# Arhitecturi Paralele

## Consens în Sisteme Distribuite 2

Lect. Dr. Ing. Cristian Chilipirea
cristian.chilipirea@mta.ro

# Consens rezistent la Faults - Paxos

## The Part-Time Parliament

LESLIE LAMPORT
Digital Equipment Corporation

---

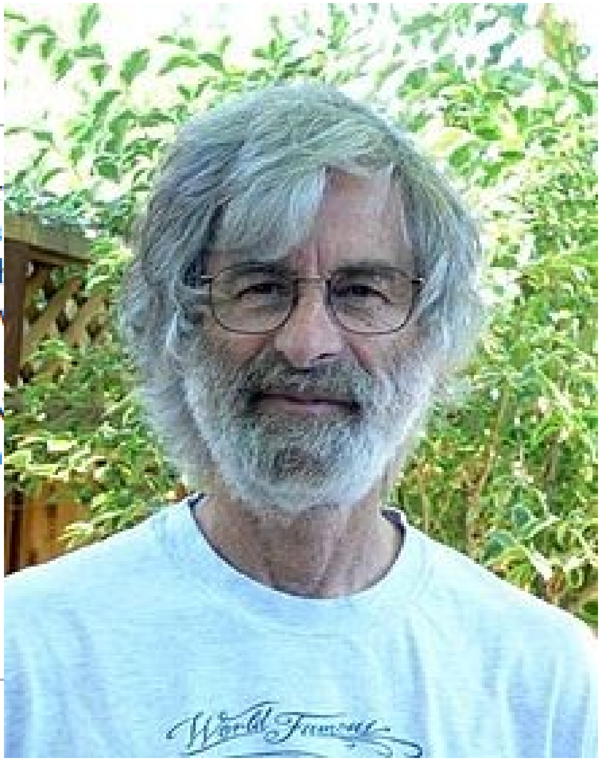
Recent archaeological discoveries on the island of Paxos reveal that the par
spite the peripatetic propensity of its part-time legislators. The legislators
copies of the parliamentary record, despite their frequent forays from the c
fulness of their messengers. The Paxon parliament's protocol provides a ne
the state-machine approach to the design of distributed systems.

Categories and Subject Descriptors: C2.4 [**Computer-Communications N**
Systems—*Network operating systems*; D4.5 [**Operating Systems**]: Relia
J.1 [**Administrative Data Processing**]: Government

