On the origins of blockchains

technology motivates systems require principles

Sergio Rajsbaum
Instituto de Matemáticas
UNAM, Mexico

École Polytechnique
22 October 2023

What are blockchains?

Musée des Arts et Métiers in Paris

Exhibit of 2500 objects

- Since its opened its doors in 1802, has been in the same building from 1060 by King Henry I

Many of the objects are computing objects

- the first mechanical calculator

Pascal's Pascaline
1652

Some are crypto objects

Automatic cryptograph
Alexis Kohl, 1889

Many others are communication objects

Théâtrophone
1889

By 1911 Proust had a pair of wires trailing into a headset to hear live music.

What about interaction of objects ?

computer +
communication =

What about interaction of objects ?

computer +
communication =

The blockchain creature is a kind of universal computing machine

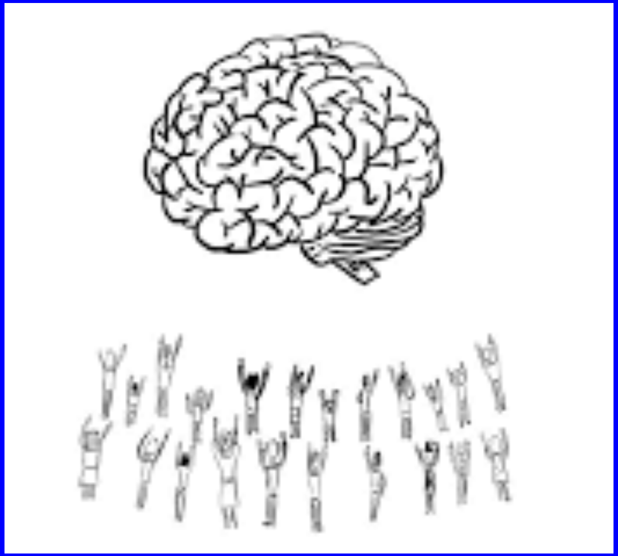
- Running on top of many computers
- Always accessible
- Un-killable

But accountable, temper-proof

But accountable, temper-proof

A distributed state machine

- Receives a sequence of commands
- Successively changes state
- returning a response to each one



The history of executed commands can be examined by anybody

And it is temper-proof

How to build it ?!

And control it, understand it…

a tale of the past 60 years!

How to build it ?!

And control it, understand it…

starting in the early 1960's

Distributed
computing
≠
sequential computing

A few milestones

focusing on the origins, DC biased

A few milestones

focusing on the origins, DC biased

technological

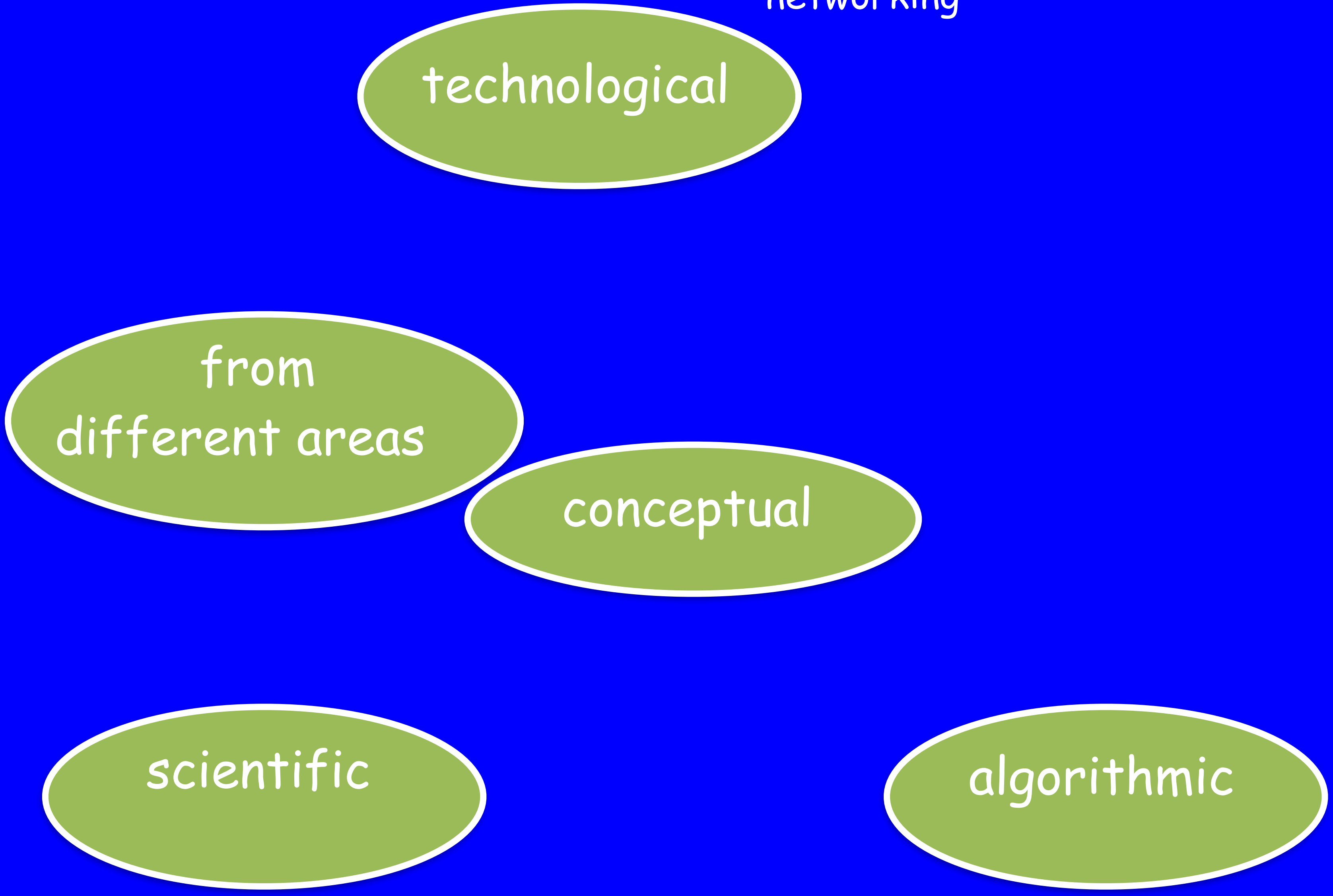
from
different areas

conceptual

scientific

algorithmic

- multicore, concurrency
- networking



- multicore, concurrency
- networking

technological

- cypto
- databases
- distributed computing

from
different areas

conceptual

scientific

algorithmic

- multicore, concurrency
- networking

technological

- cypto
- databases
- distributed computing

- transactions
- consensus
- signatures
- synchronous vs asynchronous

from
different areas

conceptual

- FT consensus
- leader election
- 2-phase commit
- efficient crypto

scientific

algorithmic

technological

- multicore, concurrency
- networking

from different areas

- cypto
- databases
- distributed computing

conceptual

- transactions
- consensus
- signatures
- synchronous vs asynchronous

algorithmic

- FT consensus
- leader election
- 2-phase commit
- efficient crypto

scientific

- how many faults
- impossibilities
- topology

A few milestones

starting when ??
- still, some of the main notions go back many years

New Kingdon XVIII Dynasty
https://www.sciencephoto.com/media/140538/view/ancient-egyptian-scribes

Ledger from 1828 in Germany

electromechanical calculator 1927
https://www.technikum29.de/en/computer/electro-mechanical.php

A few milestones

starting when ??
- still, some of the main notions go back many years

New Kingdon XVIII Dynasty
https://www.sciencephoto.com/media/140538/view/ancient-egyptian-scribes

- Notaries: 2500 B.C. in Ancient Egypt
- Ledgers: 1500 or earlier in churches
- Fault tolerance: 1940s, 50s, 60s

Ledger from 1828 in Germany

electromechanical calculator 1927
https://www.technikum29.de/en/computer/electro-mechanical.php

A few milestones

starting when ??
- still, some of the main notions go back many years

New Kingdon XVIII Dynasty
https://www.sciencephoto.com/media/140538/view/ancient-egyptian-scribes

- Notaries: 2500 B.C. in Ancient Egypt
- Ledgers: 1500 or earlier in churches
- Fault tolerance: 1940s, 50s, 60s

Ledger from 1828 in Germany

The first computers made of relays and tubes, which were noted for a lack of reliability. Thus, a large effort was expended in the area of computer checking and self-repair.

