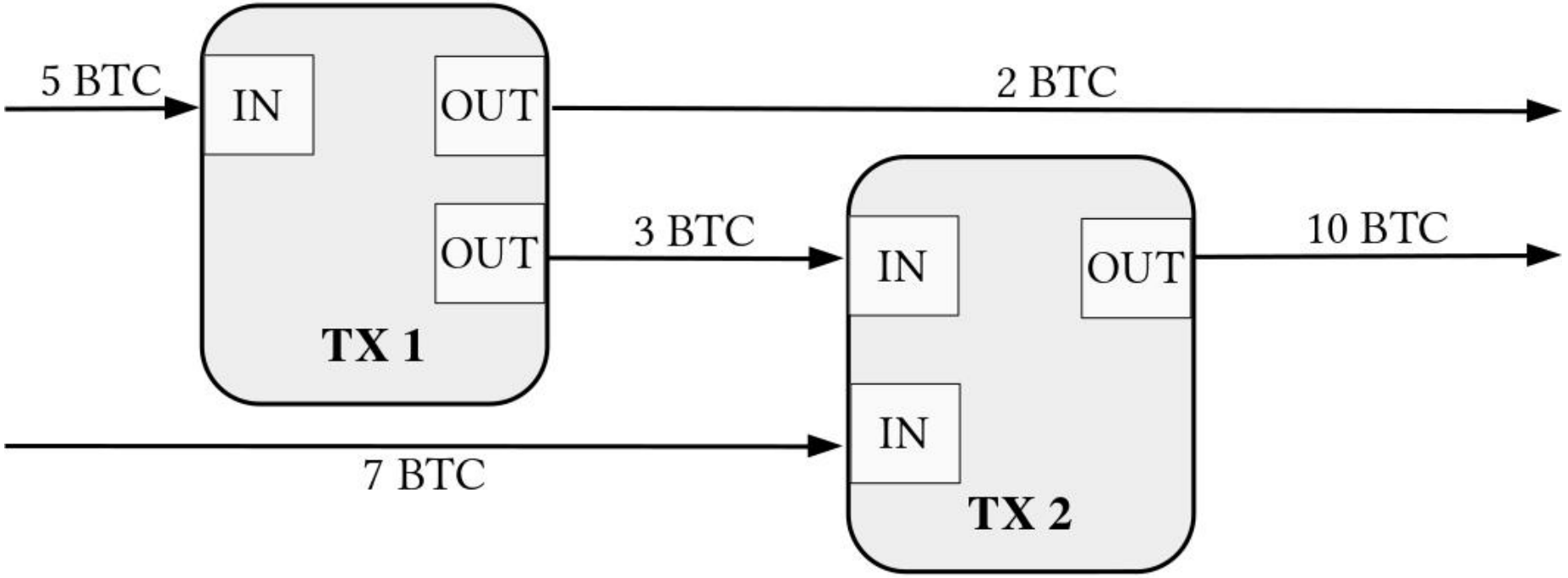
BITCOIN UTXOS



Bitcoin tracks Unspent Transaction Outputs (UTXOs) instead of account balances

- Nodes track the set of all unspent coins (or unspent transaction outputs)
- Each transaction consumes of **input coins** and creates a number of **output coins**

BITCOIN UTXOS



Advantages of the UTXO model:

- Requires only one simple data structure
- Provides a little more anonymity compared to accounts

THE UTXO SET

The UTXO set contains the all unspent coins

Each coin can be spent exactly once

- Cannot partially spend a coin

When processing a transaction:

- Remove all its input coins from the UTXO set
- Add all its output coins to the UTXO set