General Terms: Design, Reliability

Additional Key Words and Phrases: State machines, three-phase commit,

---

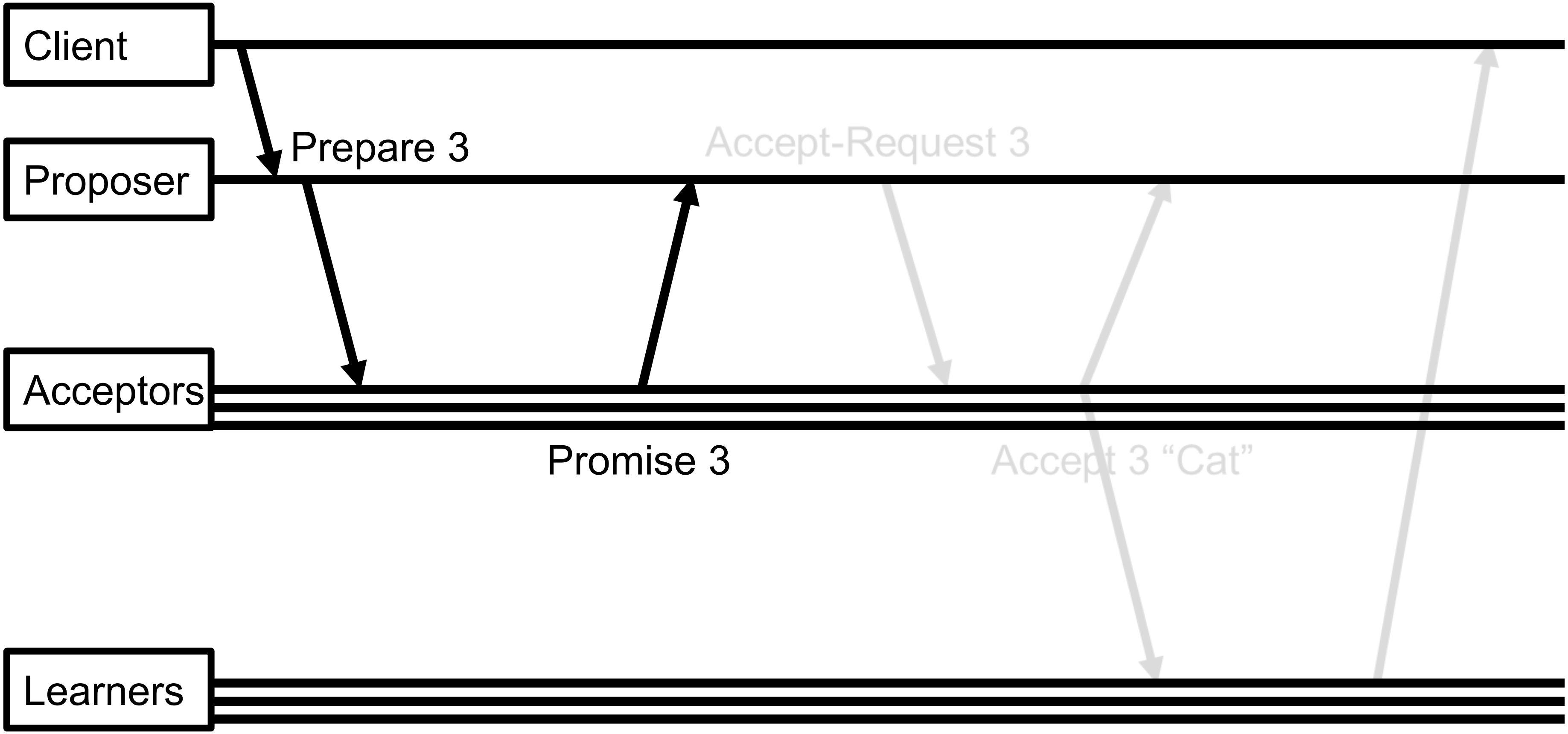


# Paxos

**Phase 1.** (a) A proposer selects a proposal number $n$ and sends a *prepare* request with number $n$ to a majority of acceptors.

(b) If an acceptor receives a *prepare* request with number $n$ greater than that of any *prepare* request to which it has already responded, then it responds to the request with a promise not to accept any more proposals numbered less than $n$ and with the highest-numbered proposal (if any) that it has accepted.

