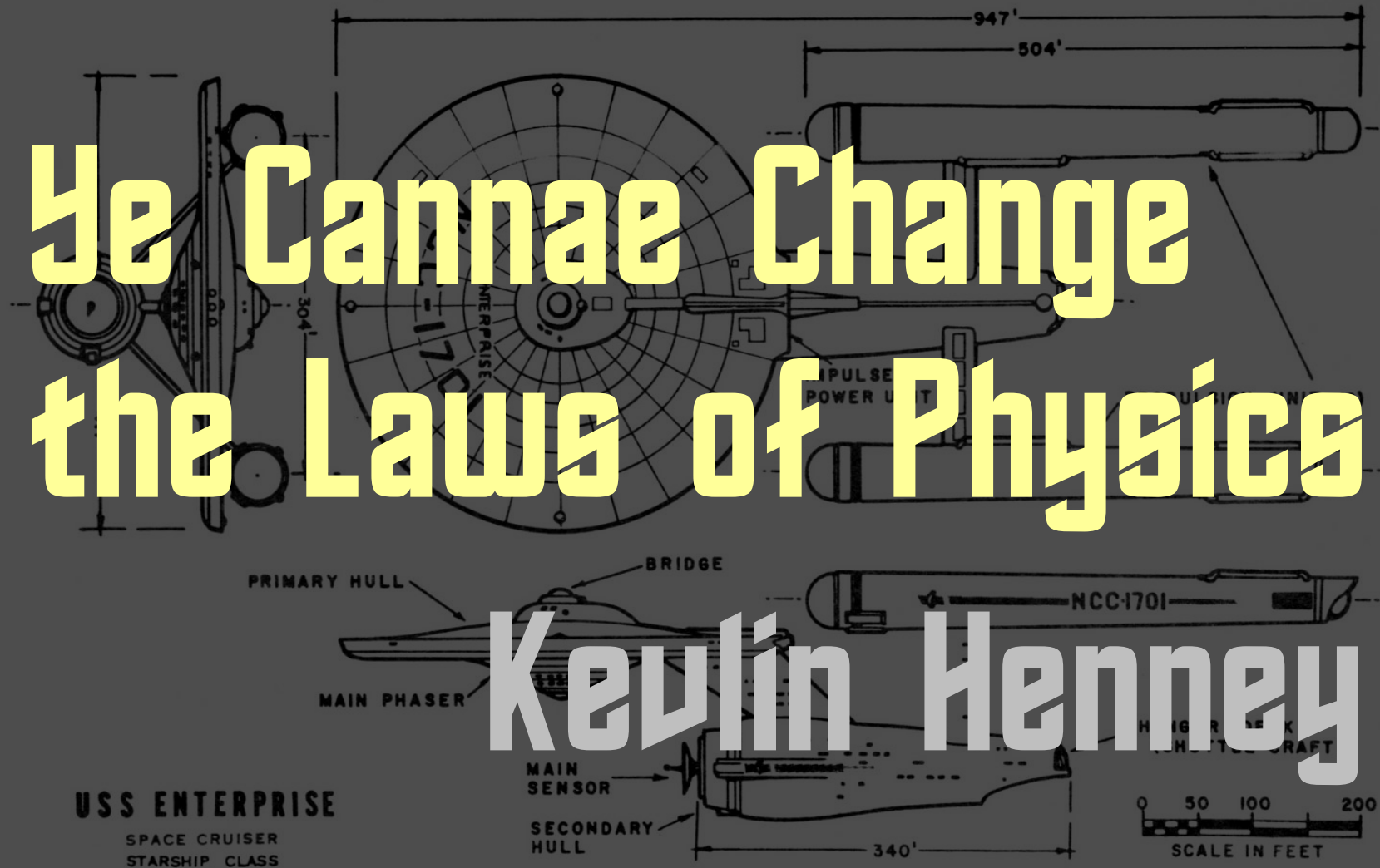
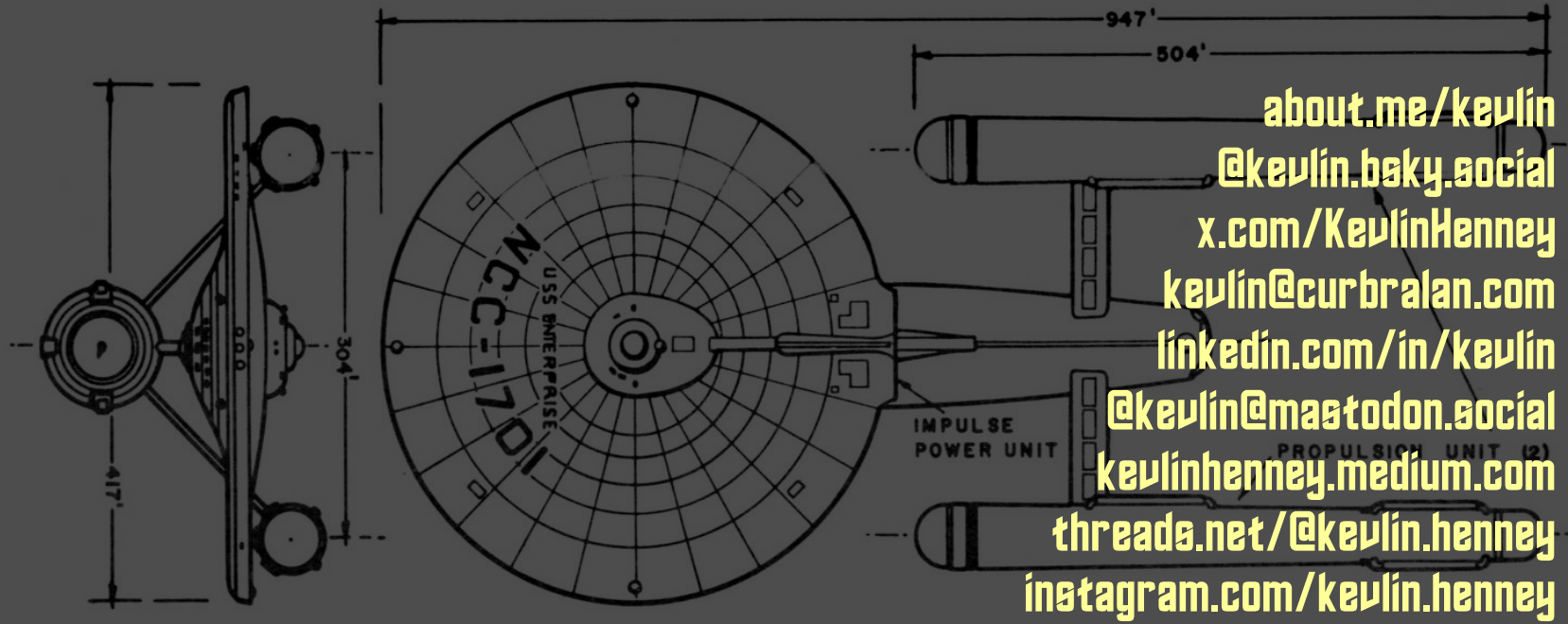


Ye Cannae Change the Laws of Physics

Kevlin Henney





about.me/kevin
[@kevin.bsky.social](https://x.com/KevlinHenney)
x.com/KevlinHenney
kevin@curbralan.com
linkedin.com/in/kevin
@kevin@mastodon.social
kevinhenney.medium.com
threads.net/@kevin.henney
instagram.com/kevin.henney