Carter and Bouricius 1971

Namely: a backup computer waiting to run in case the main one fails

electromechanical calculator 1927
https://www.technikum29.de/en/computer/electro-mechanical.php

A few milestones (CS and Eng)

1961	concurrent computing	Atlas computer
1965	Mutual exclusion	Dijkstra
1971	wide-area packet-switched network	Arpanet
1974, 1975	Distributed databases: timestamps Central sequence generator	Johnson and Beeler Steve Bunch
1975	Impossibility of agreement	Stony Brook System by Akkoyunlu, Ekanadham, Huber
1976	Transactions and concurrency control	Eswaran, **Gray**, Lorie, Traiger
1976	Primary-Backup for fault-tolerance	Alsberg and Day
1976	Public key crypto	Diffie, Hellman
1978	Digital signatures	Rabin
1978	State machine replication	Lamport
1978	Byzantine agreement	SIFT aircraft control
1978	3n+1 processors are needed to tolerate n Byzantine faults, and consensus definition	Lamport, Pease, Shostak
1979	Merkle trees	Ralph Merkle
1982	Consensus synchronous lower bound	Fischer, Lynch
1983	Consensus impossibility: crashes	Fischer, Lynch, Paterson
1983	Approximate agreement	Dolev, Lynch et al
1990	Sharing Memory Robustly in Message-Passing Systems	Attiya, Bar Noy, Dolev
1990	crypto timestaps,	Haber, Stornetta
1993	Topology	Herlihy, Shavit et al
2008	Bitcoin, blockchain	Nakamoto

1961 supercomputers: programs began to run concurrently

Atlas, the most powerful computer in the world 1960's-70s

1965 mutual exclusion

- By the end of the 1960s a crisis was emerging: programs were riddled with errors
- 1965 Dijkstra discovered mutual exclusion
- opened the way for the first books of principles on concurrent programming

1970s
origins of distributed databases

:

- First not resilient:
  - 1974 timestamping updates by the host that generates it and then applying in them in that order [Johnson and Beeler]
  - 1975 a central sequence generator [Steve Bunch]

- Resiliency:
  - 1976 with the primary-backup approach for resiliency by [Alsberg and Day]

1971 Packet switched networks

The world's first packet switched network, ARPANET

included FTP, Email, rlogin, and one of the first to implement the TCP/IP

Packet switched networks generated a great deal of work in distributed resource sharing

1970s
origins of distributed computing

1975 Design and implementation of the Stony Brook

[Akkoyunlu , Ekanadham, Huber]

- System aimed at building a flexible communication facility between processes

[1975 Akkoyunlu , Ekanadham, Huber]

Gangsters divided in two groups are about to pull off a big Job.

[1975 Akkoyunlu , Ekanadham, Huber]

Gangsters divided in two groups are about to pull off a big Job.

Some of the men are holed up in a warehouse across town, awaiting precise instructions.

[1975 Akkoyunlu , Ekanadham, Huber]

Gangsters divided in two groups are about to pull off a big Job.

Some of the men are holed up in a warehouse across town, awaiting precise instructions.

It is absolutely essential that the two groups act with complete reliance on each other in executing the plan.

[1975 Akkoyunlu , Ekanadham, Huber]

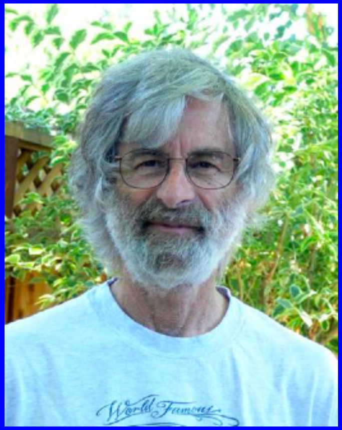
First Impossibility Result

Of course, they will never get around to putting the plan into action, because

... simultaneity cannot he achieved by this means.

[1975 Akkoyunlu , Ekanadham, Huber]

There was no clear understanding of how many faults could be tolerated



A sequence of papers by Lamport et al initiated the science of distributed computing

1978 “SIFT software implemented fault tolerance”

[Wensley, Lamport, Goldberg, Green, Levitt, Melliar-Smith, Shostak, Weinstock]

it was generally assumed that “tasks are redundantly executed by 3 computers, thus a single failure can be tolerated, using voting”

1978 first formal step

Reaching Agreement in the Presence of Faults [Lamport, Pease, Shostak]

Today " byzantine generals problem"
motivated by the SIFT project

Lamport, Pease, Shostak 1978

shows that "Byzantine" faults, can defeat any traditional 3-processor algorithm.

- 3n+1 processors are needed to tolerate n faults.

if digital signatures are used, 2n+1 processors are enough.

First abstractions: consensus, coordination
First impossibility results: are of a topological nature

- More generally distributed computing is of a topological nature in 1993

Coordination is needed all over

Whenever need to ensure that to actions happen or non

Back to [1975 Akkoyunlu , Ekanadham, Huber]

and its proof

Coordination is needed all over

- In computer networking, e.g. TCP can't guarantee state consistency between endpoints
- Transactions: if an automatic teller dispenses cash, then the account balance is debited and vice-versa
- A key concept in epistemic logic, common knowledge.
- generalizations provide a base of realistic expectations for our modern distributed systems.

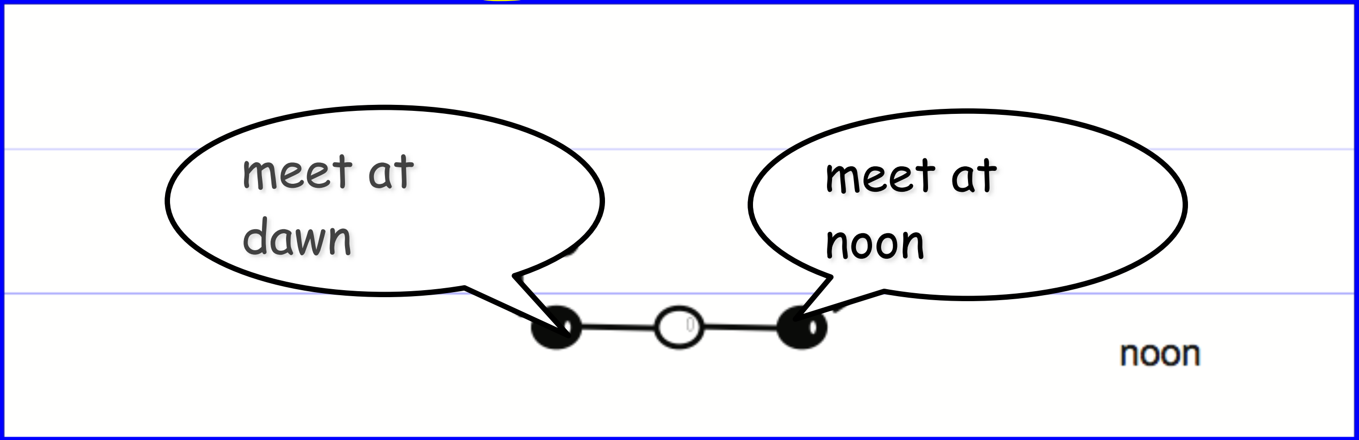
A basic illustration of the role of topology when computers interact

Impossibility of coordination and a basic illustration of the role of topology when computers interact

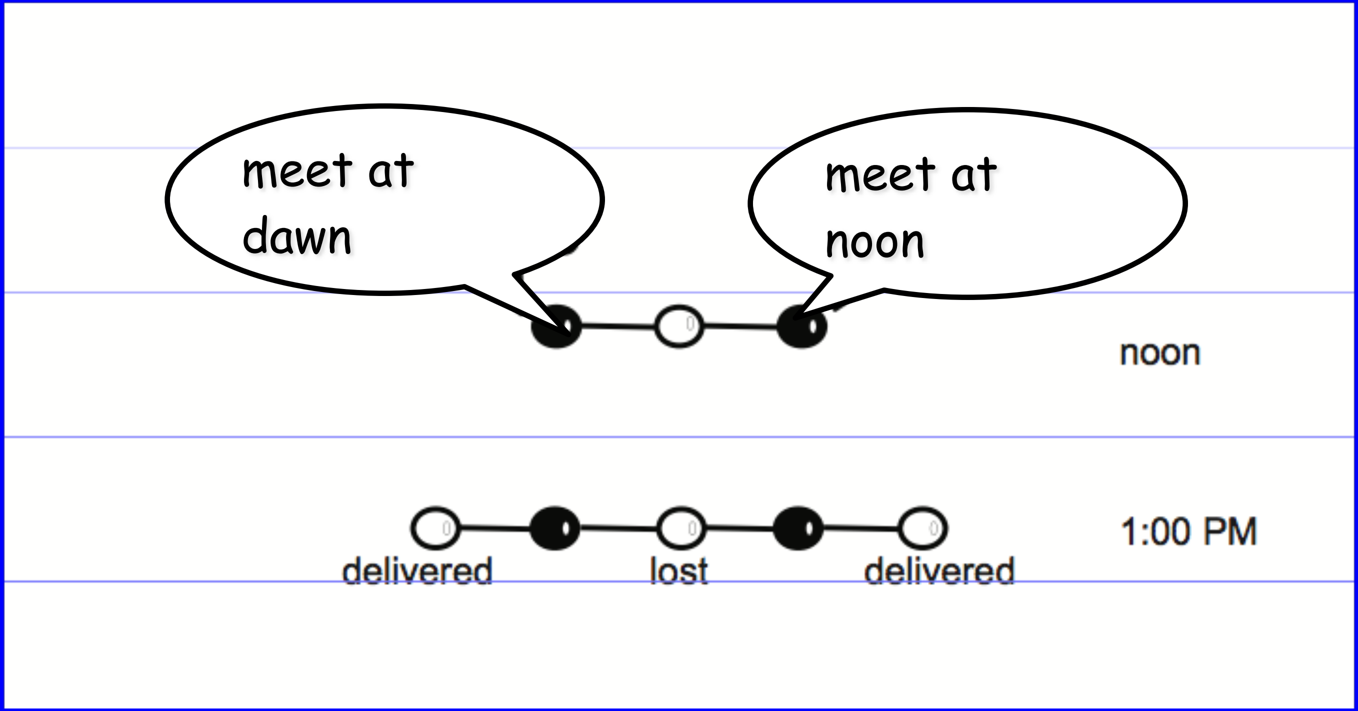
Alice and Bob want to schedule a meeting