# Paxos

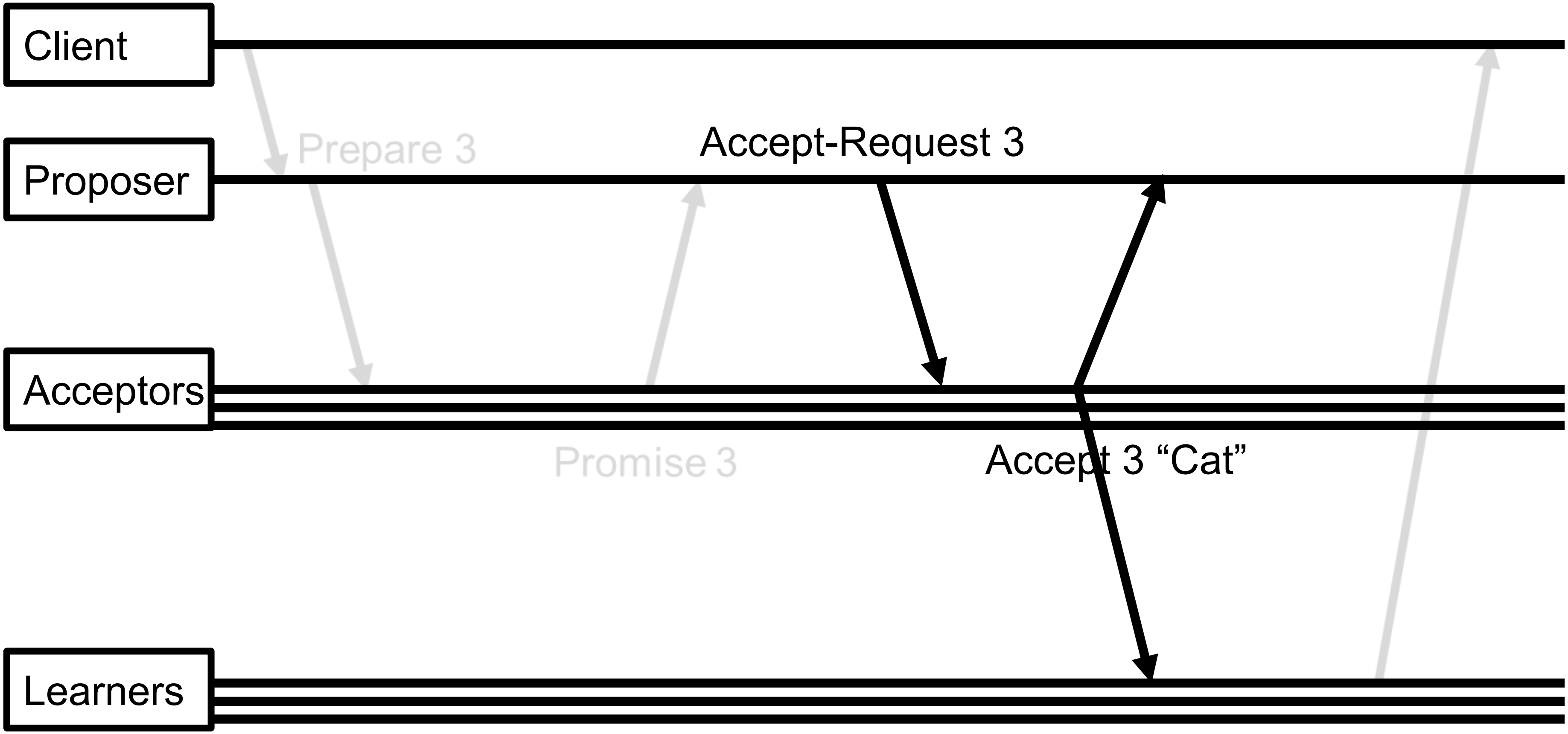

Client
Proposer
Acceptors
Learners
Prepare 3
Promise 3
Accept-Request 3
Accept 3 “Cat”

# Paxos

**Phase 2.** (a) If the proposer receives a response to its *prepare* requests (numbered $n$) from a majority of acceptors, then it sends an *accept* request to each of those acceptors for a proposal numbered $n$ with a value $v$, where $v$ is the value of the highest-numbered proposal among the responses, or is any value if the responses reported no proposals.

(b) If an acceptor receives an *accept* request for a proposal numbered $n$, it accepts the proposal unless it has already responded to a *prepare* request having a number greater than $n$.