Kevlin Henney

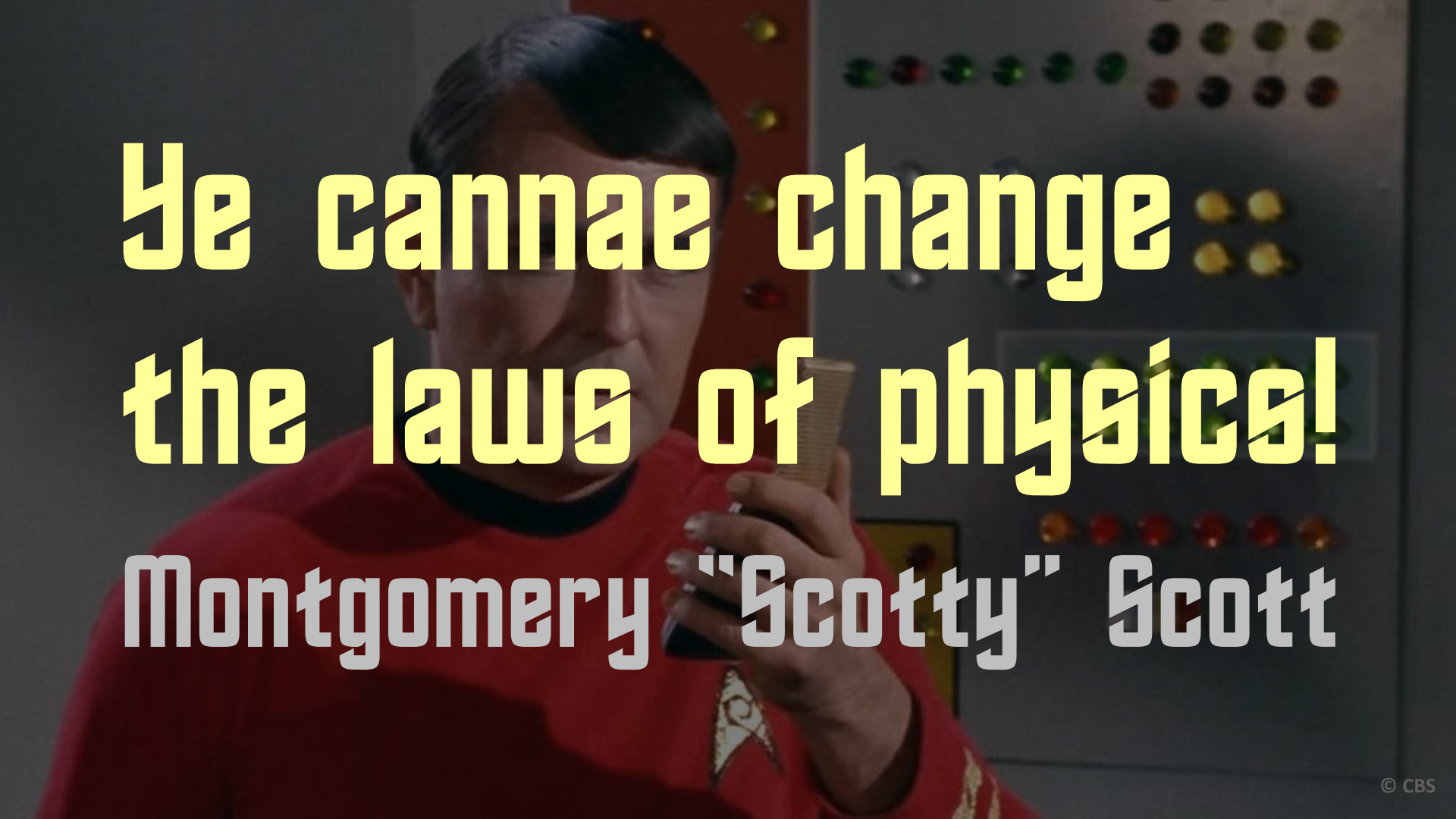
USS ENTERPRISE

SPACE CRUISER
STARSHIP CLASS

*about.me/kevin
@kevin.bsky.social
x.com/KevinHenney
kevin@curbralan.com
linkedin.com/in/kevin
@kevin@mastodon.social
kevinhenney.medium.com
threads.net/@kevin.henney
instagram.com/kevin.henney*

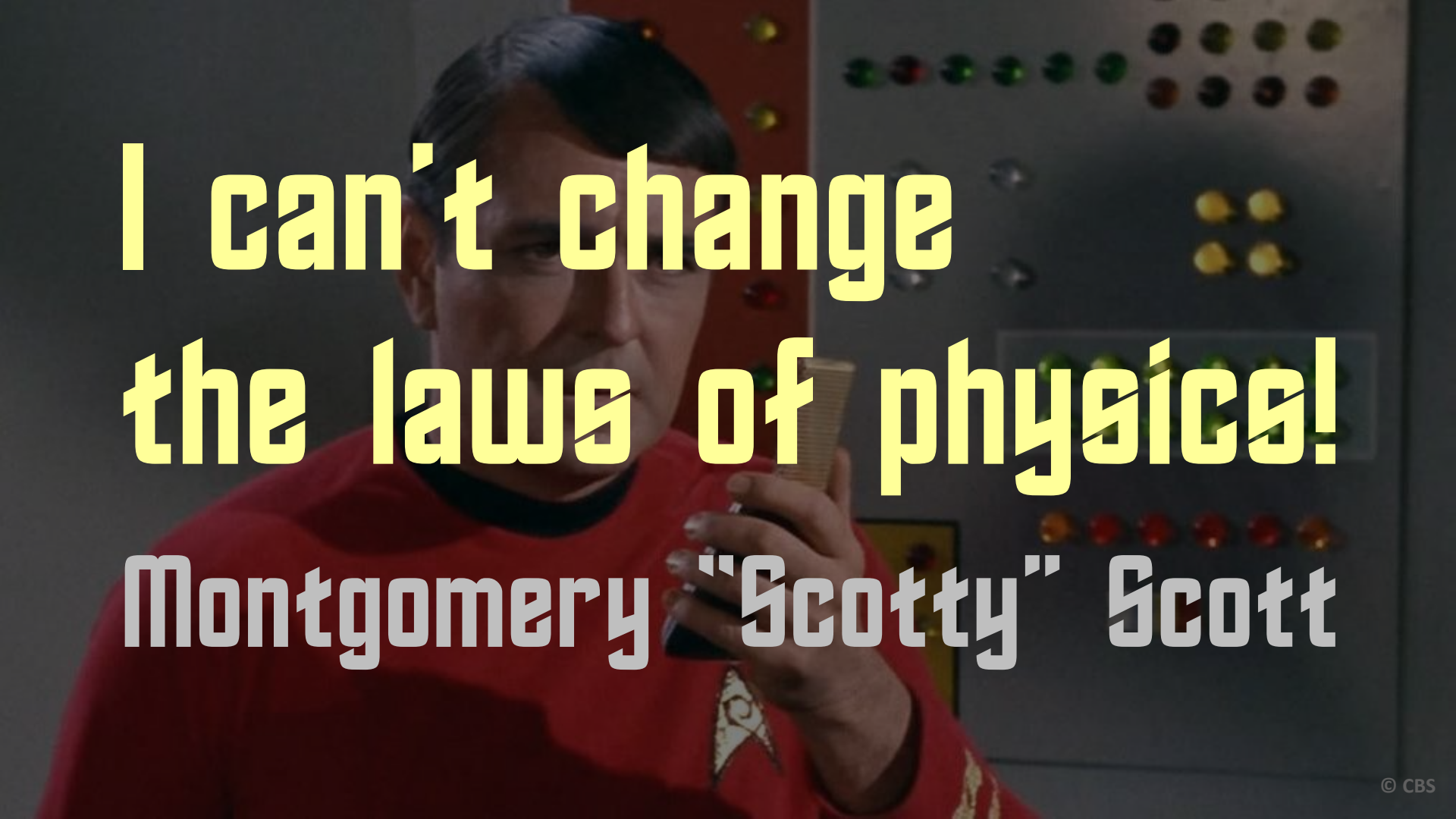




A man in a red Star Trek uniform, identified as Montgomery Scott, is shown from the chest up. He is holding a small, rectangular object in his hands and looking down at it with a focused expression. The background is a dark, metallic control panel with several rows of small, round buttons or lights in various colors (green, yellow, red).