If both attend, good, if only one attends, bad

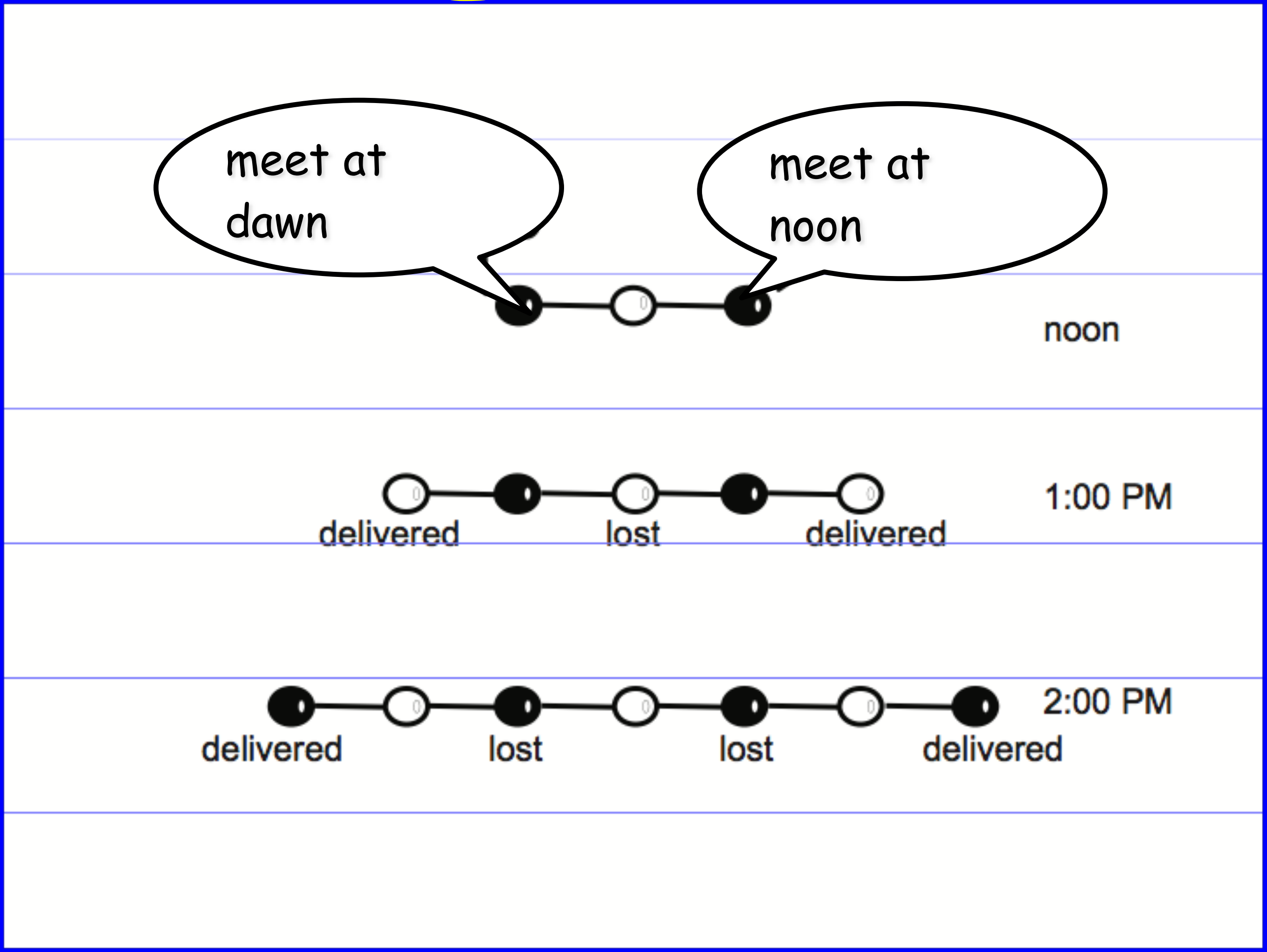
Evolution



Evolution


meet at dawn
meet at noon
noon
1:00 PM
delivered
lost
delivered

Evolution

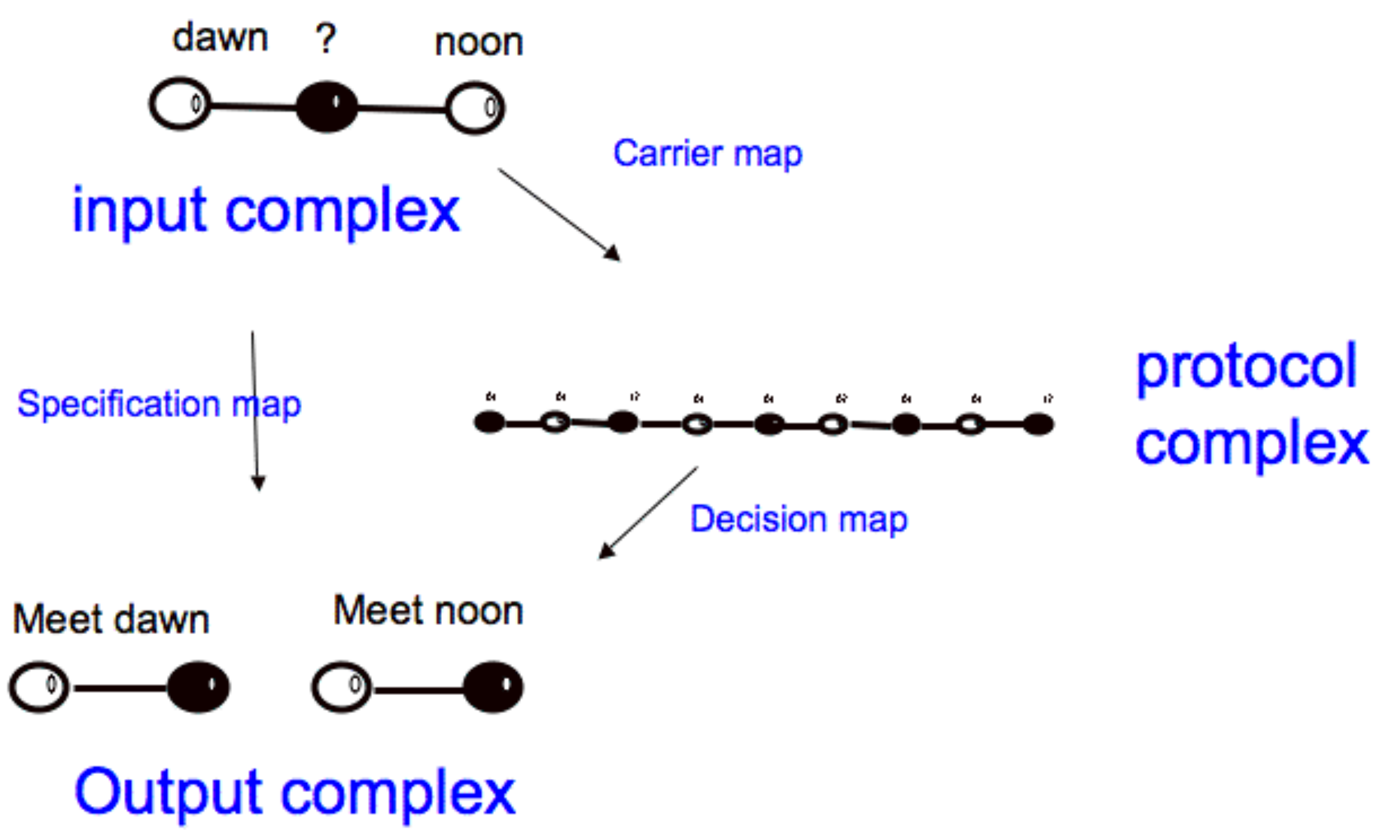


meet at dawn
meet at noon
noon
1:00 PM
delivered
lost
delivered
2:00 PM
delivered
lost
lost
delivered