# Paxos

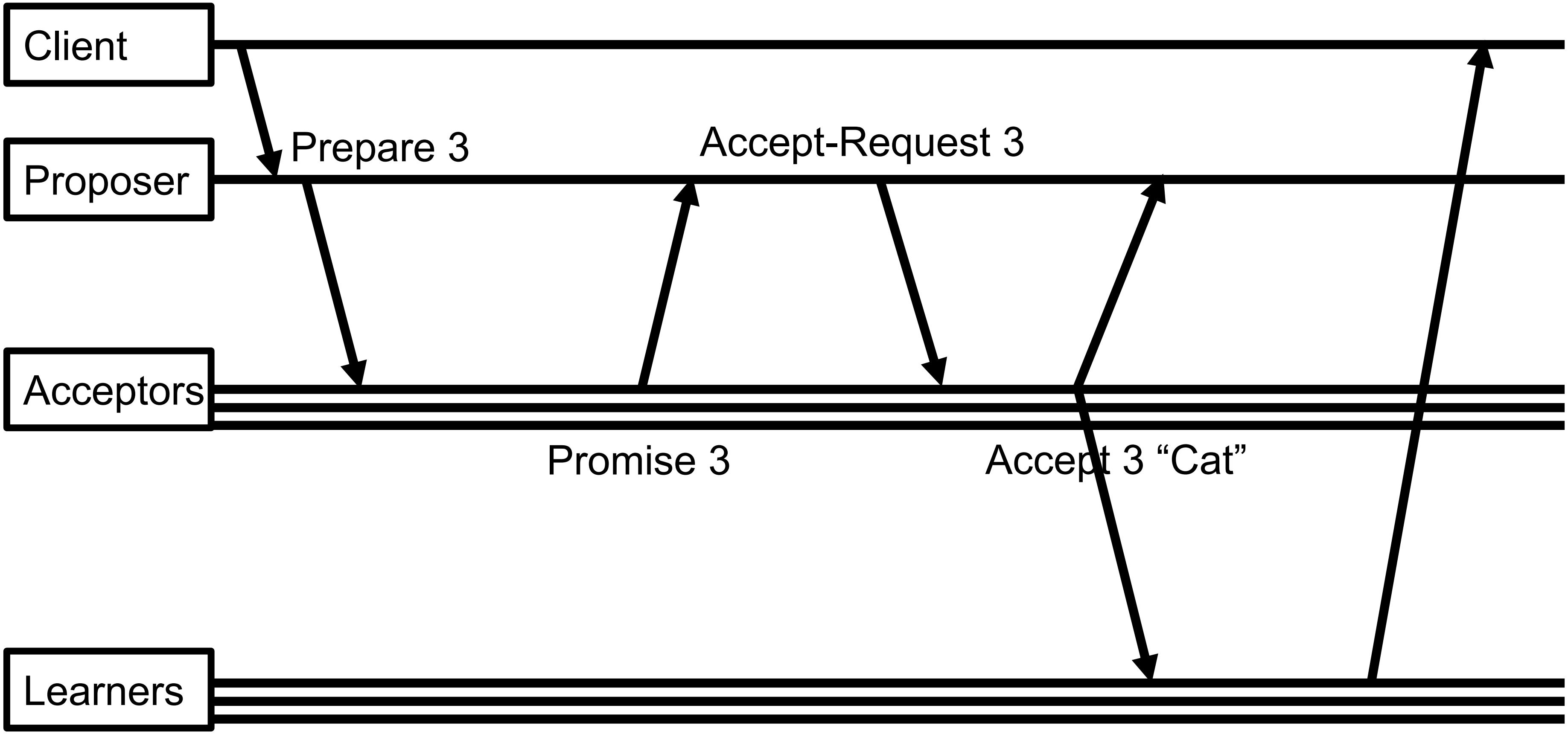

Client
Proposer
Acceptors
Learners
Prepare 3
Accept-Request 3
Promise 3
Accept 3 “Cat”