**Ye cannae change
the laws of physics!**

Montgomery "Scotty" Scott

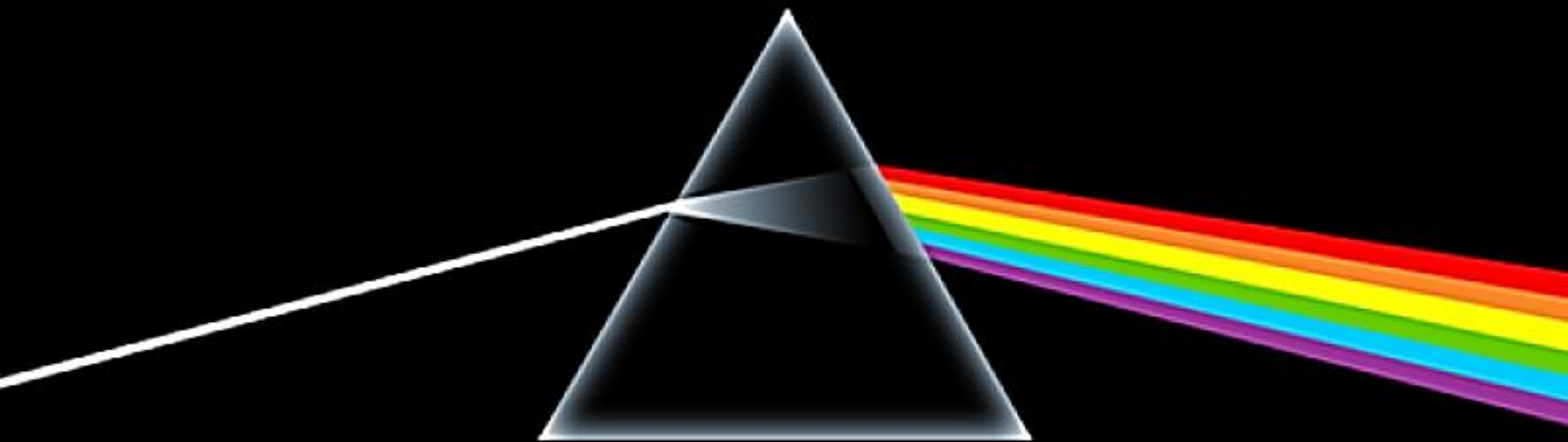
A man in a red Star Trek uniform, identified as Montgomery Scott, is shown from the chest up. He is holding a small, rectangular object in his right hand. The background is a dark control panel with various colored buttons and lights.

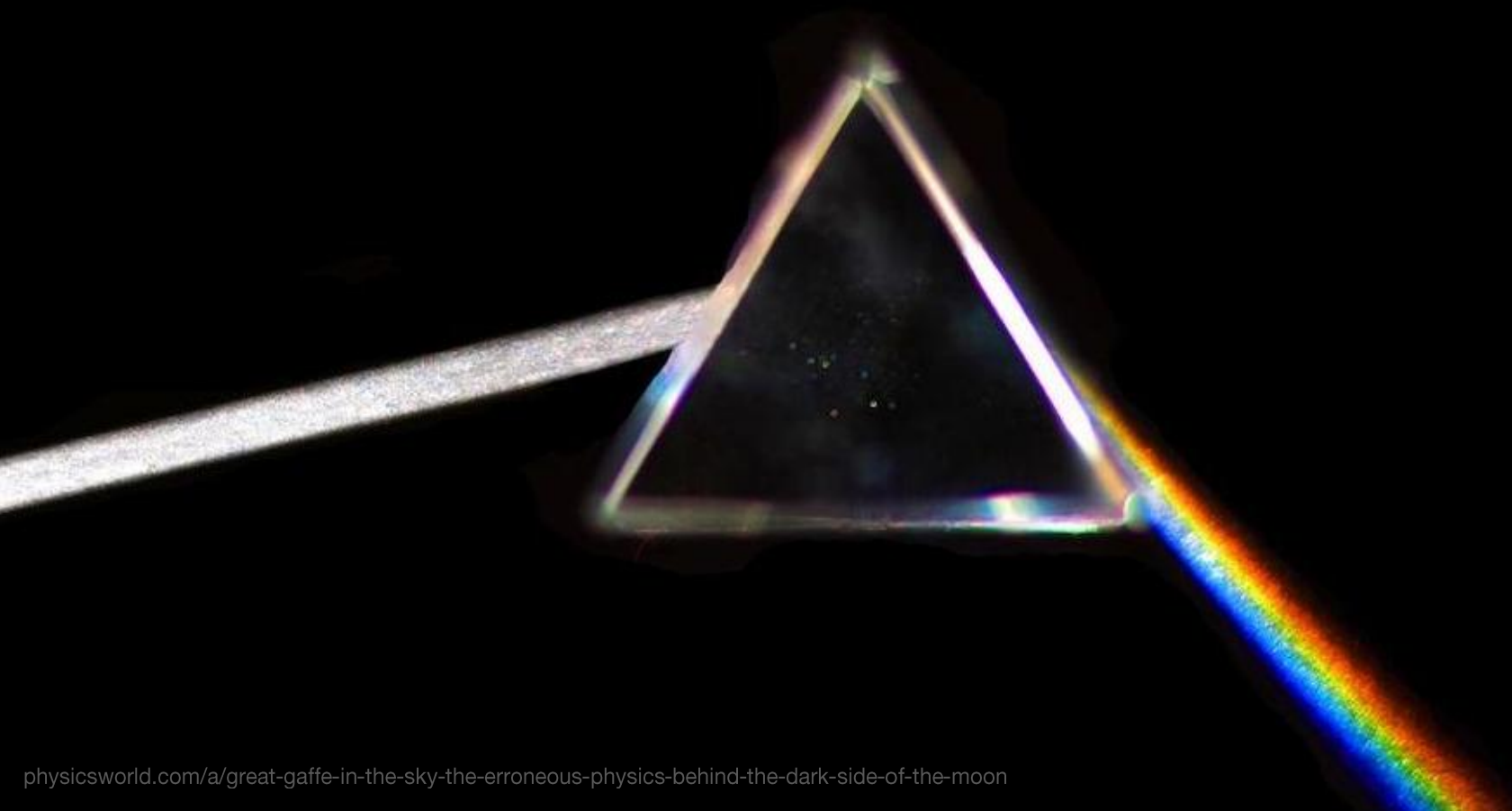
**I can't change
the laws of physics!**

Montgomery "Scotty" Scott

physics

art





physics

mathematics

FizzBuzz



Richard Dalton

@richardadalton

FizzBuzz was invented to avoid the awkwardness of realising that nobody in the room can binary search an array.

10:29 AM · Apr 24, 2015



13



15



Share this Tweet

fizzbuzz

1

2

Fizz

4

Buzz

Fizz

7

8

Fizz

Buzz

11

Fizz

13

14

FizzBuzz

numbers	fizzes	buzzes	words
---------	--------	--------	-------

1			
2			
3	Fizz		Fizz
4			
5		Buzz	Buzz
6	Fizz		Fizz
7			
8			
9	Fizz		Fizz
10		Buzz	Buzz
11			
12	Fizz		Fizz
13			
14			
15	Fizz	Buzz	FizzBuzz

numbers

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

words

Fizz

Buzz

Fizz

Fizz

Buzz

Fizz

FizzBuzz

fizzbuzz

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz

```
numbers    = map show [1..]  
fizzes     = cycle [ "", "", "Fizz"]  
buzzes     = cycle [ "", "", "", "", "Buzz"]  
words      = zipWith (++) fizzes buzzes  
fizzbuzz   = zipWith max words numbers
```



Nothing in nature
is truly infinite.

Michael Burnham



Functional programming
relies on pure functions that
avoid side effects, making
code more predictable and
easier to reason about.

ChatGPT

side effects



side effects

work

joule

kWh

DKK

work



As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.

