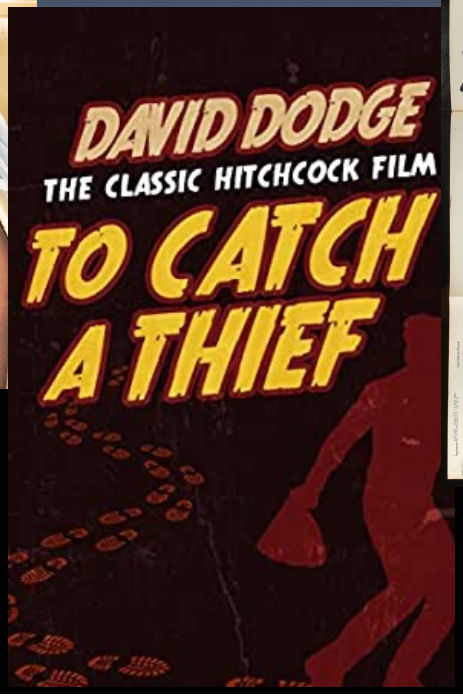
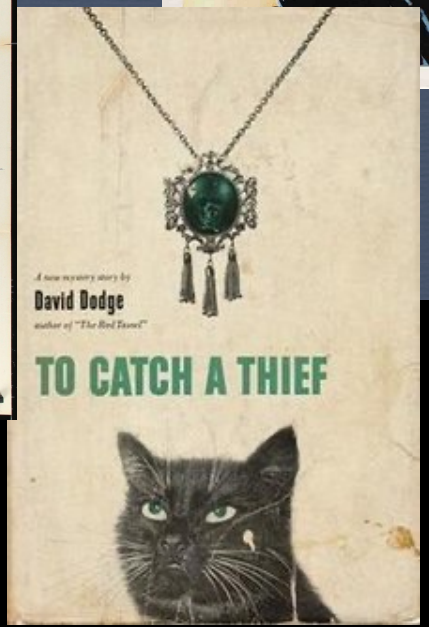
To Catch a Thief

Seth Gilbert





Accountability in distributed systems

Why do we care about accountability?

What is accountability?

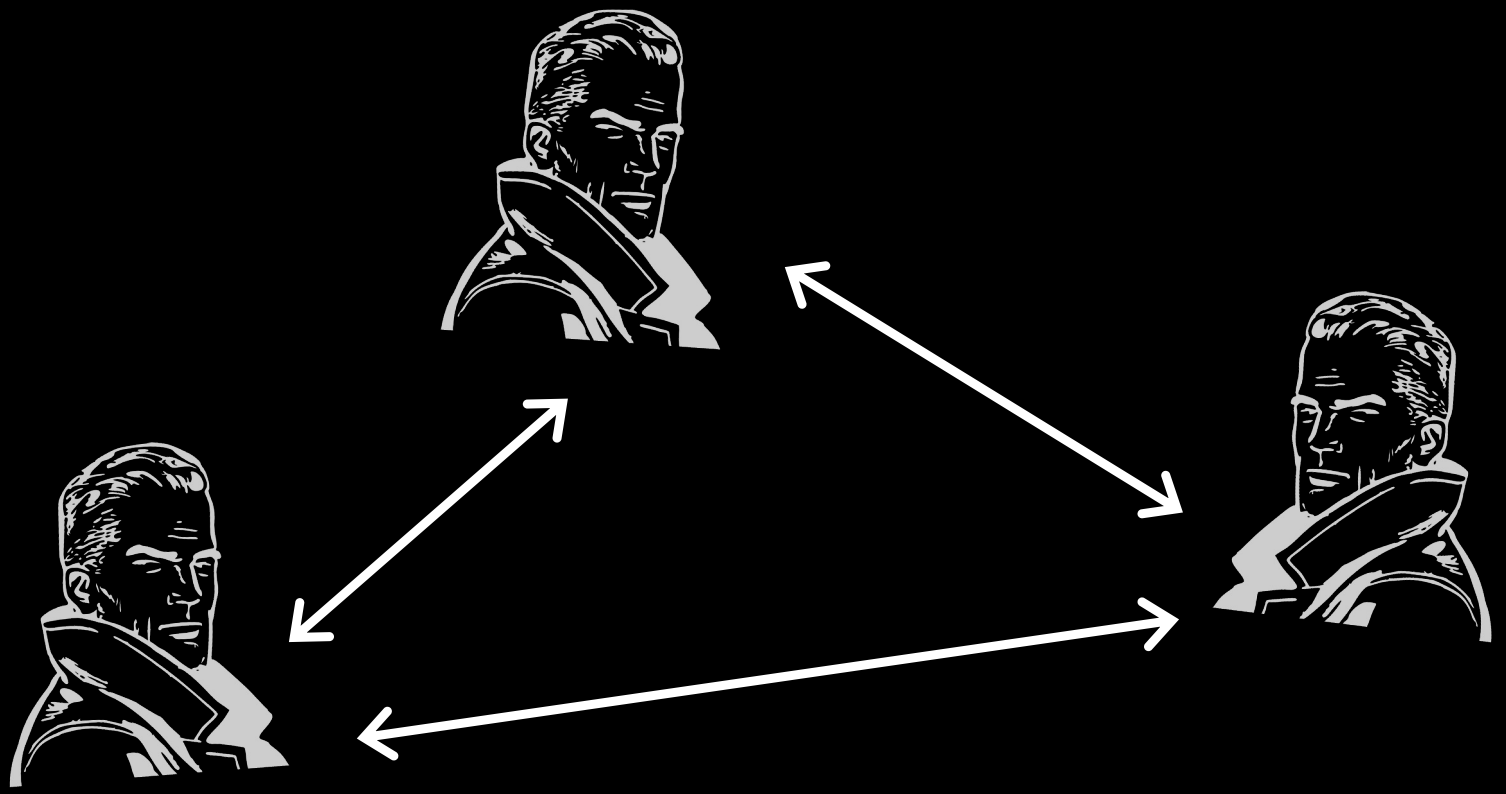
How do we implement accountability?

What should we be working on?