Topology implies impossibility

No number of successfully delivered acks will be enough,

because the graph of possible states gets longer, but remains <u>connected</u>


dawn
?
noon
input complex
Carrier map
Specification map
protocol complex
Decision map
Meet dawn
Meet noon
Output complex

And then the generality

Herlihy, Shavit's Theorem 1993

opened the way to characterizations of the problems that are solvable in other models opened the way to characterizations of the problems that are solvable in other models

- t – crash resilient, Byzantine, dependent failures

opened the way to characterizations of the problems that are solvable in other models

- t – crash resilient, Byzantine, dependent failures
- Message passing and shared memory, including powerful shared memory objects such as test&set

opened the way to characterizations of the problems that are solvable in other models

- t – crash resilient, Byzantine, dependent failures
- Message passing and shared memory, including powerful shared memory objects such as test&set
- Synchronous and partially synchronous systems

opened the way to characterizations of the problems that are solvable in other models

- t – crash resilient, Byzantine, dependent failures
- Message passing and shared memory, including powerful shared memory objects such as test&set
- Synchronous and partially synchronous systems
- Distributed monitoring

opened the way to characterizations of the problems that are solvable in other models

- t – crash resilient, Byzantine, dependent failures
- Message passing and shared memory, including powerful shared memory objects such as test&set
- Synchronous and partially synchronous systems
- Distributed monitoring
- Robot algorithms

opened the way to characterizations of the problems that are solvable in other models

- t – crash resilient, Byzantine, dependent failures
- Message passing and shared memory, including powerful shared memory objects such as test&set
- Synchronous and partially synchronous systems
- Distributed monitoring
- Robot algorithms

And connection with formal methods: distributed specifications, epistemic logic and knowlegde

Foundation to the field

Foundation to the field

- multicore, concurrency
- networking

technological

- cypto
- databases
- distributed computing

from
different areas

- transactions
- consensus
- signatures
- synchronous vs asynchronous

conceptual

- FT consensus
- leader election
- 2-phase commit
- efficient crypto

scientific

- how many faults
- impossibilities
- topology

algorithmic

1993 main result of BG, HS, SZ :

1993 main result of BG, HS, SZ :

• Weakest form of interaction (wait-free) preserves a topological invariant:

1993 main result of BG, HS, SZ :

• Weakest form of interaction (wait-free) preserves a topological invariant:

• "computing preserves the initial topology, a contractible space remains contractible "

1993 main result of BG, HS, SZ :

- Weakest form of interaction (wait-free) preserves a topological invariant:

- "computing preserves the initial topology, a contractible space remains contractible "
  - No holes (of any dimension)

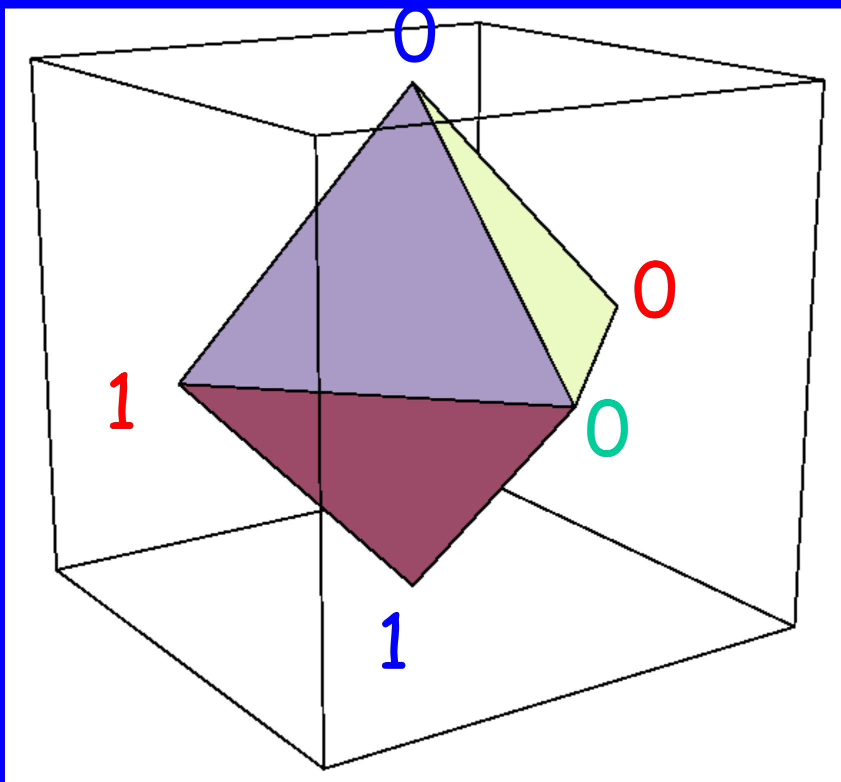
1993 main result of BG, HS, SZ :

- Weakest form of interaction (wait-free) preserves a topological invariant:

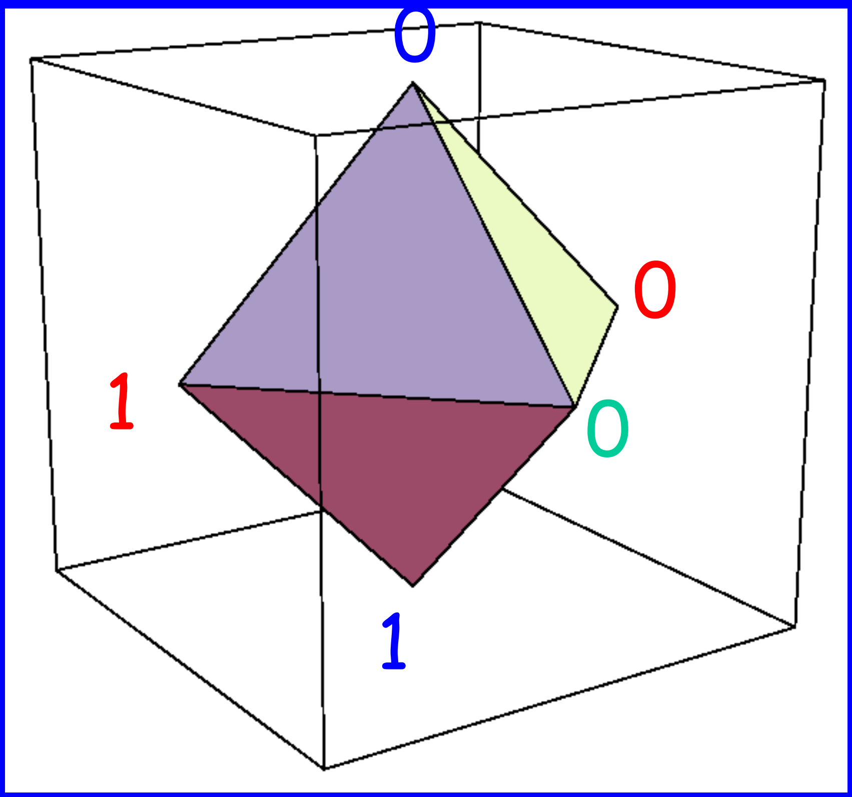
- "computing preserves the initial topology, a contractible space remains contractible "
  - No holes (of any dimension)

A scientific underlying topological framework

Initial states for binary consensus

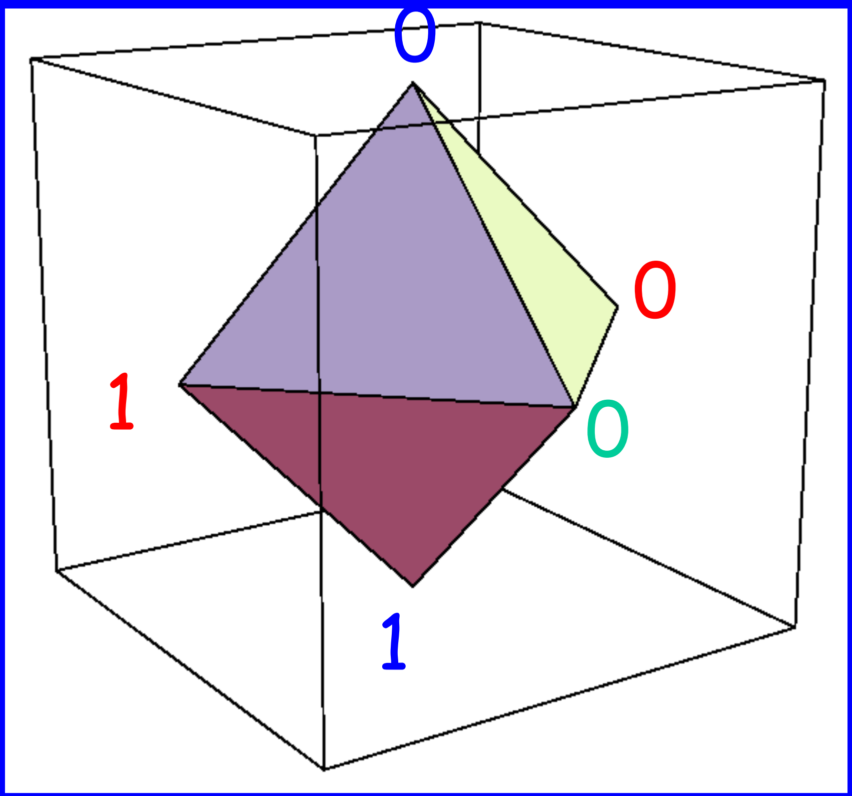


Initial states for binary consensus



Processes: blue, red, orange.