Meir M Lehman

“Programs, Life Cycles, and Laws of Software Evolution”

work

housework



Yvonne Lam

@yvonnezlam

My theory of tech debt is that housework is the correct metaphor for the thing we call tech debt, but we can't use it because tech has been made up of people who don't do housework, or manage housework being done.

9:13 PM · Mar 29, 2021



4.9K

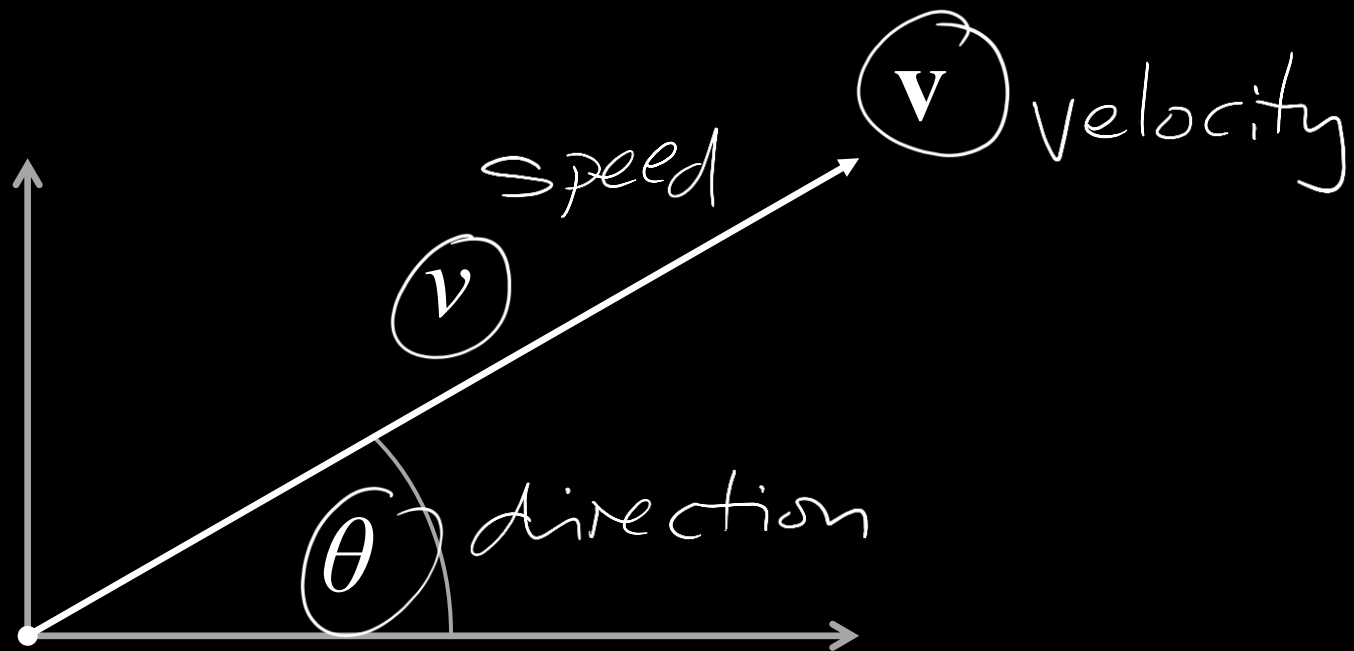


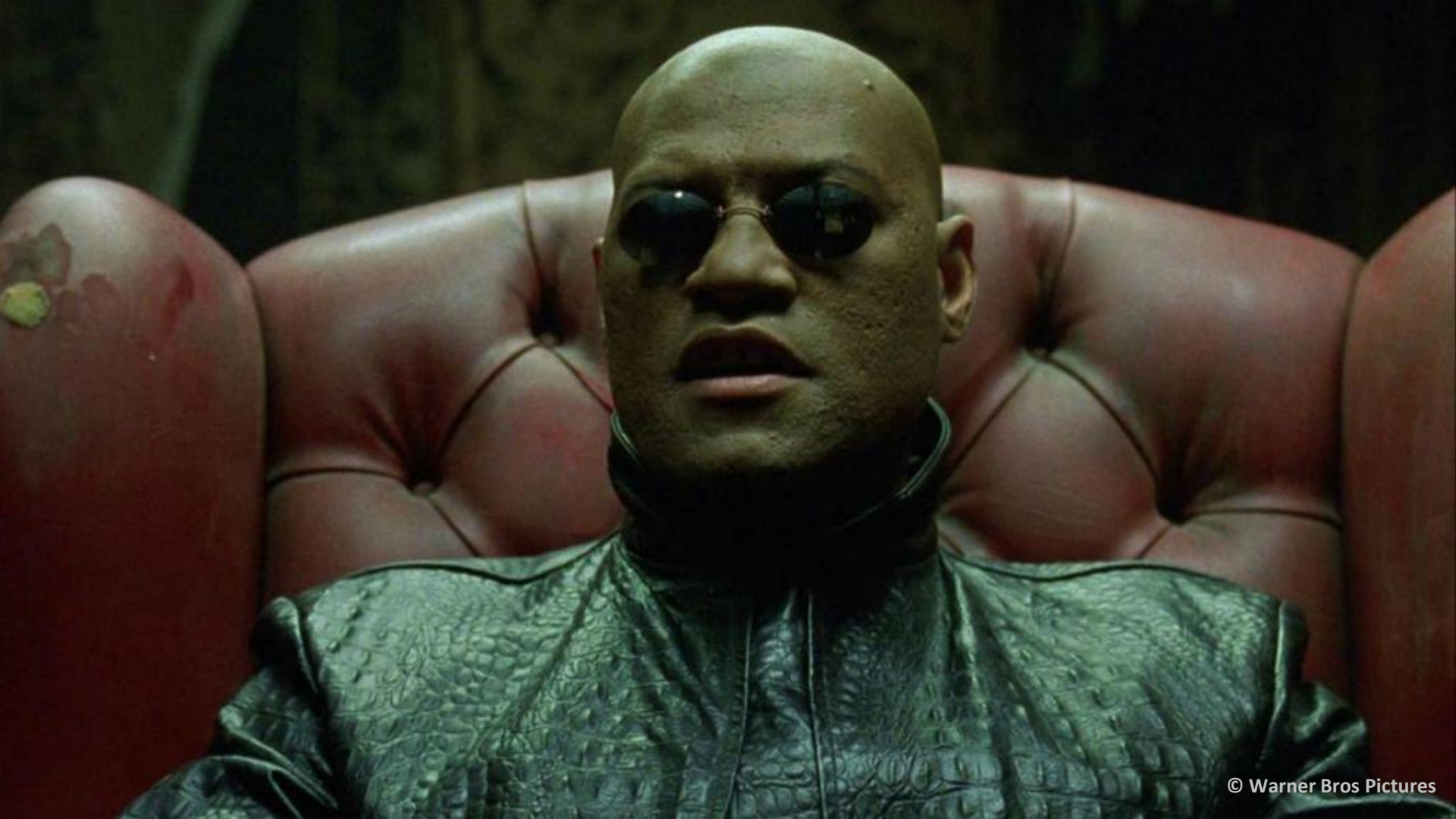
76

x.com/yvonnezlam/status/1376628481878990852

velocity

You keep using that word.
I do not think it means
what you think it means.





Stop trying
to hit me
and hit me.