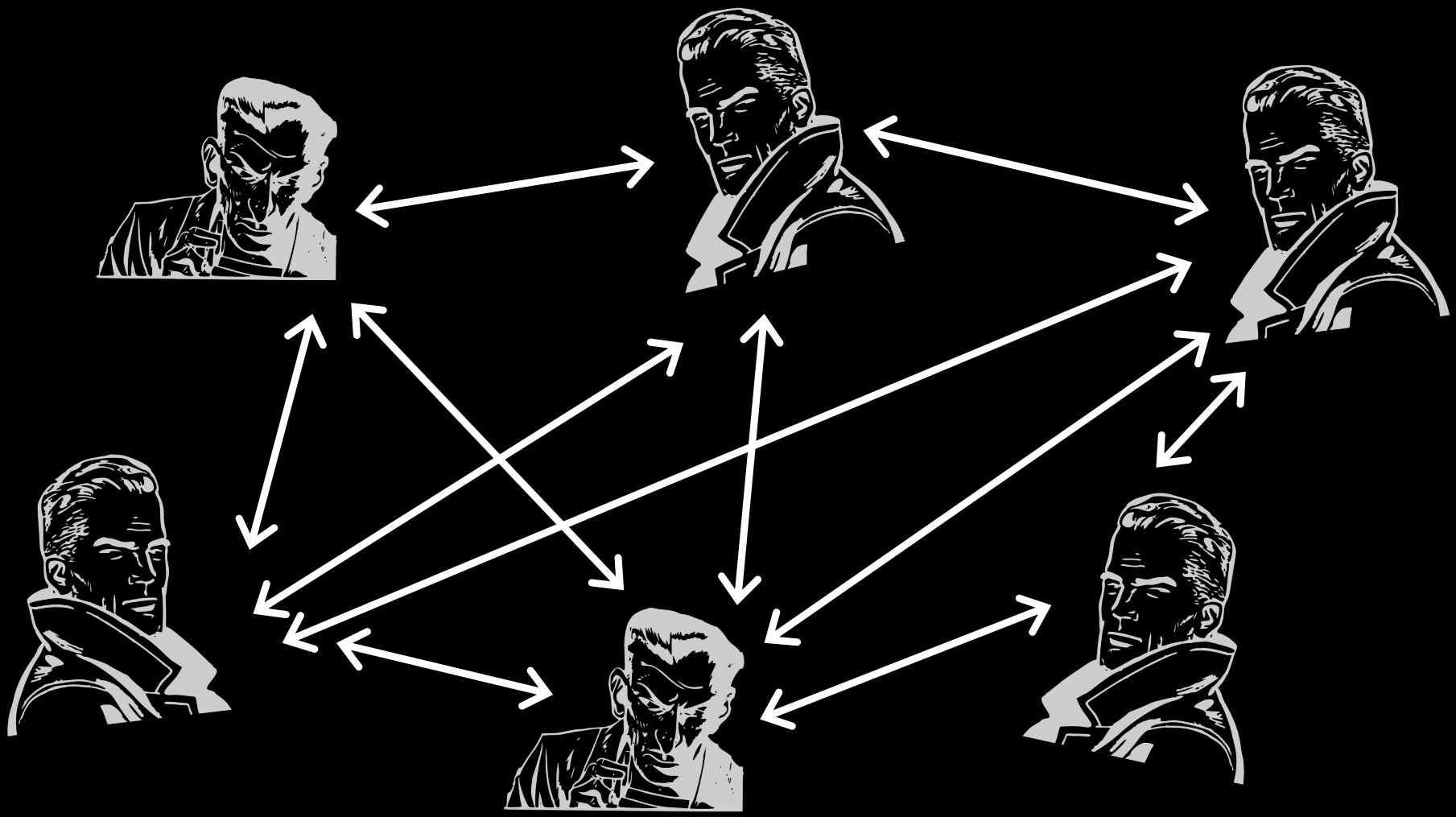
How do we make trust scale?

Goal:
To build large distributed services that allow users to cooperate and collaborate productively.

How do we make trust scale?

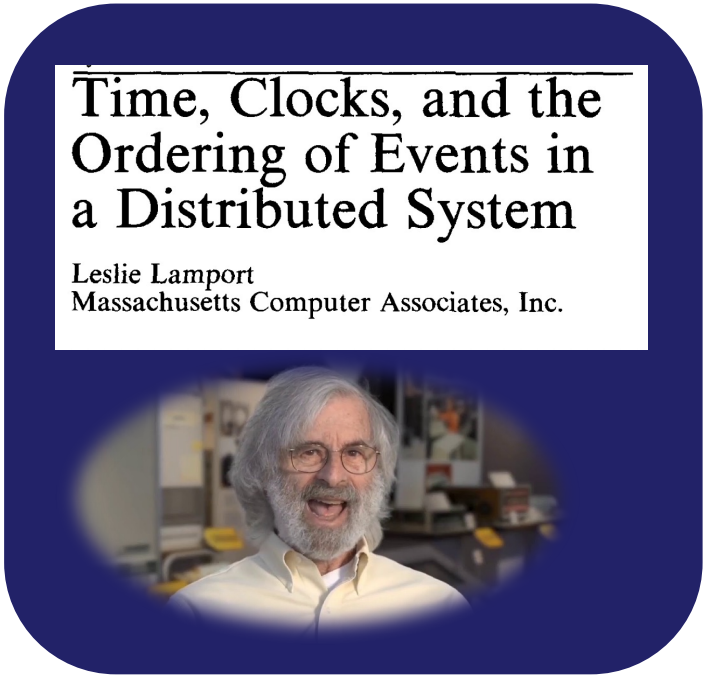


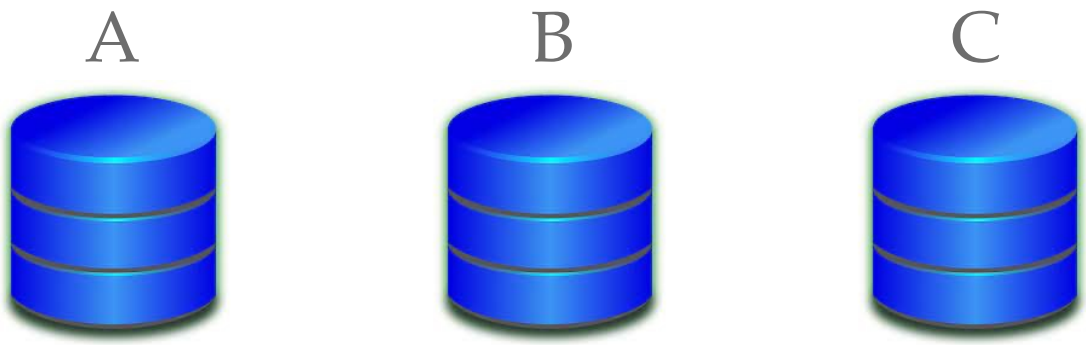
How do we make trust scale?



How do we make trust scale?

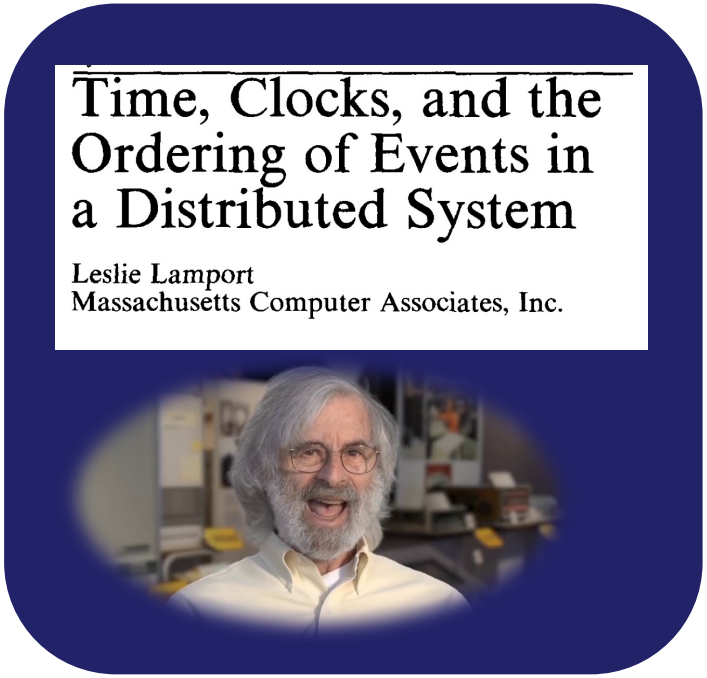
A recipe for a robust distributed service…