# Paxos

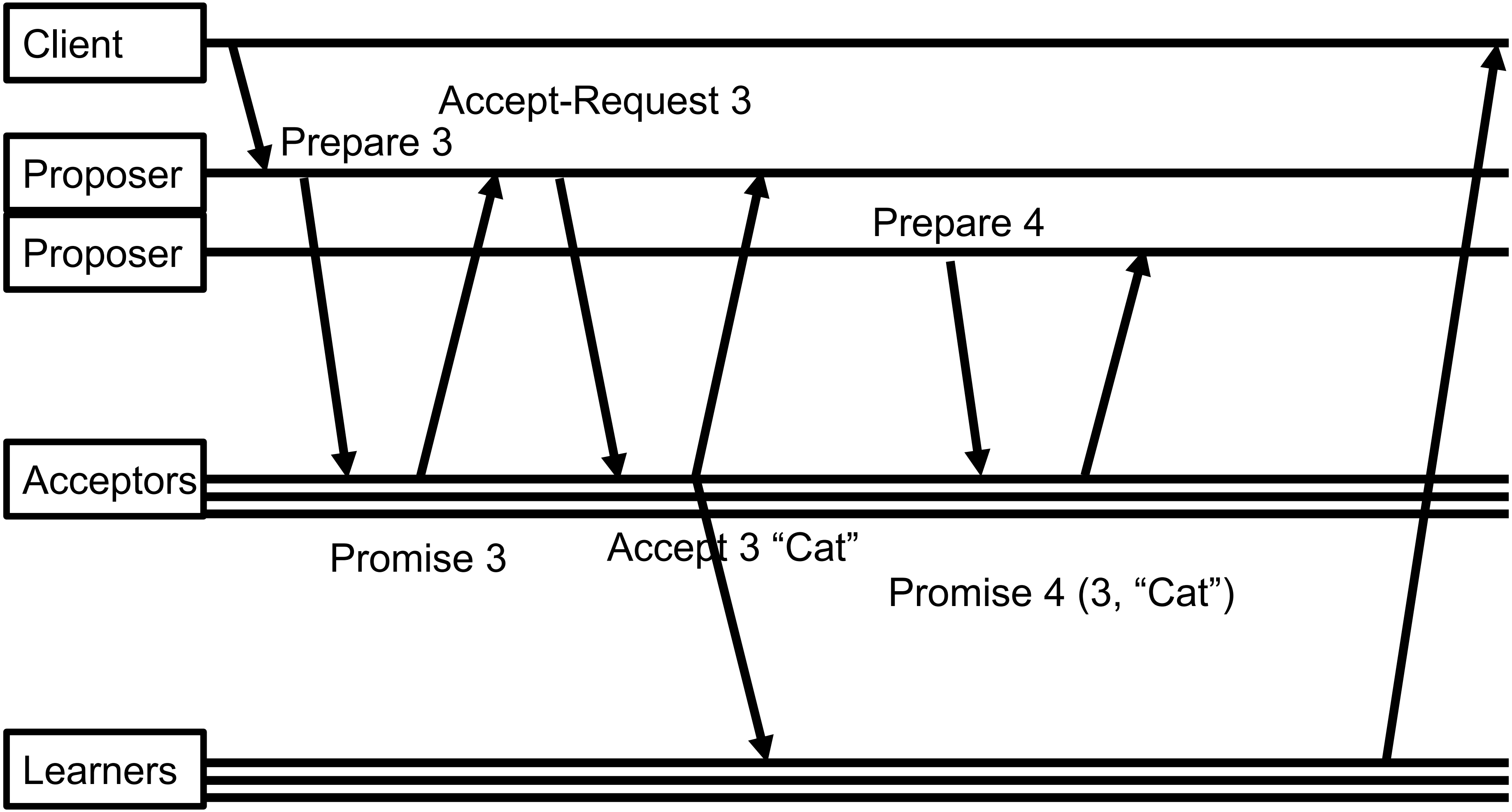
## 2.3 Learning a Chosen Value

To learn that a value has been chosen, a learner must find out that a proposal has been accepted by a majority of acceptors. The obvious algorithm is to have each acceptor, whenever it accepts a proposal, respond to all learners, sending them the proposal. This allows learners to find out about a chosen value as soon as possible, but it requires each acceptor to respond to each learner—a number of responses equal to the product of the number of acceptors and the number of learners.

# Paxos


Client
Proposer
Acceptors
Learners
Prepare 3
Accept-Request 3
Promise 3
Accept 3 “Cat”

# Paxos



Client
Accept-Request 3
Prepare 3
Proposer
Proposer
Prepare 4
Acceptors
Promise 3
Accept 3 “Cat”
Promise 4 (3, “Cat”)
Learners

# CAP Theorem

**Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services**

Seth Gilbert and Nancy Lynch
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
sethg@mit.edu,lynch@theory.lcs.mit.edu

**Abstract**

When designing distributed web services, there are three pro
desired: consistency, availability, and partition tolerance. It
three. In this note, we prove this conjecture in the asynchr
then discuss solutions to this dilemma in the partially synch

## 1 Introduction

At PODC 2000, Brewer[1], in an invited talk [2], made the followin
a web service to provide the following three guarantees:

- Consistency
- Availability
- Partition-tolerance

# Bitcoin and Blockchain

## 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 attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.


### 1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for

Cheats on the CAP theorem

**Eventual** consistency
Availability
Partition-tolerance (**But short**)

# Proof of Work

# Proof of Work

## Pricing via Processing or Combatting Junk Mail

Cynthia Dwork and Moni Naor

IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

**Abstract.** We present a computational technique for combatting junk mail in particular and controlling access to a shared resource in general. The main idea is to require a user to compute a moderately hard, but not intractable, function in order to gain access to the resource, thus preventing frivolous use. To this end we suggest several *pricing functions*, based on, respectively, extracting square roots modulo a prime, the Fiat-Shamir signature scheme, and the Ong-Schnorr-Shamir (cracked) signature scheme.

### 1 Introduction

Recently, one of us returned from a brief vacation, only to find 241 messages in our reader. While junk mail has long been a nuisance in hard (snail) mail, we

# Proof of Work

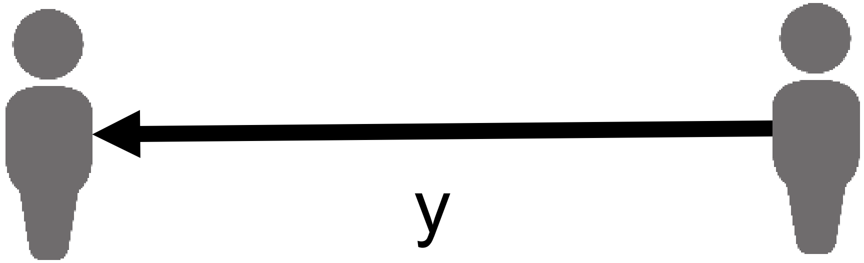
- $y = f(x)$
  - f() este extrem de lentă computațional
- $x = g(y)$
  - g() este extrem de rapidă computațional