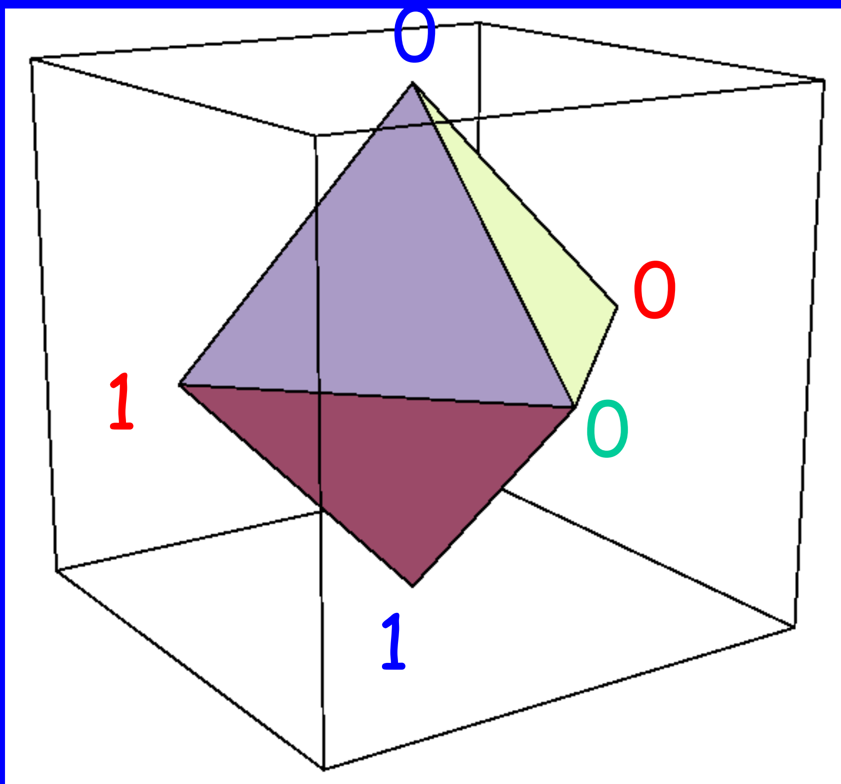
Initial states for binary consensus



Processes: blue, red, orange.

Independently assign 0 or 1

Initial states for binary consensus

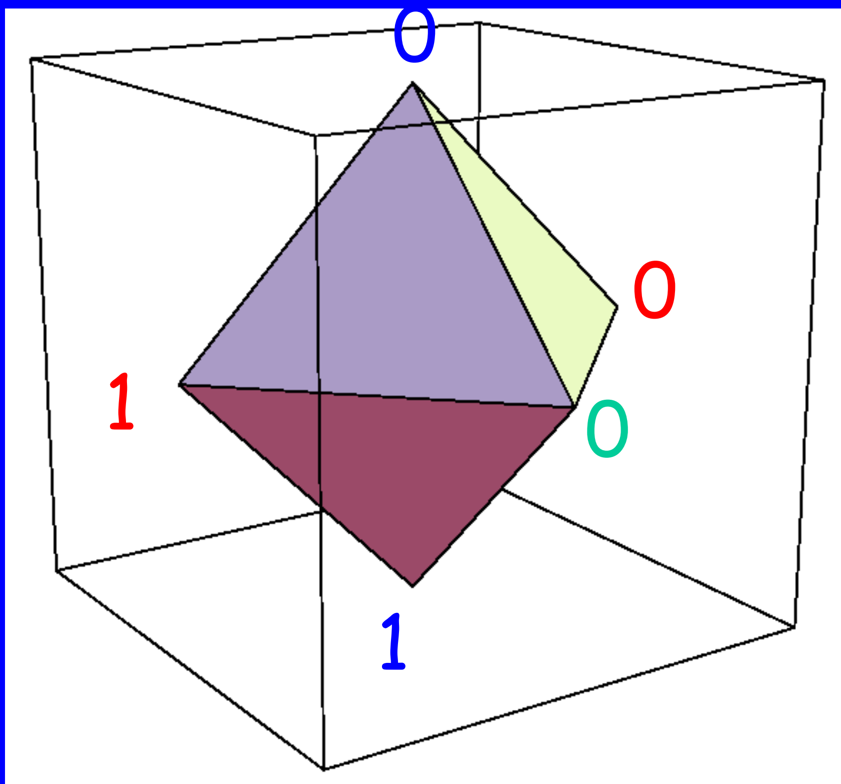


Processes: blue, red, orange.

Independently assign 0 or 1

Isomorphic to 2-sphere

Initial states for binary consensus



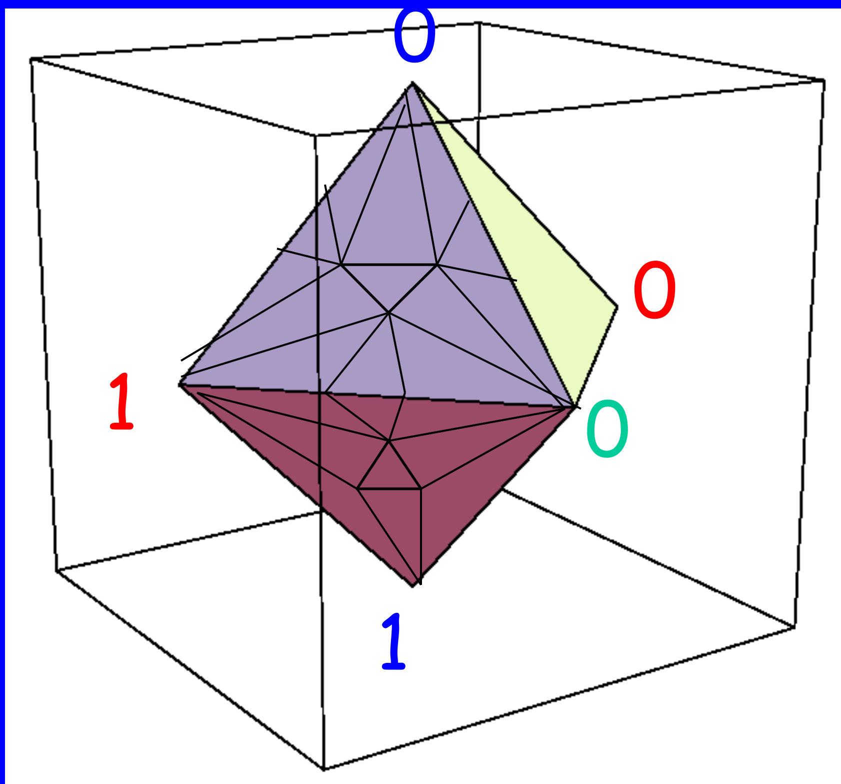
Processes: blue, red, orange.