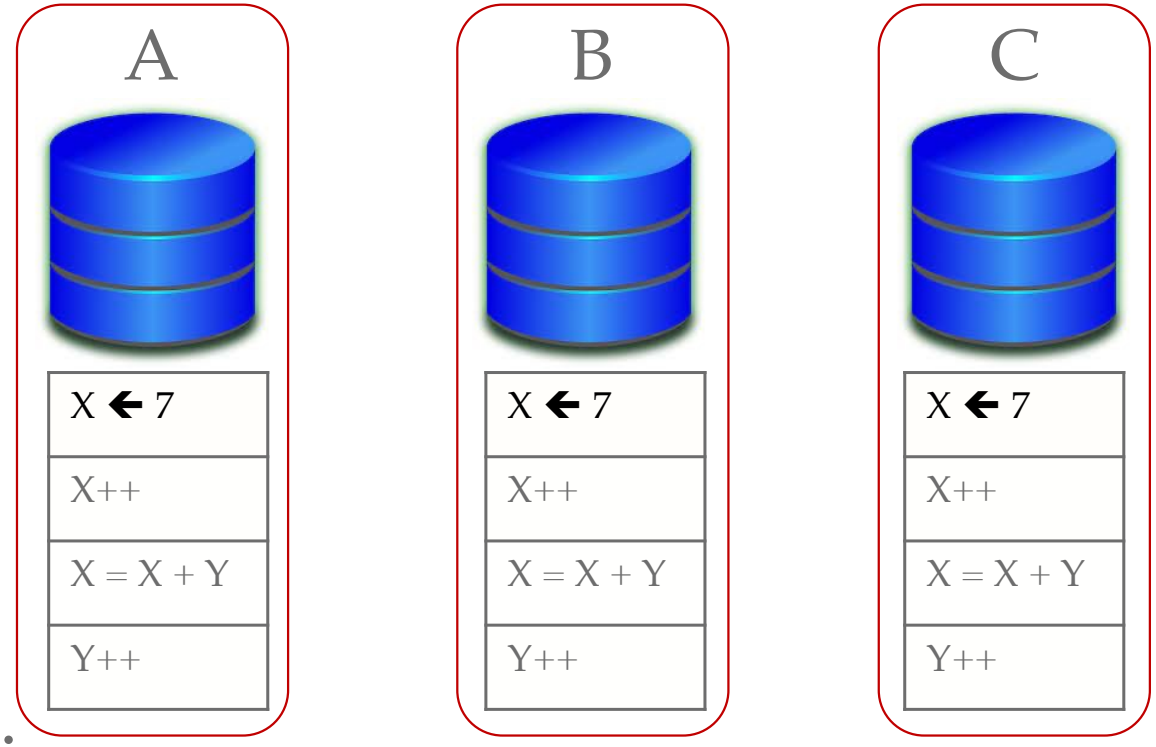




1. Replicate your data.

A recipe for a robust distributed service…



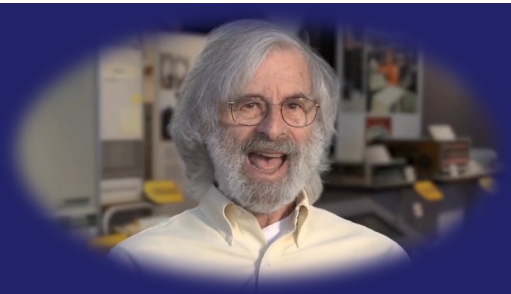


1. Replicate your data.
2. Each replicate executes an identical "operation log."

A recipe for a robust distributed service…

Time, Clocks, and the Ordering of Events in a Distributed System

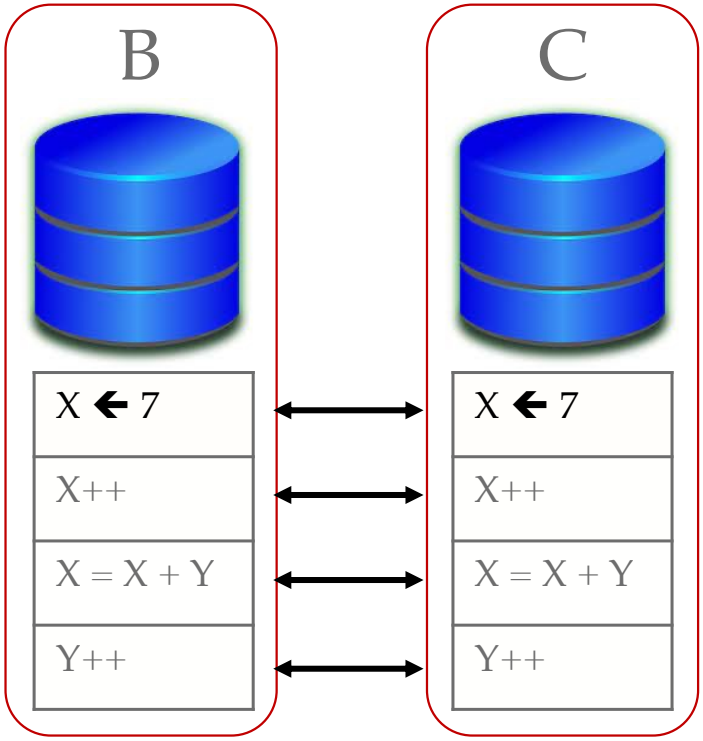
Leslie Lamport
Massachusetts Computer Associates, Inc.



How to Build a Highly Available System Using Consensus

Butler W. Lampson[1]

Microsoft
180 Lake View Av., Cambridge, MA 02138

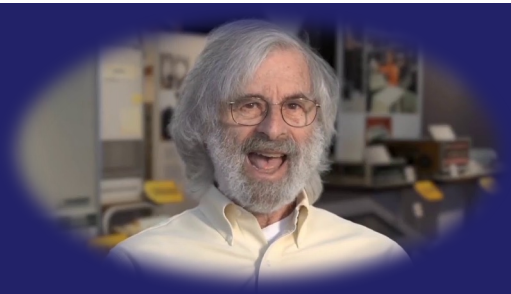


1. Replicate your data.
2. Each replicate executes an identical "operation log."
3. Use consensus protocols to agree on the logs.

A recipe for a robust distributed service…

Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport
Massachusetts Computer Associates, Inc.



How to Build a Highly Available System Using Consensus

Butler W. Lampson[1]

Microsoft
180 Lake View Av., Cambridge, MA 02138

Practical Byzantine Fault Tolerance

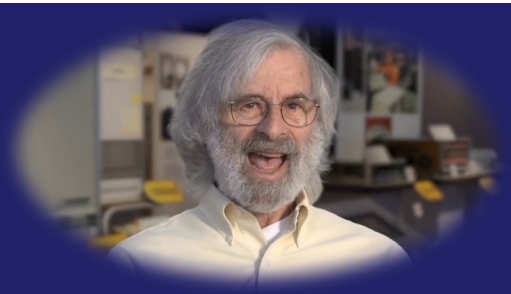
Miguel Castro and Barbara Liskov
Laboratory for Computer Science,
Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139
{castro,liskov}@lcs.mit.edu

1. Replicate your data.
2. Each replicate executes an identical “operation log.”
3. Use consensus protocols to agree on the logs.
4. Make it Byzantine fault-tolerant…

A recipe for a robust distributed service…

Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport
Massachusetts Computer Associates, Inc.



How to Build a Highly Available System Using Consensus

Butler W. Lampson[1]

Microsoft
180 Lake View Av., Cambridge, MA 02138

Practical Byzantine Fault Tolerance

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

Distributed databases

Distributed lock servers

Distributed filesystems

Blockchains

Is Byzantine Fault Tolerance enough?

What are the problems?

Efficiency / Scalability / Bandwidth Usage

Byzantine-fault tolerance is expensive:

- At least $\Omega(n^2)$ communication per "agreement," if the network is well behaved.
- Even worse when the network is not well behaved.

The real bottleneck is bandwidth!

Dolev, Reischuk: "Bound on information exchange for Byzantine agreement", JACM 1985

What are the problems?

Efficiency / Scalability / Bandwidth Usage

Byzantine-fault tolerance is expen

- At least $\Omega(n^2)$ communication per "agreement," if the network is well behaved.
- Even worse when the network is not well behaved.

Preview:
First deterministic protocol for partially synchronous networks that achieves $O(n^2)$ will appear at DISC 2022.

The real bottleneck is bandwidth!

Dolev, Reischuk: "Bound on information exchange for Byzantine agreement", JACM 1985

What are the problems?