# Proof of Work

# Proof of Work

# Proof of Work

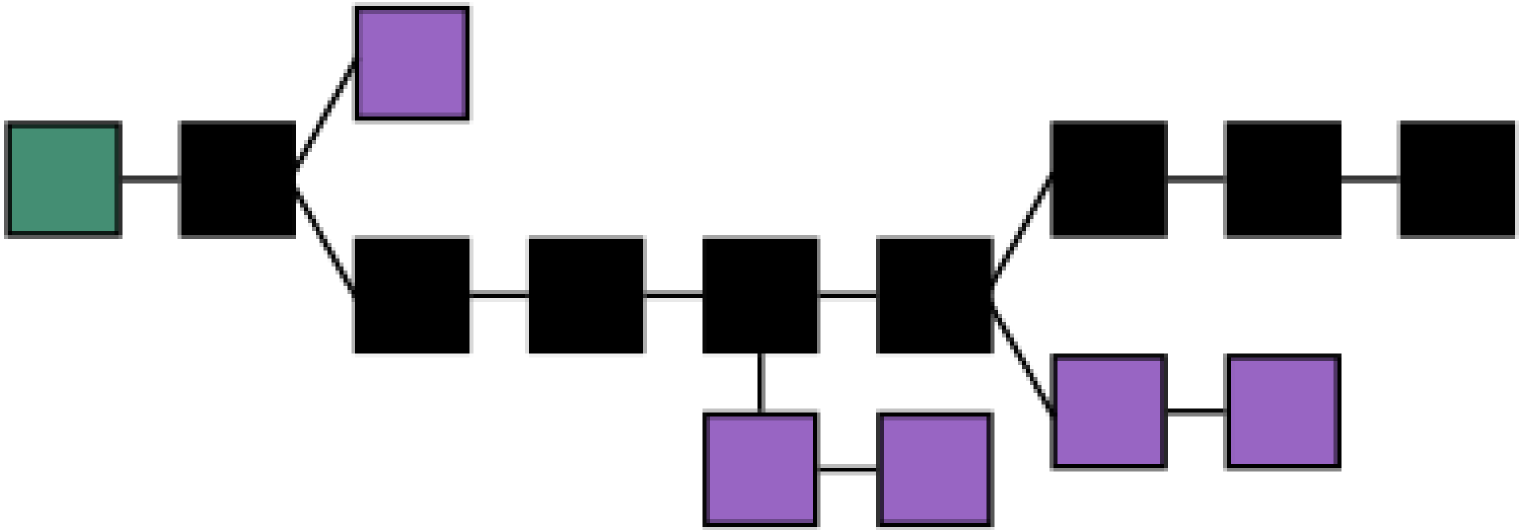
# Proof of Work



# Proof of Work

g(y) == x?
DA - știu că a muncit
NU – încearcă să trișeze

# Blockchain



# Bitcoin energy consmption

Bitcoin Mining and its Energy Footprint

**Karl J. O'Dwyer† and David Malone***

*Hamilton Institute*
*National University of Ireland Maynooth*

E-mail: †karl.a.odwyer@nuim.ie *david.malone@nuim.ie

*Abstract* — Bitcoin is a digital cryptocurrency that has generated considerable public interest, including both booms in value and busts of exchanges dealing in Bitcoins. One of the fundamental concepts of Bitcoin is that work, called mining, must be done in checking all monetary transactions, which in turn creates Bitcoins as a reward. In this paper we look at the energy consumption of Bitcoin mining. We consider if and when Bitcoin mining has been profitable compared to the energy cost of performing the mining, and conclude that specialist hardware is usually required to make Bitcoin mining profitable. We also show that the power currently used for Bitcoin mining is comparable to Ireland's electricity consumption.