Independently assign 0 or 1

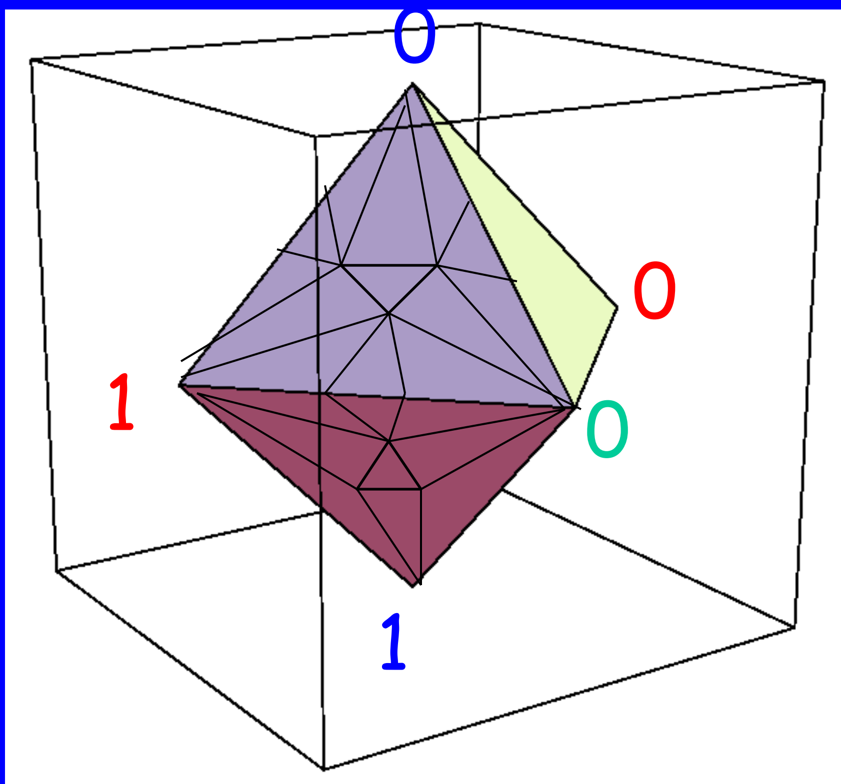
Isomorphic to 2-sphere

This is the input complex

States after 1 round, starting in the initial states for consensus

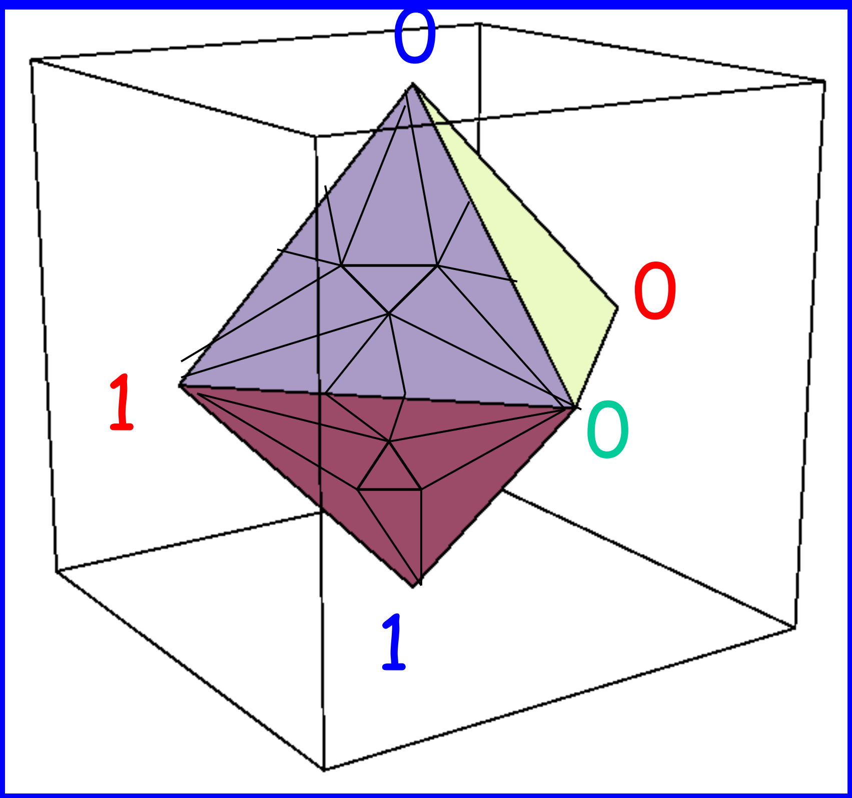


States after 1 round, starting in the initial states for consensus



Running an asynchronously the

States after 1 round, starting in the initial states for consensus



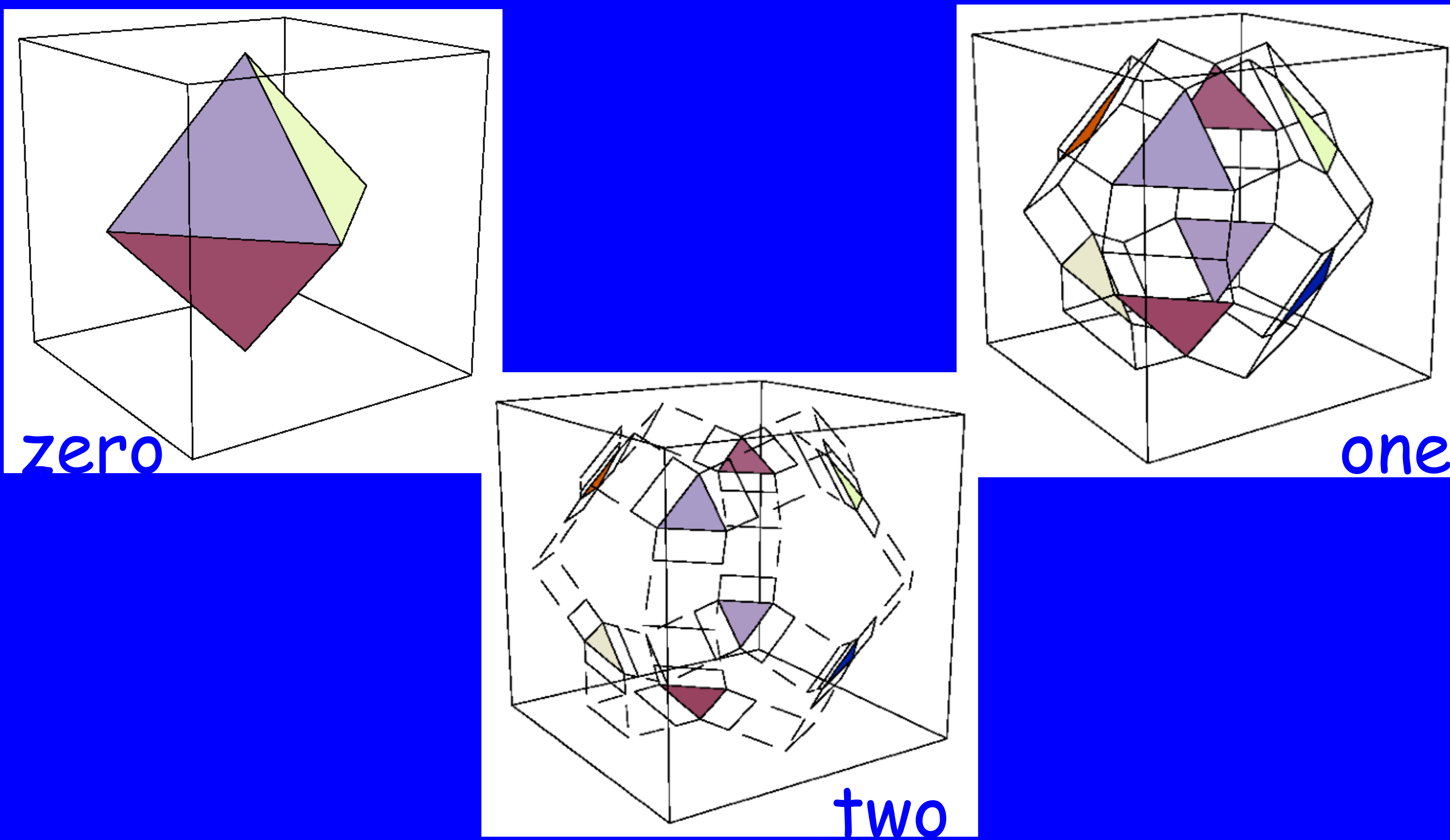
Running an asynchronously the topology of the input complex is preserved

Synchronous Model

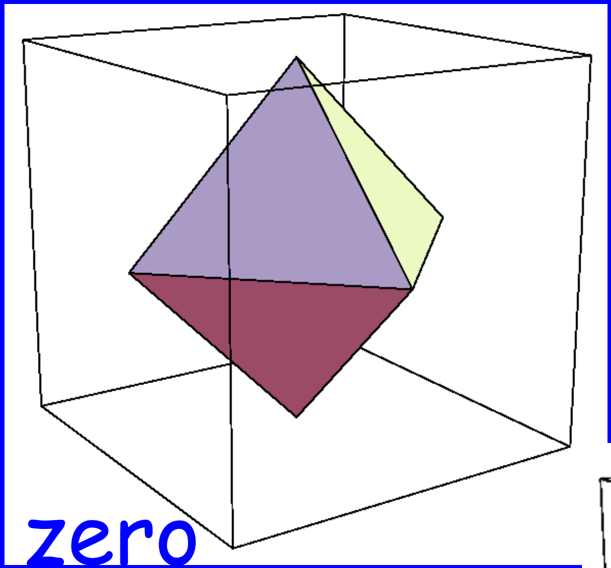
In t-resilient computation, t >1 there are holes, but do not change their type with the number of runs