TRANSACTION VALIDITY

For a (Bitcoin) transaction to be valid three things must hold

1. The transaction is authorized to spend the input coins
- We will talk in more detail how that works some other time

2. The inputs of the transactions have a value of greater or equal to that of the outputs

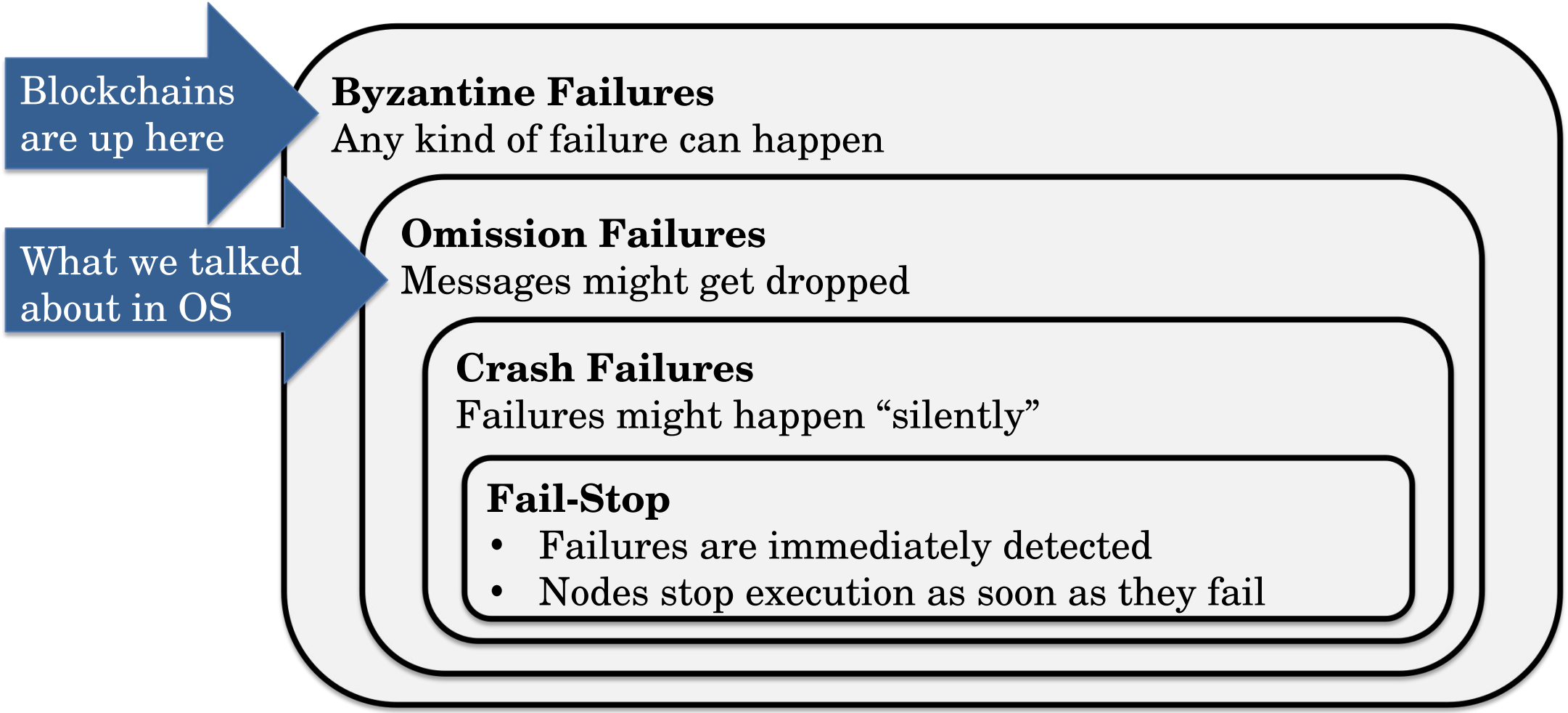
3. None of the inputs have been spent yet
- We can use the UTXO set to verify this

ABOUT FAILURES...

Blockchains operate in the face of a **strong failure model**

- Nodes can be faulty or misbehave intentionally
- Smart contracts contain bugs or malicious code

FAILURE MODELS



BYZANTINE FAILURES

What kind of arbitrary failures could there be?

- Software bugs
- OS failures
- Compromised nodes
- Malicious node operators
- Corrupted hardware

What kind of symptoms could these failures cause?

- Dropped or reordered messages
- Incorrect messages
- Nodes stop executing
- Leakage of private information

WHY YOU SHOULD TAKE THE COURSE

- Blockchain technologies are a rapidly emerging field
  - Smart contract developers are hard to find
  - Blockchain protocol developers are almost impossible to find
- Blockchains still have a long way to go
  - They are pretty slow, provide bad privacy, and waste tons of energy
  - Future you could fix some of these things!
- Many concepts you will learn here are applicable to other (distributed) systems

HOW TO SUCCEED IN THIS CLASS

- Attend lecture and ask questions
  - Totally fine to stay home, e.g., if you are sick
  - Watch recordings (and email questions) if you miss it
- Finish assigned readings and homeworks
  - There will be lecture notes and some white papers to read
  - Short homeworks are published weekly
- Start projects early!
- Come to office hours (or email) if you feel lost
  - The sooner you let me know the better I can help

THAT'S ALL (FOR TODAY)

Feel free to talk to me after class!