Strong analytic assumptions, weak reality

Theory vs. practice:

- Network is well connected and timely

Example: Eclipse Attack
Bitcoin communication is via an *overlay network*, where malicious users can hijack all the communication of an honest node.

- Most users follow the protocol perfectly

Example: Selfish Mining Blocs
Greedy users may work together to violate the protocol and make more money for everyone.

Big research questions today:

Can we improve “common case” efficiency?

Can we cope with extraordinary situations?

Can we disincentivize attacks on the system?

A little accountability goes a long way….

Accountability

If users attack the system / violate the protocol, we can (provably and reliably) identify them.

Accountability

If users attack the system / violate the protocol, we can (provably and reliably) identify them.

Key pioneering work in distributed computing:

PeerReview: Practical accountability for distributed systems (SOSP 2007)
by Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel

➔ General transformation to make any distributed system accountable!

…but there are some limitations.

Accountability

If users attack the system / violate the protocol, we can (provably and reliably) identify them.

Real-world motivation (starting next week?):

Ethereum Proof-of-Stake depends on accountability to incentivize stakers.

- Casper implements a version of "accountable" agreement (with "*plausible liveness*").
- Stakers (i.e., processes that perform agreement) deposit a large stake.
- If they violate the protocol, their stake is "slashed"

Accountability

If users attack the system / violate the protocol, we can (provably and reliably) identify them.

Unfortunate reality:

- In non-synchronous systems, impossible to guarantee that we will identify malicious users.
- How do we differentiate “slow network” from “protocol violation where message is not sent”??

Accountability

If users violate the protocol
in a way that (visibly) causes trouble
then we can (provably and reliably) identify them.

Example: Consensus (Byzantine Agreement)

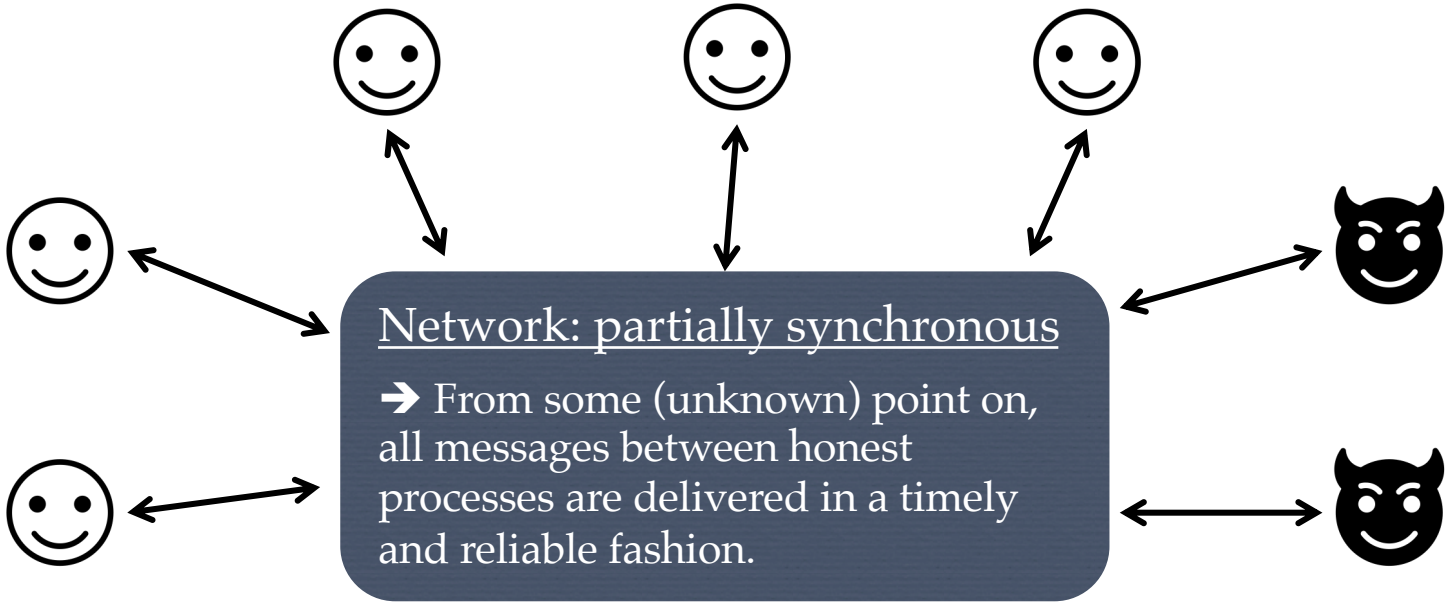
- Key safety property: every process outputs the same decision.
- Accountability property: *if there is disagreement,* then any honest process can produce irrefutable (cryptographic) evidence of the identities of the attackers.

Some algorithms…

System Model

Consensus is only achievable if $t < n/3$

n processes



t processes are faulty / malicious / Byzantine

Accountable Consensus

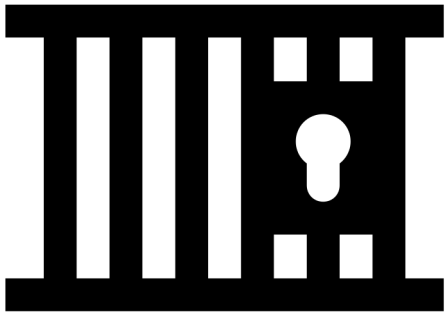
Input: Propose

Every process receives a proposal as input.

Output: Decide

Every (honest) process outputs a decision.

Output: Detect



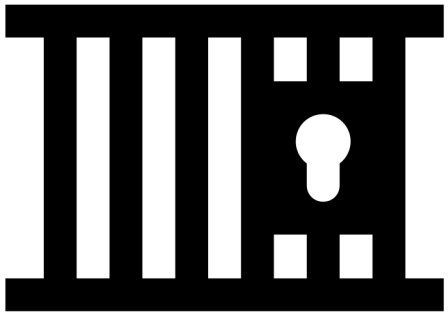
Every (honest) process outputs accountability violations.

Accountable Consensus

Input: Propose

Output: Decide

Output: Detect



Guarantees:

1. *Conditional Agreement*: If $t < n/3$, then all honest processes outputs the same value.
2. *Conditional Validity*: If $t < n/3$ and all honest processes start with v, then v is the only decision.
3. *Conditional Termination*: If $t < n/3$, then all honest processes terminate.
4. *Accountability*: If 2 honest nodes output different values, then every honest process outputs $n/3$ ids that provably cheated. (No honest processes are every suspected.)

Are there any accountable consensus protocols?

The Polygraph Protocol

Polygraph: Accountable Byzantine Agreement
by Pierre Civit, Seth Gilbert, and Vincent Gramoli
ICDCS 2021

Of note:

- First (provably) accountable consensus protocol.
- $O(n^4)$ message complexity.
- Implemented in Red Belly Blockchain

The Polygraph Protocol

Polygraph: Accountable Byzantine Agreement
by Pierre Civit, Seth Gilbert, and Vincent Gramoli
ICDCS 2021

Key ideas:

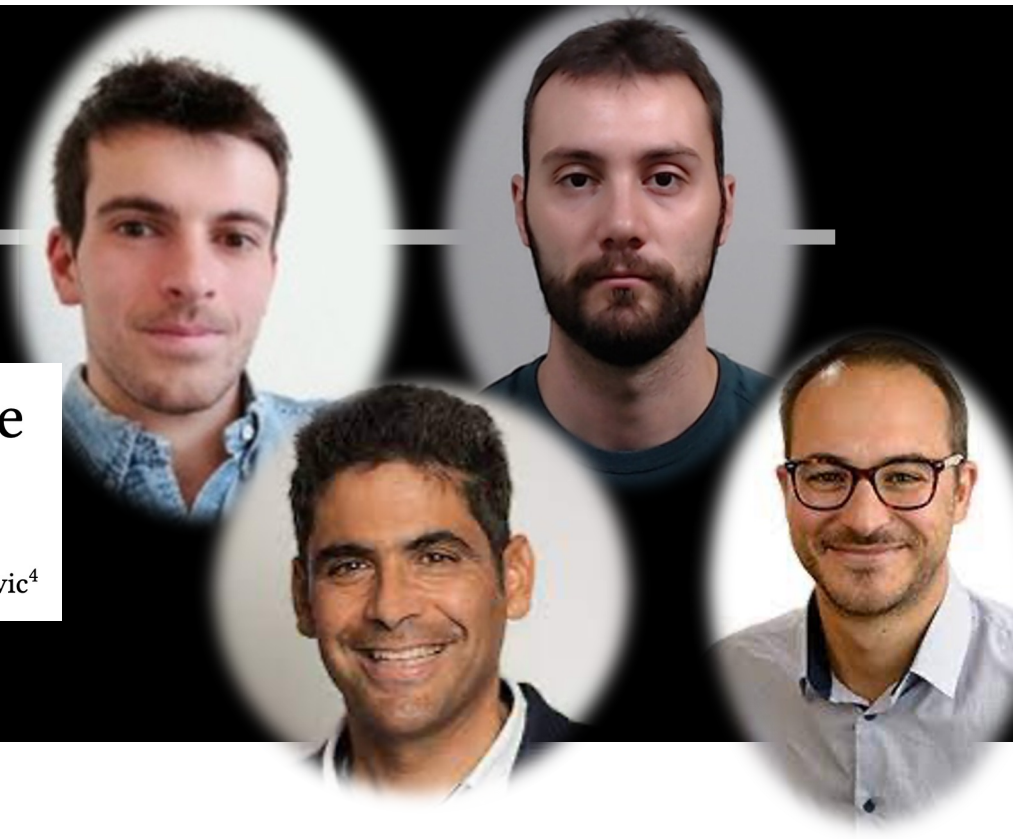
- Only decide when you have signatures proving that enough other processes have committed to that value.
- However, attaching n signatures for every possible decision to every message significantly increases the communication complexity.
- Main challenge: carefully limiting which signatures are sent at which times to maintain reasonable communication complexity.

Can we make any agreement protocol accountable?

Polygraph is a good proof of concept…

But it is too expensive.

As easy as ABC…



As easy as ABC: Optimal (A)ccountable (B)yzantine (C)onsensus is easy!

Pierre Civit[1], Seth Gilbert[2], Vincent Gramoli[3,4], Rachid Guerraoui[4] and Jovan Komatovic[4]

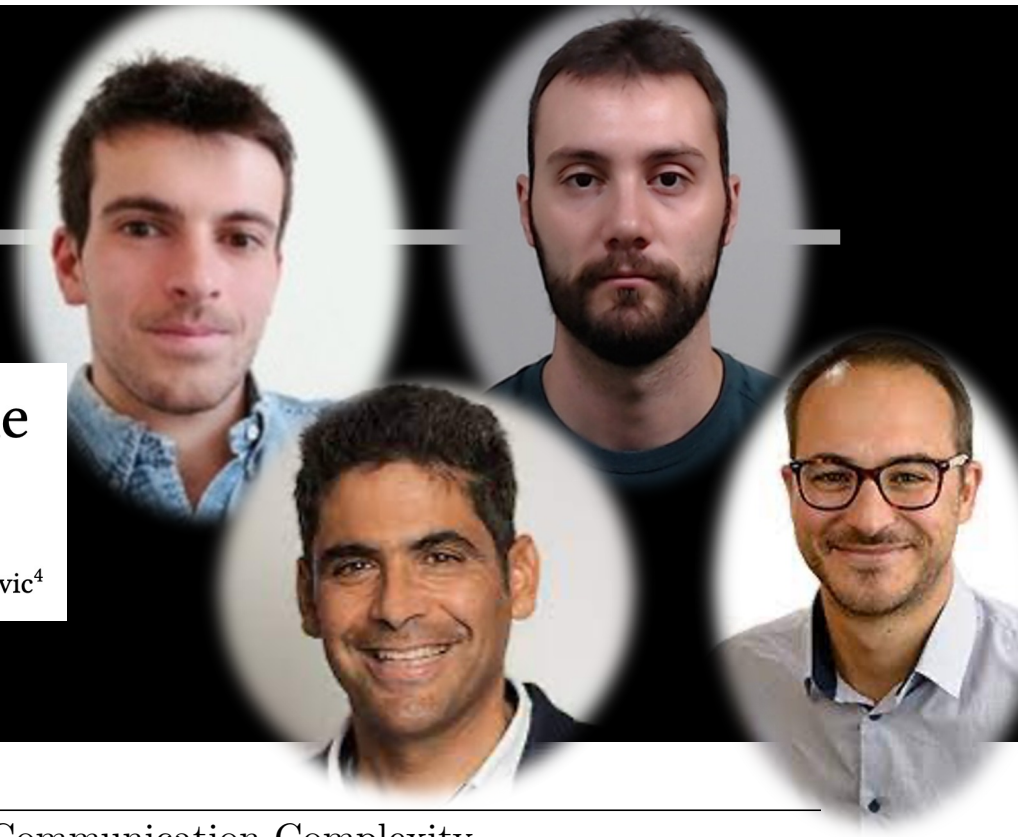
(IPDPS 2022)

Of note:

- Makes any agreement protocol accountable.
- When no accountability violation, minimal overhead: $O(n^2)$
- When accountability violate: $O(n^3)$ extra communication

Every agreement protocol must have communication Complexity $\Omega(n^2)$.

As easy as ABC…



As easy as ABC: Optimal (A)ccountable (B)yzantine (C)onsensus is easy!

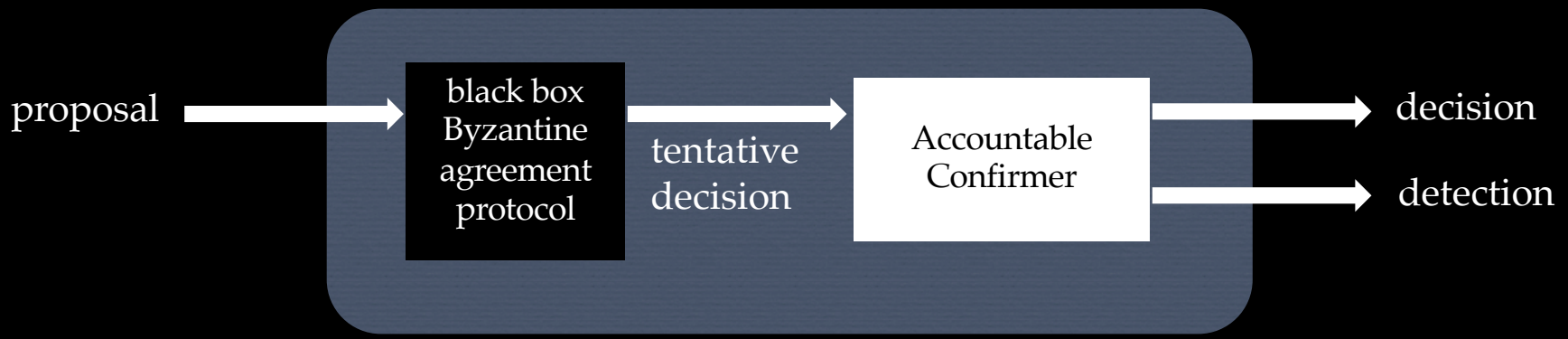
Pierre Civit[1], Seth Gilbert[2], Vincent Gramoli[3,4], Rachid Guerraoui[4] and Jovan Komatovic[4]