In synchronous computation yes…

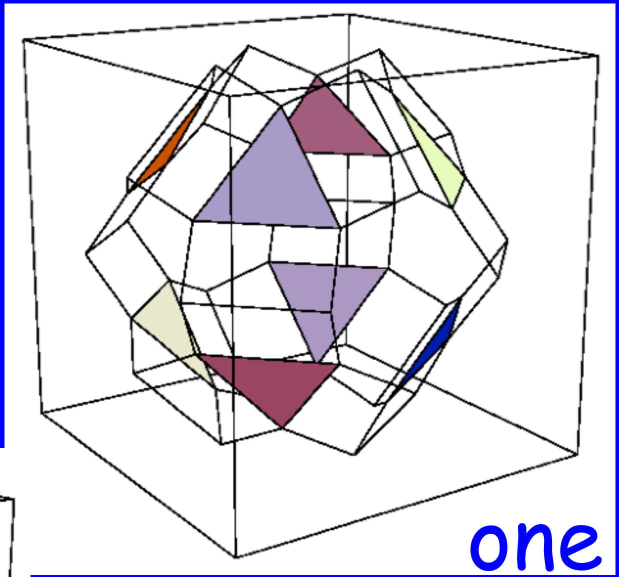
Synchronous protocol complex evolution

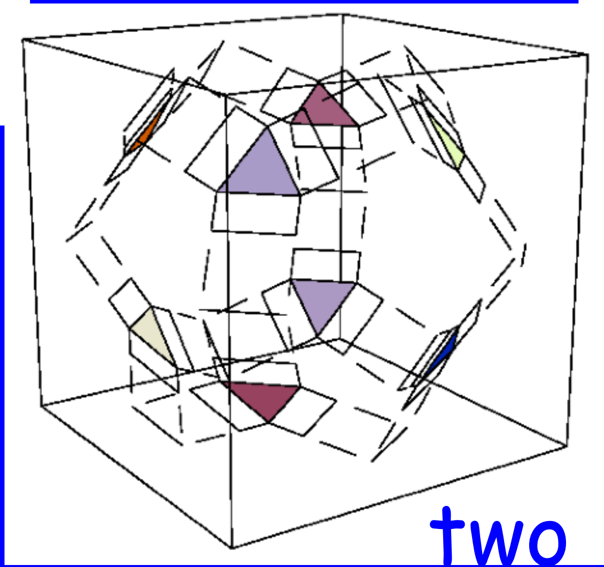


Synchronous protocol complex evolution

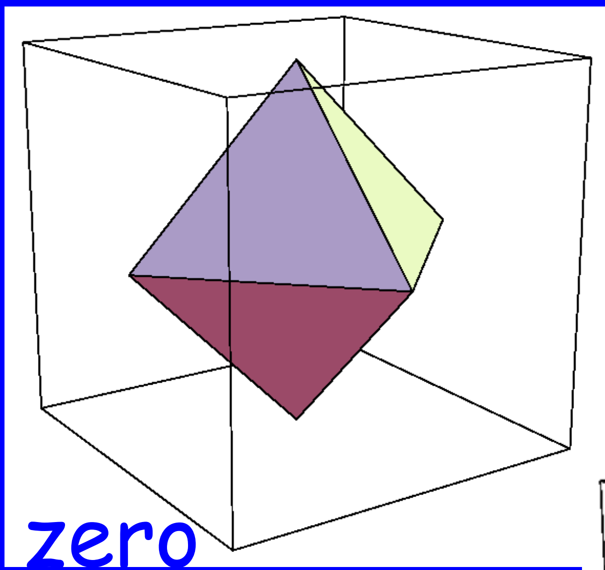


Connected but
not 1-connected

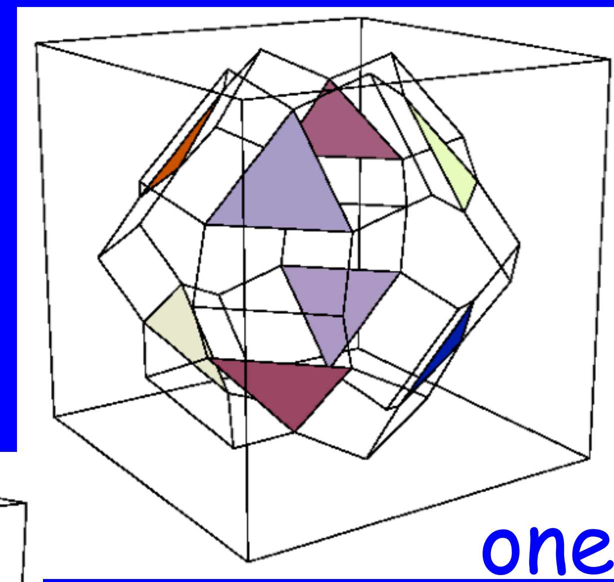




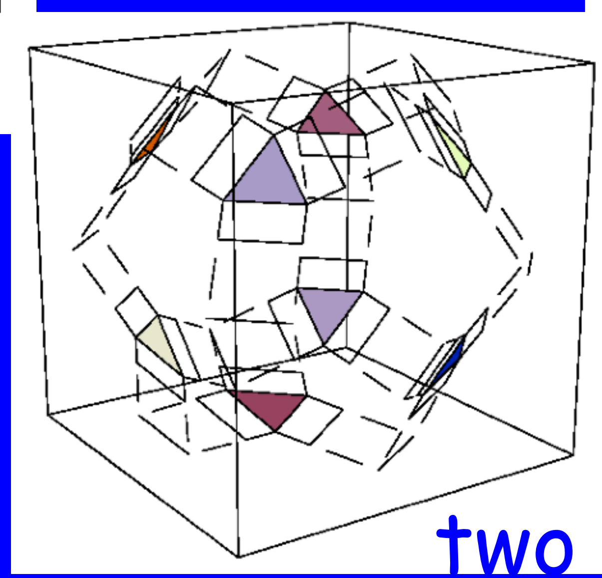
Synchronous protocol complex evolution



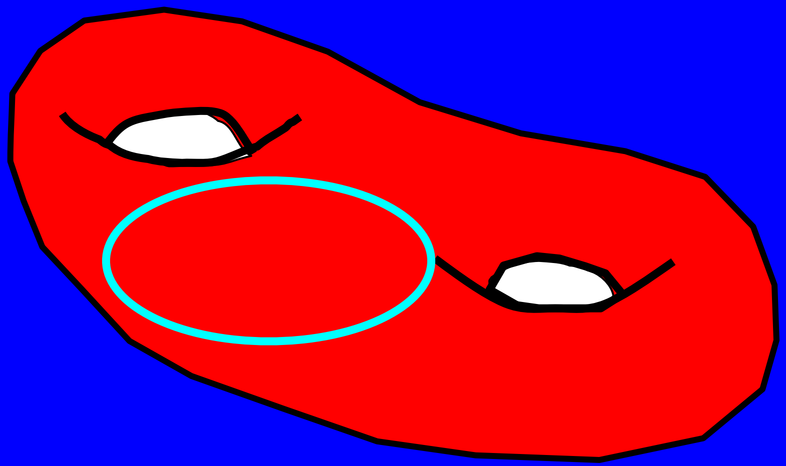
Connected but not 1-connected



Disconnected

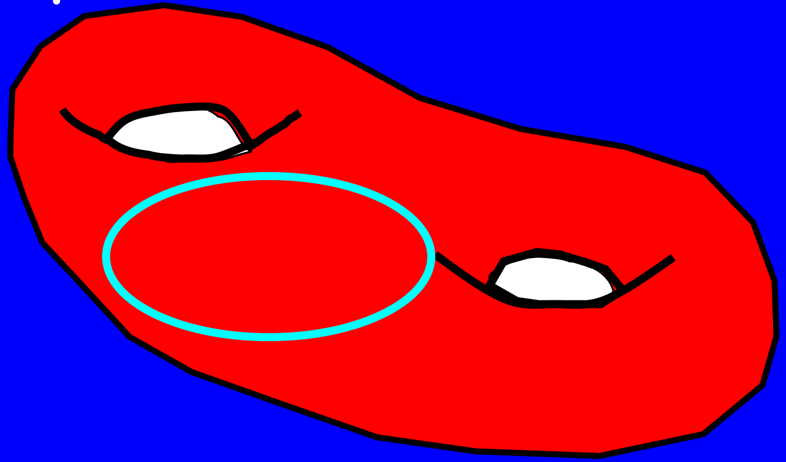


Problem solvability is undecidable



Problem solvability is undecidable

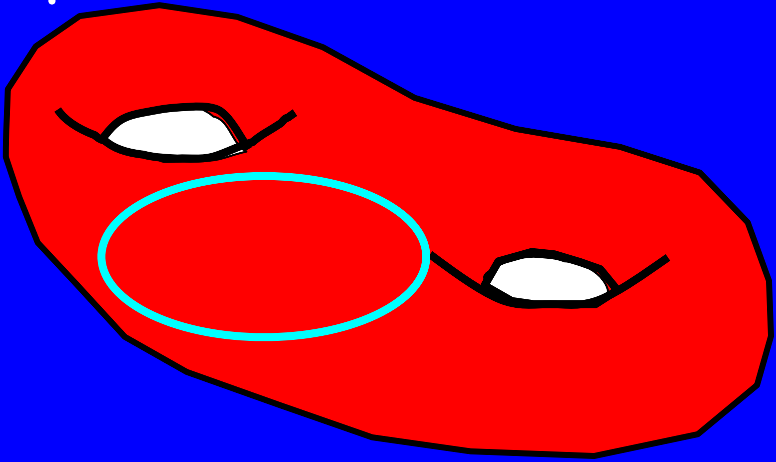
For a distributed computing model, is there an algorithm solving a given problem?