Seth Rosen

@sethrosen

When naming things like tables, always future proof

If you sell wine, calling a table "wines" will be confusing when expanding to beer

"Beverages" will be confusing when you sell ice

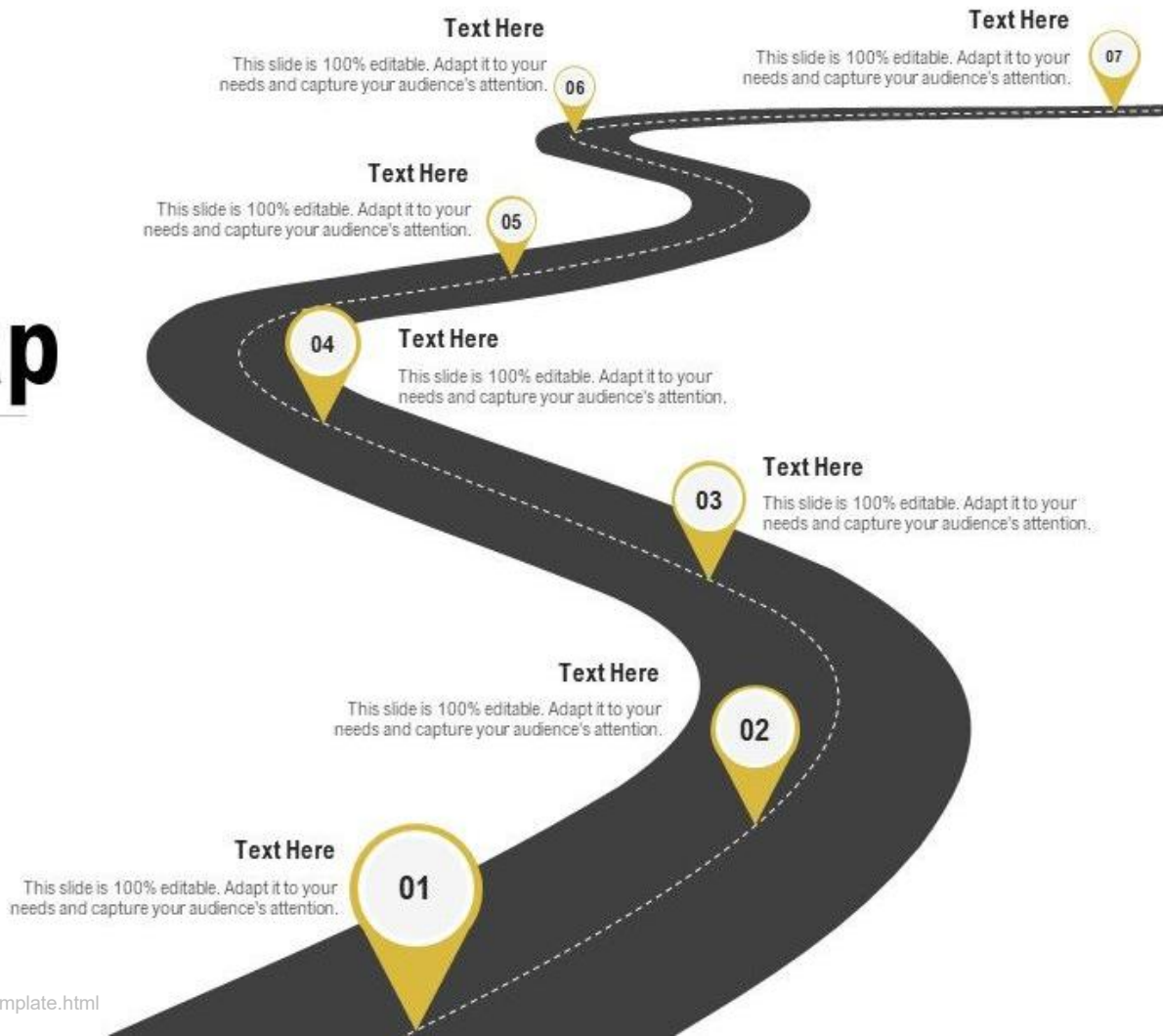
"Products" confusing when expand to services

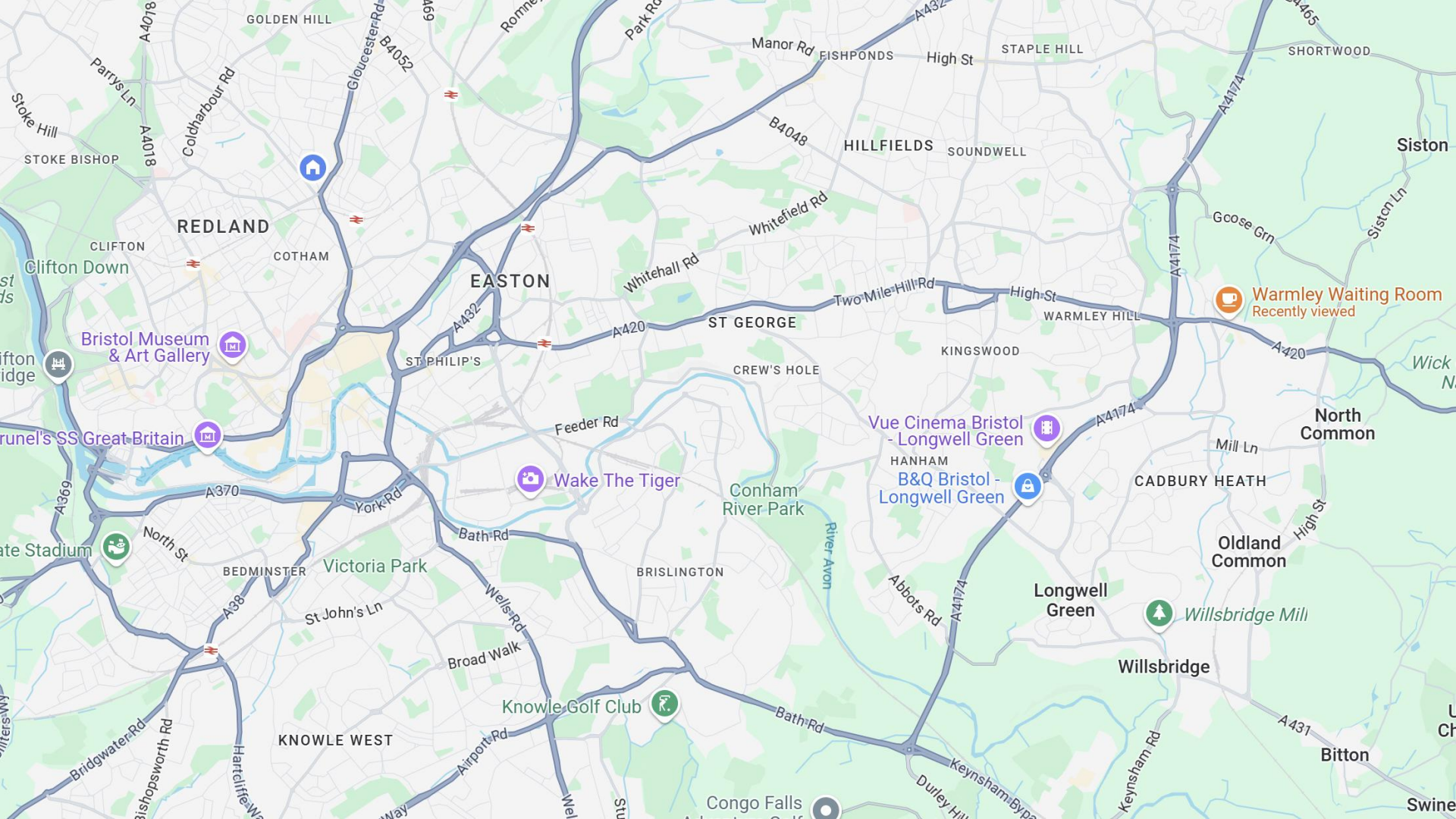
recommend tables be called "stuff" or "table1"

1:15 PM · Aug 20, 2021

x.com/sethrosen/status/1428692052968185863

Roadmap





The whole point of a
roadmap is that it shows
you a network of roads,
not just one.

Kevlin Henney

devm.io/programming/software-development-roadmap

A real roadmap is filled
with possibilities and
choices.