(IPDPS 2022)

Base Consensus Protocol	Communication Complexity of the Base Consensus Protocol	Communication Complexity of the Accountable Counterpart in the Good Case	Accountability Threshold
PBFT *	$O(n^4)$	$O(n^4)$	$2n/3$
HotStuff *	$O(n^3)$	$O(n^3)$	$2n/3$
DBFT/Polygraph	$O(n^4)$	$O(n^4)$	n
ABC	X	$O(X)$	n

* P. Sheng, G. Wang, K. Nayak, S. Kannan, P. Viswanath, BFT Protocol Forensics, in CCS 2021

ABC Architecture

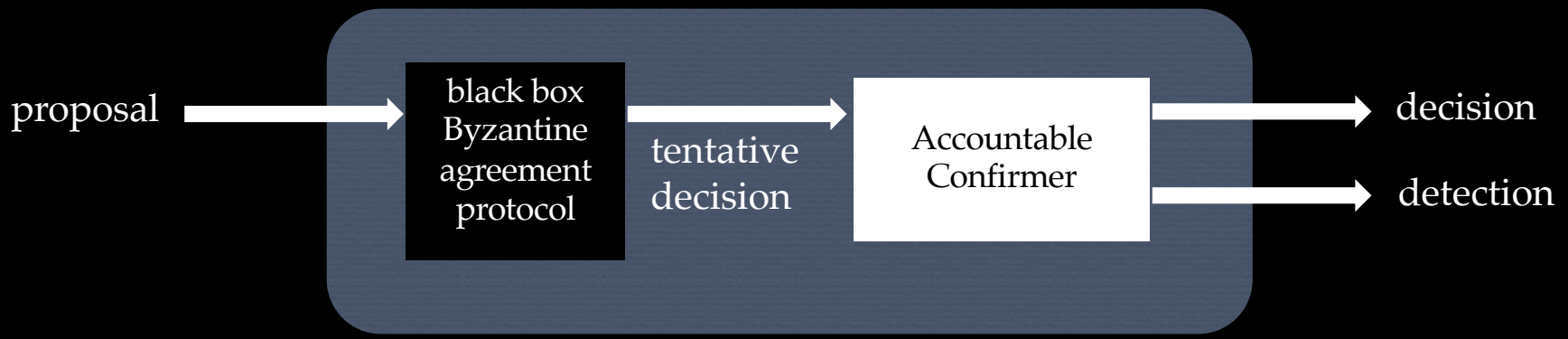


Accountable Confirmer:

- Terminating convergence: if $t < n/3$ and all honest processes submit the same tentative decision, then that value is output by all honest processes.
- Validity: honest processes only output a decision submitted by honest processes.
- Accountability: if 2 correct processes output different decisions, every process outputs proof of culpability for at least $n/3$ processes.

Note: if $> n/3$ malicious, or if agreement protocol fails,
then confirmer can output almost anything, or not terminate at all!

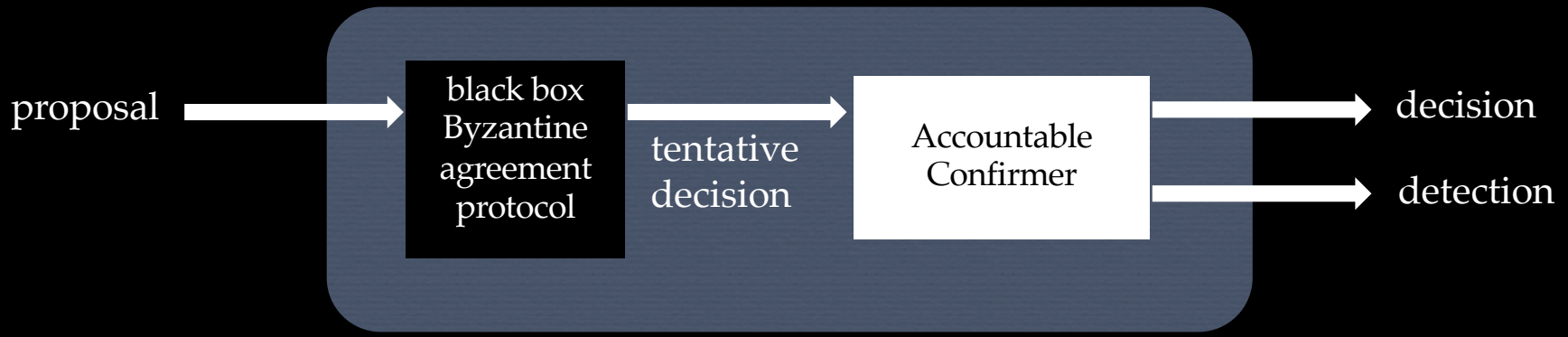
ABC Architecture



Accountable Agreement Analysis:

- Conditional Agreemeent: if $t < n/3$, black box returns same tentative decision to all, so confirmer outputs that decision.
- Conditional Validity: follows from validity of black box + validity confirmer.
- Conditional Termination: if $t < n/3$, black box terminates and returns same tentative decision to all, so confirmer terminates.
- Accountability: if two decisions differ, confirmer guarantees evidence of $n/3$ cheaters.

ABC Architecture

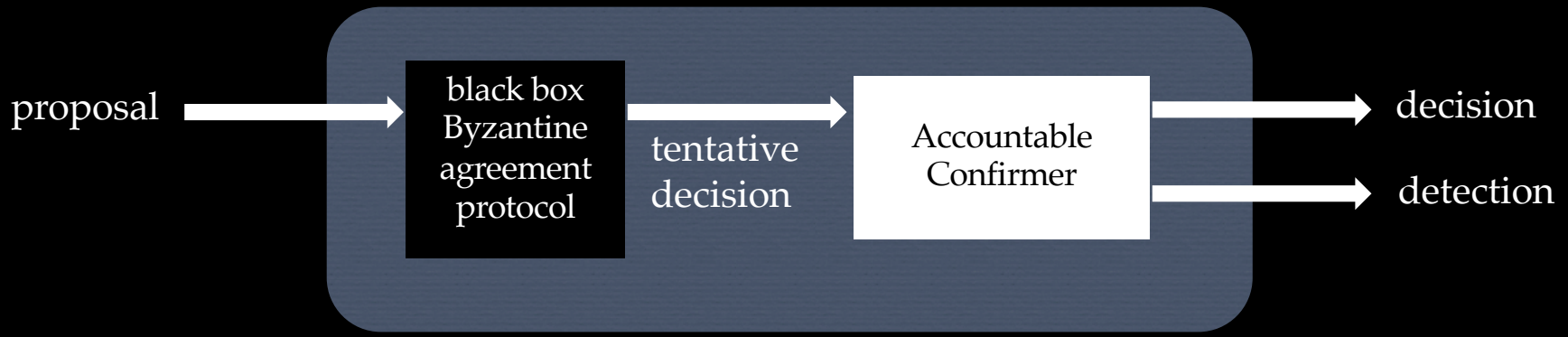


Simple Accountable Confirmer Implementation:

- Broadcast (signed) tentative decision to all.
- If $2n/3 + 1$ broadcast same decision, output that decision.
- Broadcast decision and $2n/3 + 1$ signed broadcast messages to all.
- If disagreement, then find intersection between $(2n/3+1)$ and $(2n/3+1)$ signed sets of size at least $n/3$.