Problem solvability is undecidable

For a distributed computing model, is there an algorithm solving a given problem?

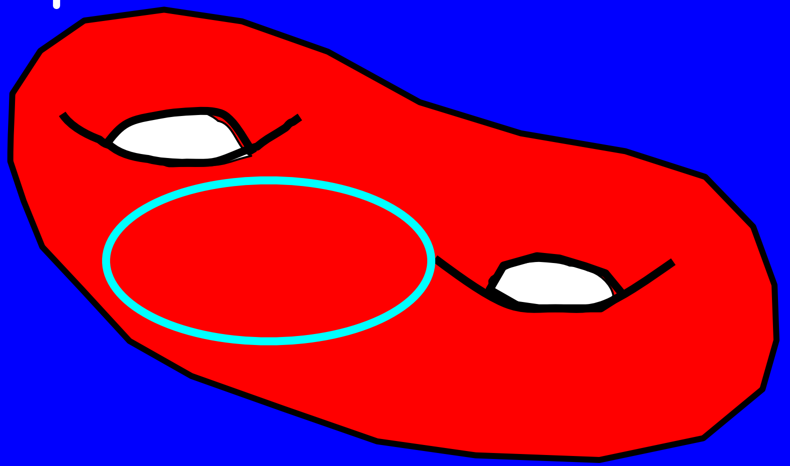
➢ Not in most models



Problem solvability is undecidable

For a distributed computing model, is there an algorithm solving a given problem?

- Not in most models

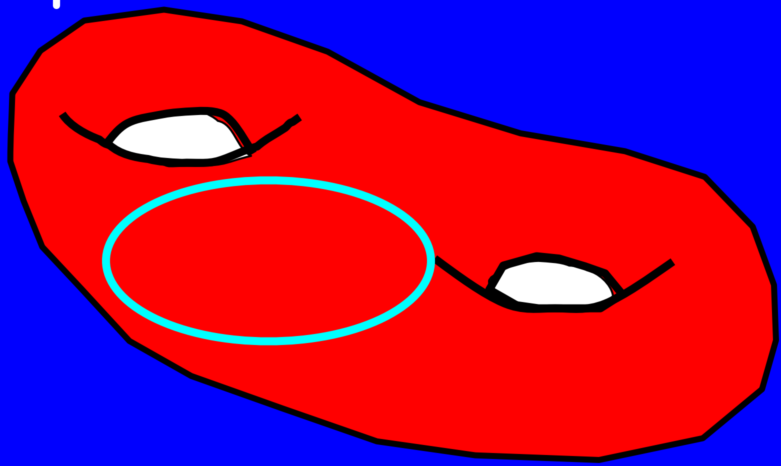


By reduction to a classic topology problem: can a given loop be contracted in a complex?

Problem solvability is undecidable

For a distributed computing model, is there an algorithm solving a given problem?

➢ Not in most models

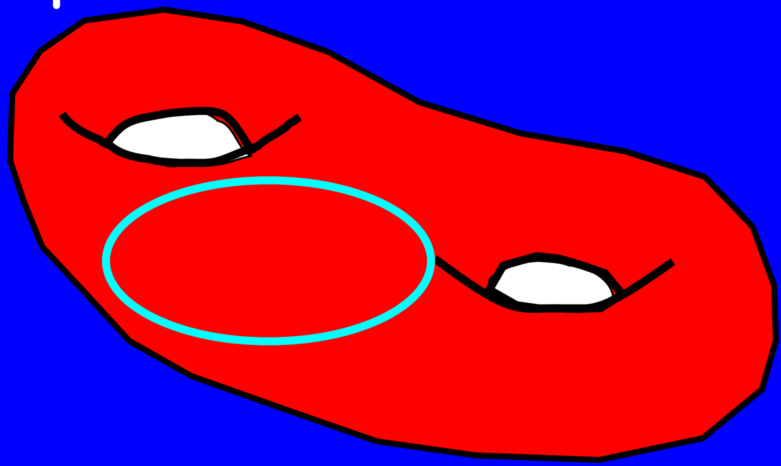


By reduction to a classic topology problem: can a given loop be contracted in a complex?

Problem solvability is undecidable

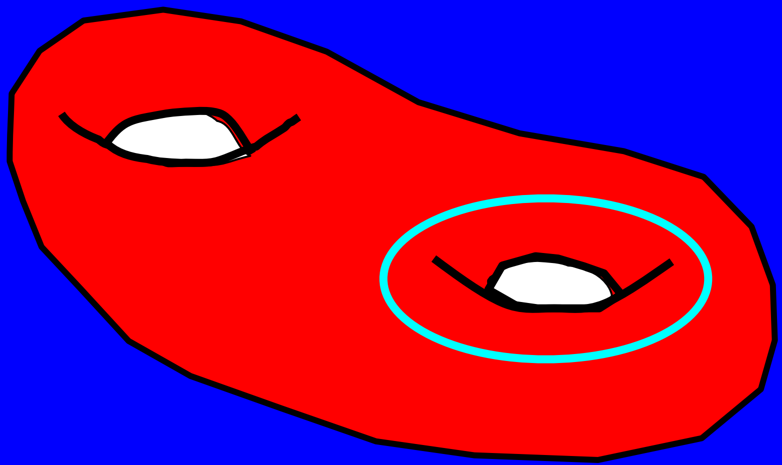
For a distributed computing model, is there an algorithm solving a given problem?

➢ Not in most models

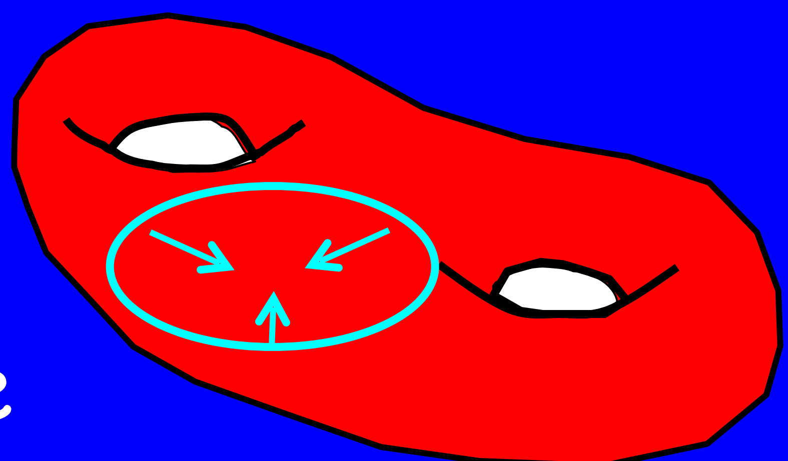


By reduction to a classic topology problem: can a given loop be contracted in a complex?

Contractibility is undecidable



not contractible



contractible

Conclusions

It all started in 1970's with…

It all started in 1970's with…

When it became clear that computers were going to be flying commercial aircraft,

It all started in 1970's with…

When it became clear that computers were going to be flying commercial aircraft,

NASA began funding research to figure out how to make them reliable enough for the task.

Part of that effort was the SIFT project at SRI.

Lamport


BOEING
737 MAX

Over the years, I often wondered whether the people who actually build airplanes know about the problem of Byzantine failures.

Over the years, I often wondered whether the people who actually build airplanes know about the problem of Byzantine failures.

… in the late 80s and early 90s, the people at Boeing working on military aircraft and on the space station, and the people at McDonnell-Douglas, did not understand the problem.

Over the years, I often wondered whether the people who actually build airplanes know about the problem of Byzantine failures.

… in the late 80s and early 90s, the people at Boeing working on military aircraft and on the space station, and the people at McDonnell-Douglas, did not understand the problem.

Lamport

We have come a long way since the time distributed systems were being built without understanding what exactly the problem being solved was, and which failures were tolerated

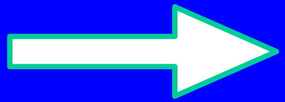
Distributed computing is different from sequential computing

It is a matter of perspectives,

of course

But perspectives can be complicated, they can evolve and they can depend on the environment

Rashomon, Kurosawa 1950



END

Thanks for your attention