Kevlin Henney

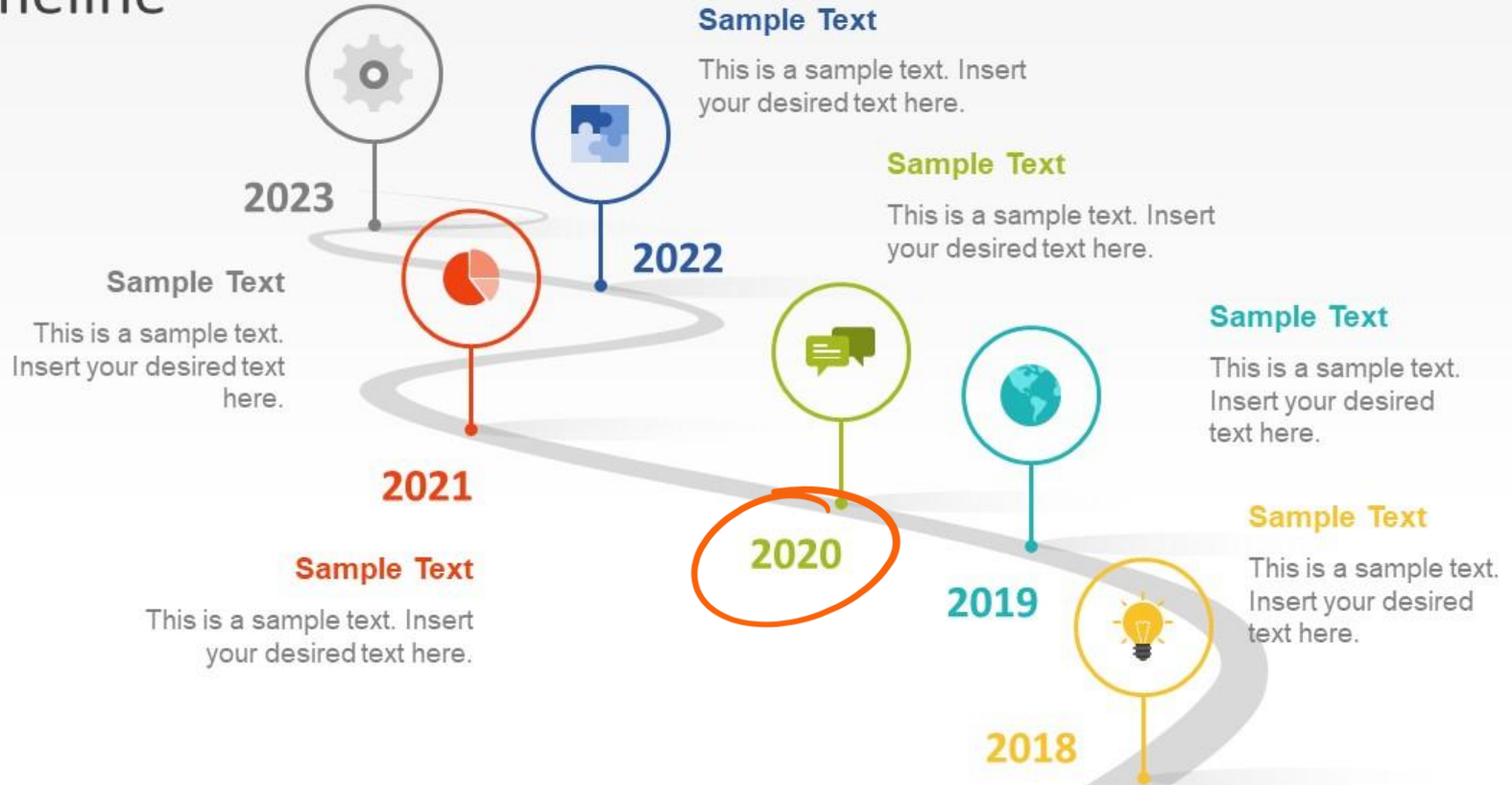
devm.io/programming/software-development-roadmap

Your decisions are not
set in stone; you can
respond to change.

Kevlin Henney

devm.io/programming/software-development-roadmap

Curved Roadmap with Poles Milestones PowerPoint Timeline





A close-up of Yoda's face and upper body. He has a wrinkled, green skin with a textured pattern. His large, expressive eyes are looking slightly to the right. He has long, thin ears and a small, pointed nose. He is wearing a brown, textured robe. His hands are clasped in front of him, showing his green fingers and claws. The background is dark and out of focus.

**Impossible to
see the future is.**

business value

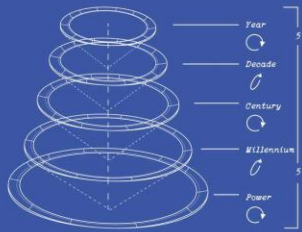
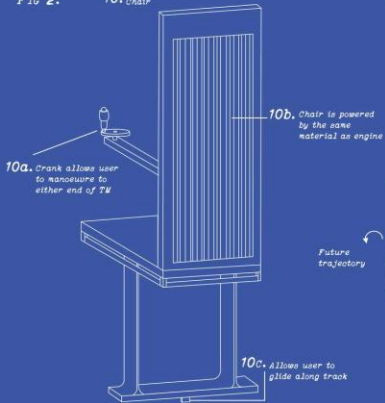
~~prioritise by
business value~~

*Designs provided by
Wells & Co.*

Signature: _____

Witnessed by E. Diaz

*Machine intended
for the travel and
exploration of time.*



These rings calculate the amount of years travelled. They generate an electric current which travels to the bulb NO. 8, revealing the exact year the user is currently in.

NOTE:
Levers must
be fully
screwed into
bolt before
any time
travel can
take place