Ignoring floors and ceilings…

ABC Architecture



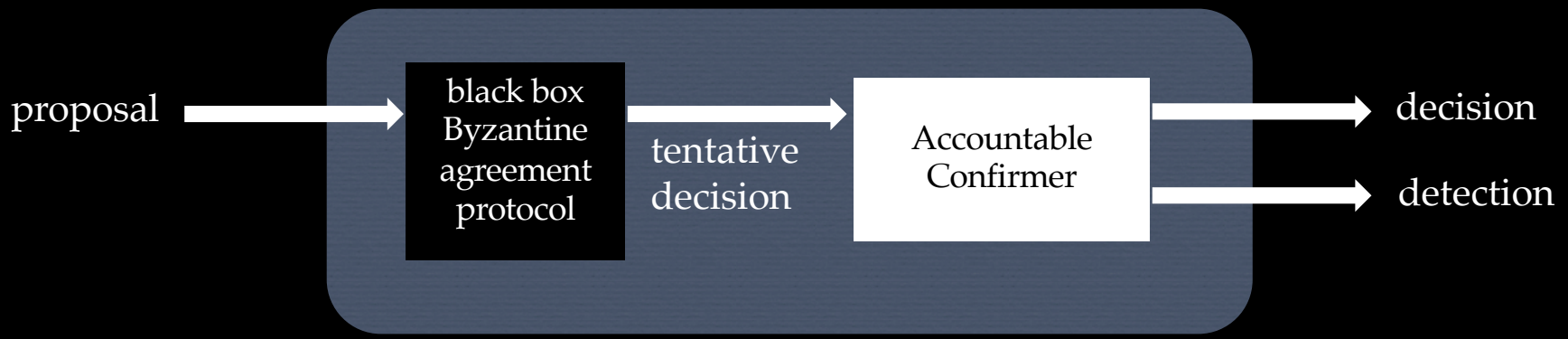
Simple Accountable Confirmer Implementation:

- Broadcast (signed) tentative decision to all.
- If 2n/3 + 1 broadcast same decision, output that decision.
- Broadcast decision and 2n/3 + 1 signed broadcast messages to all.
- If disagreement, then find intersection between (2n/3+1) and (2n/3+1) signed sets of size at least n/3.

Use threshold signatures to reduce communication complexity

Ignoring floors and ceilings…

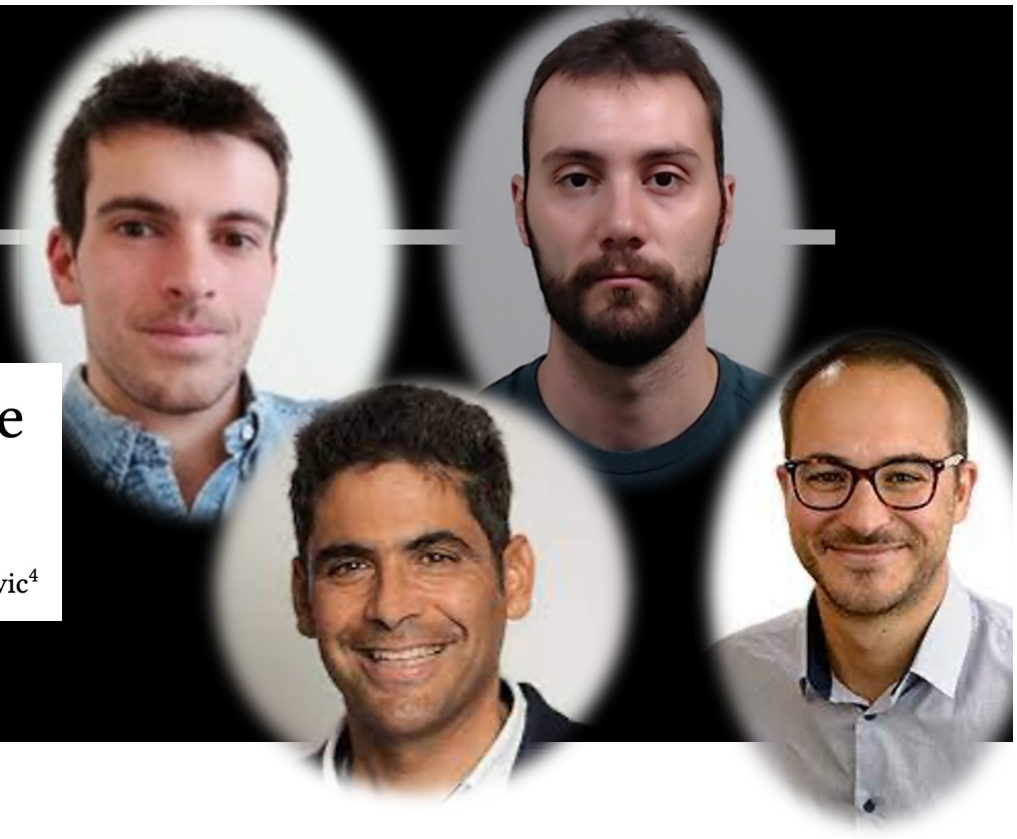
ABC Architecture



FAQ:

- How does the confirmer know who cheated inside the black box? It doesn't.
- What if everyone was honest during the confirmer, but not during the black box?
  Then the confirmer will output nothing!
- Can an honest process be accused because of a bad decision by the black box protocol (that was caused by cheating from others)?
  No, only processes that cheat (during the confirmer) are accused.

As easy as ABC…



As easy as ABC: Optimal (A)ccountable (B)yzantine (C)onsensus is easy!

Pierre Civit[1], Seth Gilbert[2], Vincent Gramoli[3,4], Rachid Guerraoui[4] and Jovan Komatovic[4]

(IPDPS 2022)

Of note:

- Makes any agreement protocol accountable.
- When no accountability violation, minimal overhead: $O(n^2)$
- When accountability violate: $O(n^3)$ extra communication

Every agreement protocol must have communication Complexity $\Omega(n^2)$.

A few more questions:

Can other distributed algorithms be accountable?

What type of attacks can we detect?

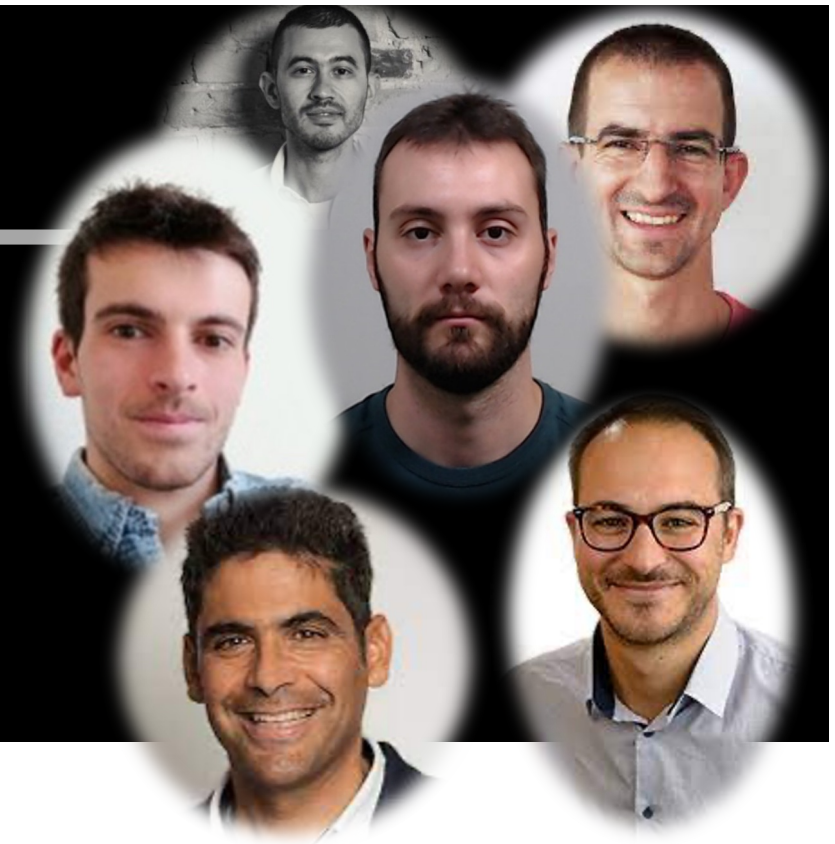
How can we design protocols that are easily made accountable?