# Bitcoin energy consmption

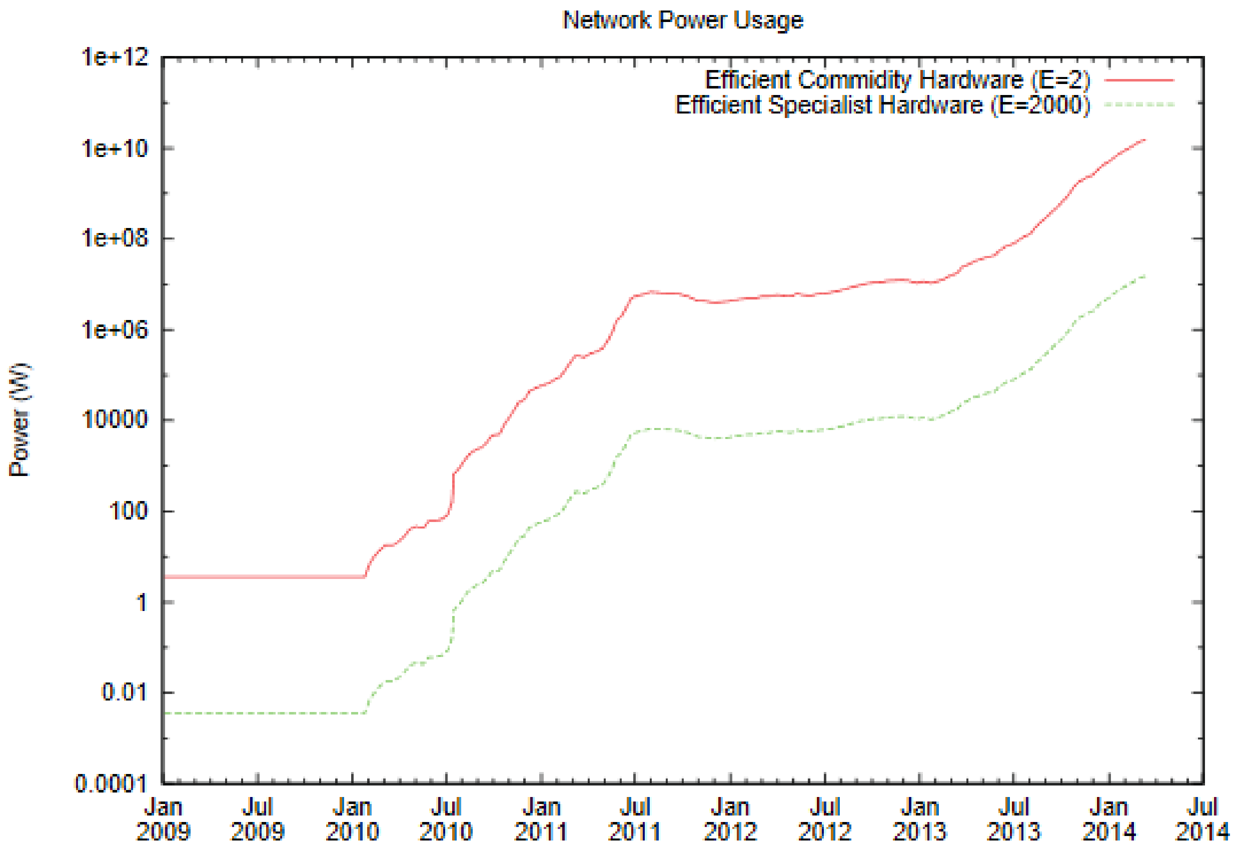



Fig. 5: Estimated Power Consumption of the Bitcoin Mining Network.

# Transactions

# ACID

**Principles of Transaction-Oriented Database Recovery**

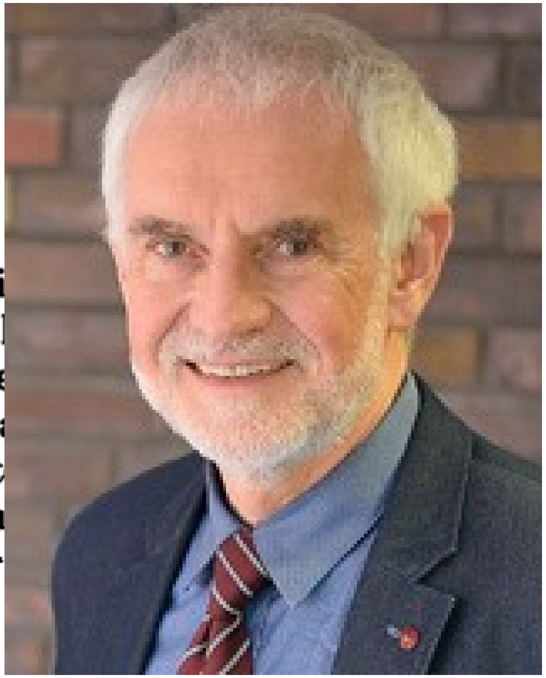
THEO HAERDER

*Fachbereich Informatik, University of Kaiserslautern, West Germany*

ANDREAS REUTER[1]

*IBM Research Laboratory, San Jose, California 95193*

er, a terminological framework is provided for descri
covery schemes for database systems in a conceptual
ation-dependent way. By introducing the terms mate
strategy, and checkpoint, we obtain a means for cl
ations from a unified viewpoint. This is complement
logging techniques, which are precisely defined by u
these criteria are related to all relevant questions



# ACID

- Atomicity
- Consistency
- Isolation
- Durability

# Distributed Hash Table Chord

## Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Ion Stoica; Robert Morris, David Karger, M. Frans Kaas
MIT Laboratory for Computer Science
chord@lcs.mit.edu
http://pdos.lcs.mit.edu/chord/

**Abstract**

A fundamental problem that confronts peer-to-peer applications is to efficiently locate the node that stores a particular data item. This paper presents *Chord*, a distributed lookup protocol that addresses this problem. Chord provides support for just one operation: given a key, it maps the key onto a node. Data location can be easily implemented on top of Chord by associating a key with each data item, and storing the key/data item pair at the node to which the key maps. Chord adapts efficiently as nodes join and leave the system, and can answer queries even if the system is continuously changing. Results from theoretical analysis, simulations, and experiments show that Chord is scalable, with communication cost and the state maintained by each node scaling logarithmically with the number of Chord nodes.

and involves relati
and leave the syster
Previous work o
aware of most othe
scale to large numb
"routing" informati
routing table is dis
communicating wi
an $N$-node system
$O(\log N)$ other no
sages to other node
nodes join and lea
event results in no
Three features th
peer lookup protoc

# Amazon DynamoDB

# Caching

# Peer to Peer

# Geolocation

# Content Delivery Networks

# Load Balancing