prioritise by
business value
estimate



T
A
R
D
I
S

TOILET

T
A
R
D
I
S

Time
And
Relative
Dimension
In
Space

Time

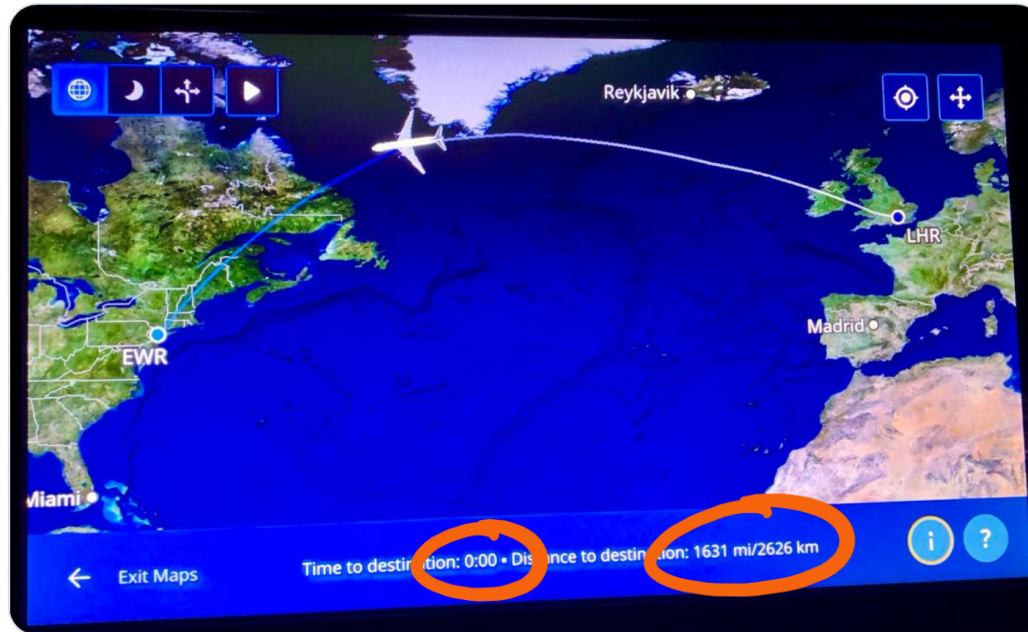
Space



Martin Fowler ✓

@martinfowler

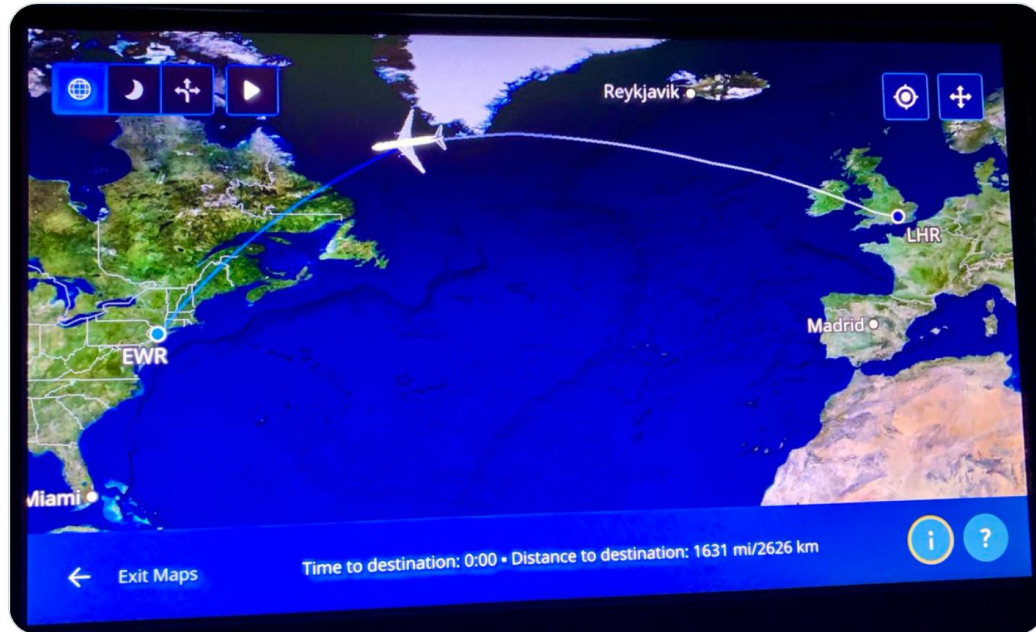
@KevlinHenney you ought to know that @united may have discovered a kink in the time-space continuum over the Atlantic. But sadly they weren't able to take advantage of it and get me home earlier.



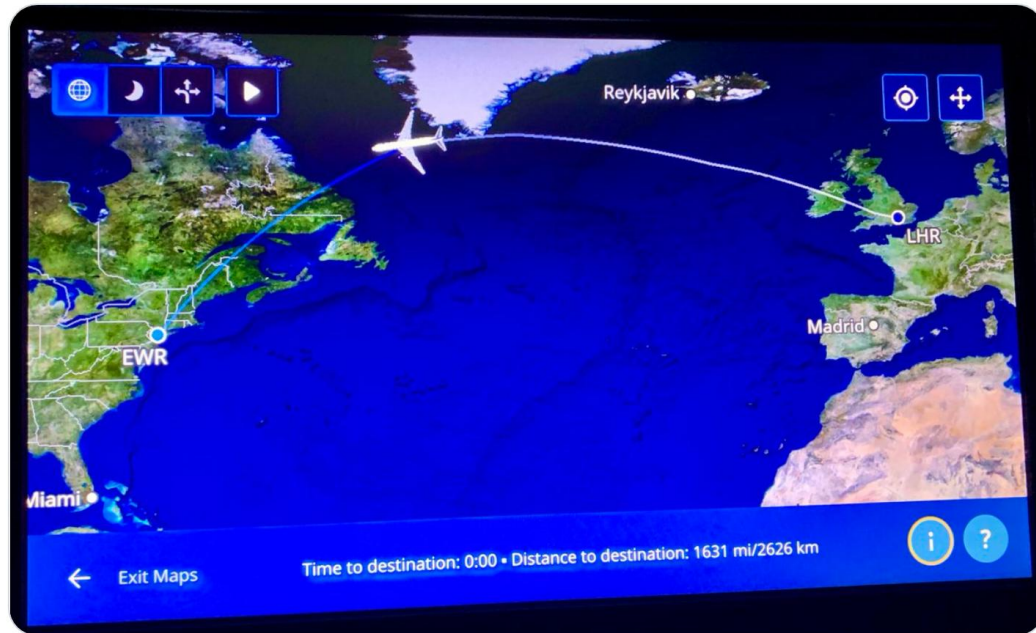
12:44 AM · Oct 7, 2022

x.com/martinfowler/status/1578154212974022657

5577 km



19 c ms



C

speed of
light

299 792 458

c

constant

celeritas

speed of
causality

Operating
Systems

R. Stockton Gaines
Editor

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

Leslie Lamport
Massachusetts Computer Associates, Inc.

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A network of interconnected computers, such as the ARPA net, is a distributed system. A single computer can also be viewed as a distributed system in which the central control unit, the memory units, and the input-output channels are separate processes. A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process.

We will concern ourselves primarily with systems of spatially separated computers. However, many of our remarks will apply more generally. In particular, a multiprocessing system on a single computer involves problems similar to those of a distributed system because of the unpredictable order in which certain events can occur.

In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation "happened before" is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

In this paper, we discuss the partial ordering defined by the "happened before" relation, and give a distributed algorithm for extending it to a consistent total ordering of all the events. This algorithm can provide a useful mechanism for implementing a distributed system. We illustrate its use with a simple method for solving synchronization problems. Unexpected, anomalous behavior can occur if the ordering obtained by this algorithm

Operating
Systems

R. Stockton Gaines
Editor

In a distributed system, it is sometimes impossible to say that one of two events occurred first.

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

Leslie Lamport

Massachusetts Computer Associates, Inc.

The relation “happened before” is therefore only a partial ordering of the events in the system.

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A network of interconnected computers, such as the ARPA net, is a distributed system. A single computer can also be viewed as a distributed system in which the central components, the memory units, and the input-output channels are separate processes. A system is distributed if the message transmission delay is not negligible compared to the time between events on a single processor.

We will concern ourselves primarily with systems of spatially separated computers. However, many of our remarks will apply more generally. In particular, a multiprocessor system or a single computer involves problems similar to those of a distributed system because of the unpredictable order in which certain events can occur.

In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation “happened before” is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

In this paper we discuss the partial ordering defined by the “happened before” relation, and give a distributed algorithm for extending it to a consistent total ordering of all the events. This algorithm can provide a useful mechanism for implementing a distributed system. We illustrate its use with a simple method for solving synchronization problems. Unexpected, anomalous behavior can occur if the ordering obtained by this algorithm

Operating
Systems

R. Stockton Gaines
Editor

We have found that problems often arise because people are not fully aware of this fact and its implications.

Leslie Lamport

Massachusetts Computer Associates, Inc.

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A network of interconnected computers, such as the ARPA net, is a distributed system. A single computer can also be viewed as a distributed system in which the central control unit, the memory unit, and the input-output channels are separate processes. A system is distributed if the message transmission delay is not negligible compared to the time between events in the process.

We will concern ourselves primarily with systems of spatially separated computers. However, many of our remarks will apply more generally. In particular, a multiprocess system on a single computer involves problems similar to those of a distributed system because of the unpredictable order in which certain events can occur.

In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation “happened before” is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

In this paper, we discuss the partial ordering defined by the “happened before” relation, and give a distributed algorithm for extending it to a consistent total ordering of all the events. This algorithm can provide a useful mechanism for implementing a distributed system. We illustrate its use with a simple method for solving synchronization problems. Unexpected, anomalous behavior can occur if the ordering obtained by this algorithm

Asynchronous



I Am Developer 

@iamdeveloper

10 Things You'll Find Shocking About Asynchronous Operations:

3.

2.

7.

4.

6.

1.

9.

10.

5.

8.

4:15 PM · Dec 12, 2016

x.com/iamdeveloper/status/808344541669498881

Operating
Systems

R. Stockton Gaines
Editor

$a \rightarrow b$ means that it is possible for event a to causally affect event b . Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport

Massachusetts Computer Associates, Inc.

This definition will appear quite natural to the reader familiar

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events.