Crime and Punishment…

Crime and Punishment in Distributed Byzantine Decision Tasks
(Extended Version)

Pierre Civit — *Sorbonne University, CNRS, LIP6*
Seth Gilbert — *NUS Singapore*
Vincent Gramoli — *University of Sydney*
Rachid Guerraoui — *Ecole Polytechnique Fédérale de Lausanne (EPFL)*
Jovan Komatovic — *Ecole Polytechnique Fédérale de Lausanne (EPFL)*
Zarko Milosevic — *Informal Systems*
Adi Seredinschi — *Informal Systems*

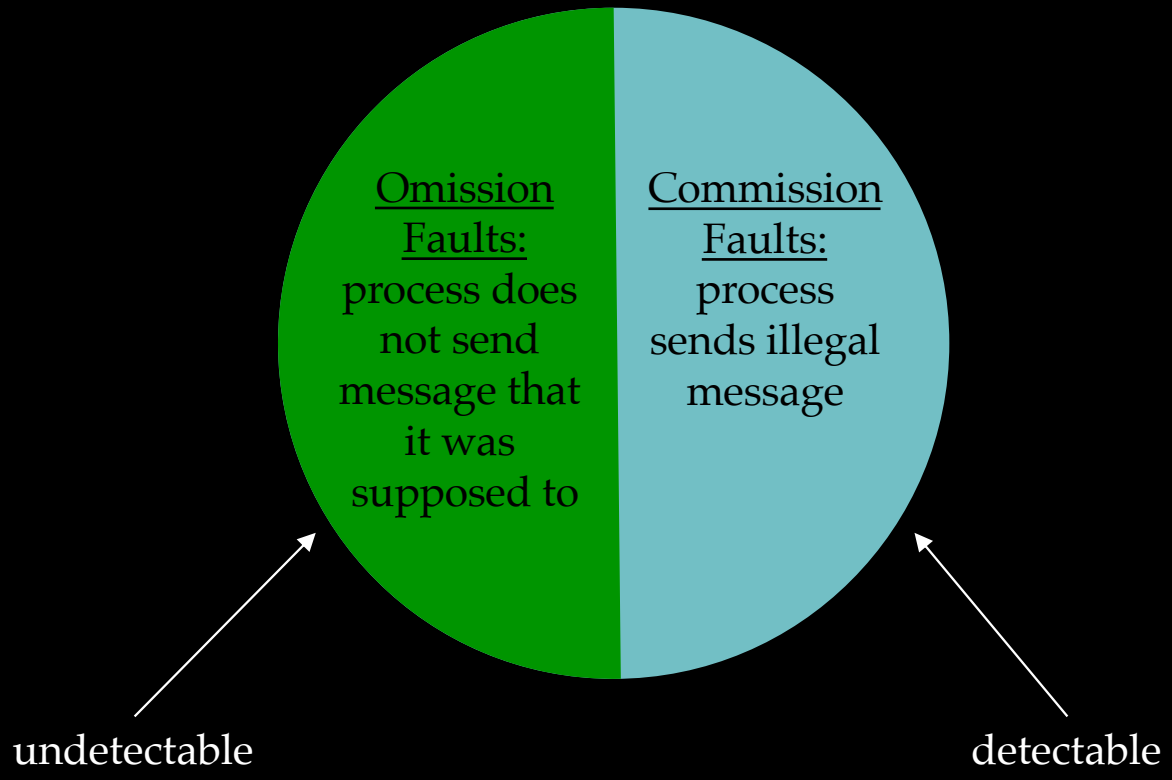
(ICDCS 2022)



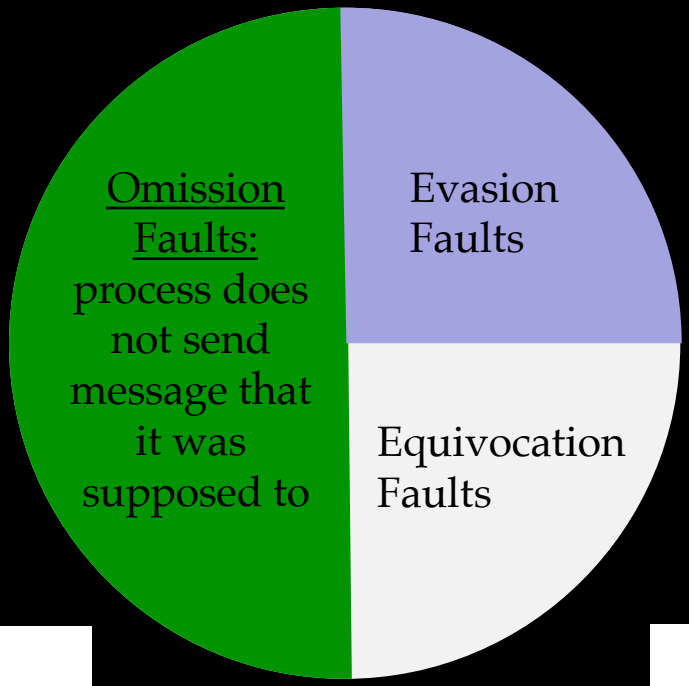
Of note:

- Necessary and sufficient conditions for accountability.
- Classification of failure types.
- Identification of key aspects for making a protocol accountable.

Types of failures (i.e., attacks)…



Types of failures (i.e., attacks)…



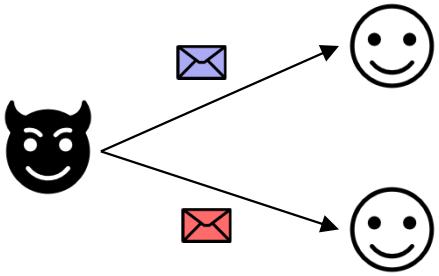
Evasion Faults:
Send a message without justification

Equivocation Faults:
Send two conflicting messages

Evasion faults are (provably) expensive to detect.

Equivocation faults are easy to detect.

Equivocation:



Crime and Punishment…

Detecting commission faults is necessary and sufficient.

Evasion faults are expensive to detect, while equivocation faults are cheap to detect.

Implications:

- Design a protocol where the *only* way to violate protocol safety is via malicious equivocation.
- Or: we can transform protocol into one where only equivocation faults cause problems, which can be detected.

What now?

Goal: Disincentivize Attacks

By detecting attacks, we can slash accounts, sue in court, or kick users out of the system.

Better disincentives:

- Can we detect attempted attacks instead of only successful attacks?
- Attack success may depend on network conditions (and random luck); the attacker may need to try many times before succeeding.
- Each attack exposes them to risk of detection…

Goal: Communication Efficiency

Accountability transformations add little overhead (especially in the good case where there is no attack).

Can we reduce the communication for accountability?

- No, not for deterministic protocols.
- And not under certain assumptions about what crypto can do.
- Maybe using randomization and new crypto (accountable signatures!).

Goal: Communication Efficiency

Byzantine agreement remains expensive

Can we trade fault-tolerance for robustness?

- What if protocol tolerates fewer failures…
- …but also uses accountability to catch the thiefs.
- Rely on disincentives rather than high levels of fault-tolerance.
- Less fault-tolerance ➔ more efficient protocol?

To Catch a Thief

Accountability is a useful and important property for distributed algorithms.

Accountable protocols can disincentivize attacks and (perhaps) improve efficiency.

Accountable agreement is easy!

Our simple ABC transformation is efficient and low overhead.

Many exciting directions…

Detect more attacks, game theoretic analysis, trade robustness for performance, reduce communication complexity, …