The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

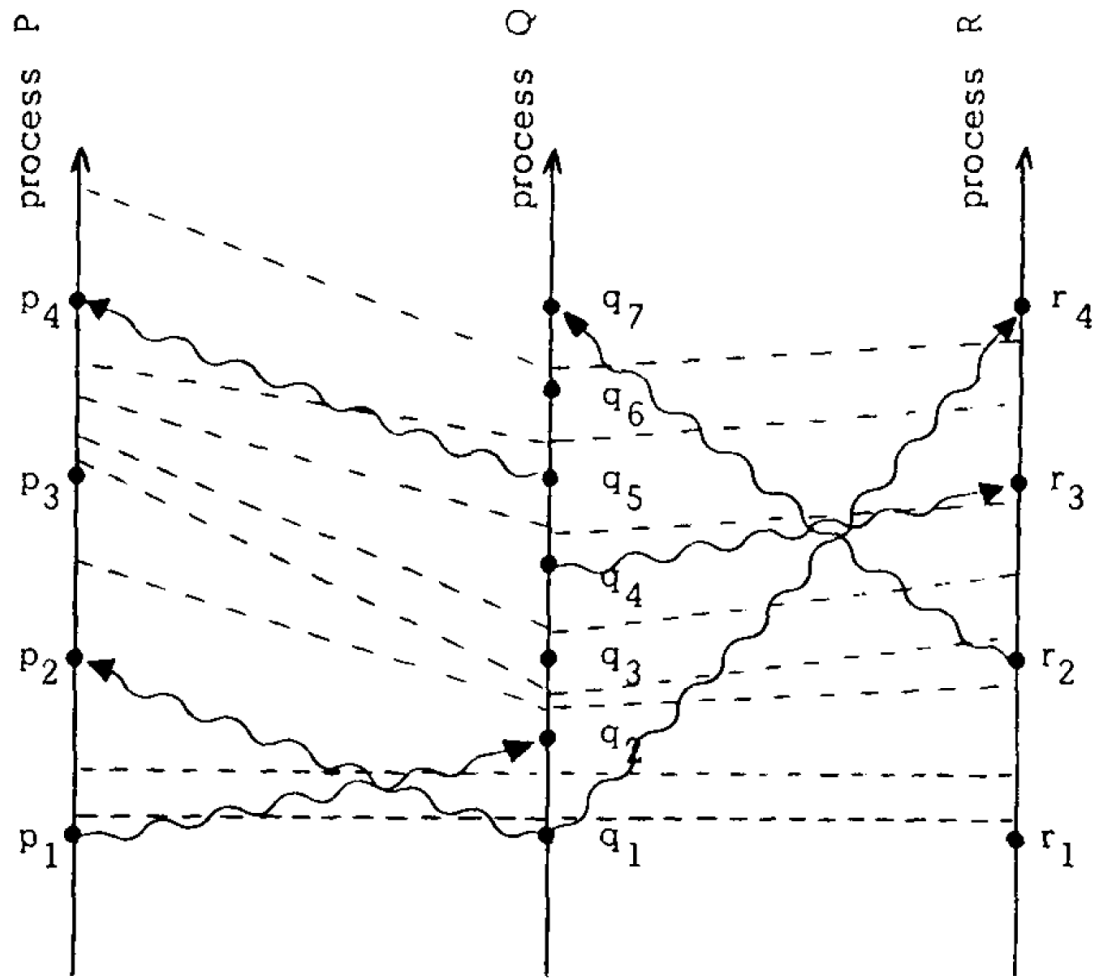
Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A network of interconnected computers, such as the ARPA net, is a distributed system. A single computer can also be viewed as a distributed system in which the central control unit, the memory units, and the input-output channels are separate processes. A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process.

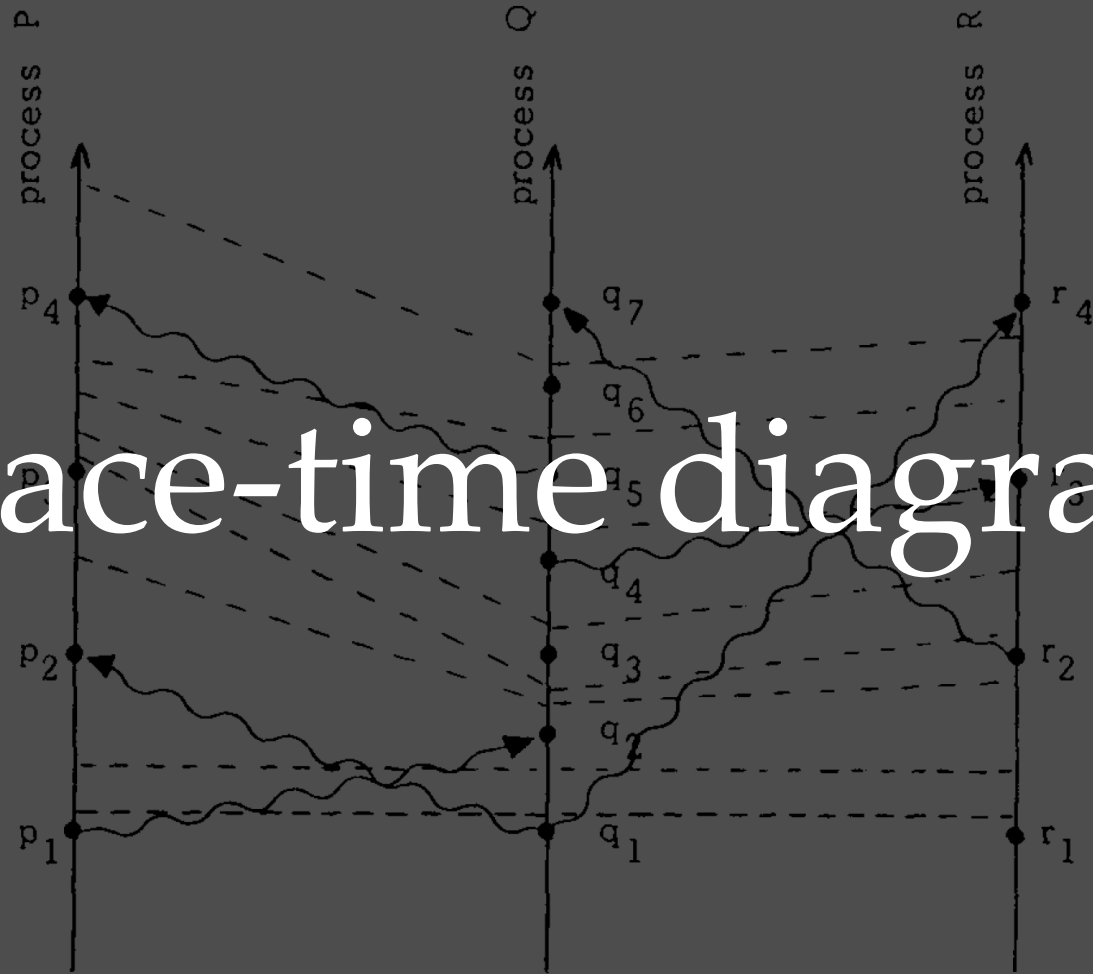
We will concern ourselves primarily with systems of spatially separated computers. However, many of our remarks will apply more generally. In particular, a multiprocessed system on a single computer involves problems similar to those of a distributed system because of the unpredictable order in which certain events can occur.

In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation "happened before" defines therefor only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

In this paper, we discuss the partial ordering defined by the "happened before" relation, and give a distributed algorithm for extending it to a consistent total ordering of all the events. This algorithm can provide a useful mechanism for implementing a distributed system. We illustrate its use with a simple method for solving synchronization problems. Unexpected, anomalous behavior can occur if the ordering obtained by this algorithm



space-time diagram



You have 0 orders

You have 3 orders

Order 1: 10 items, \$100.00

Order 2: 5 items, \$50.00

Order 3: 15 items, \$150.00



知るべき
97 Things Every Programmer Should Know

Kevlin Henney 编
李军 译 吕骏 审校
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.pri.com.cn

O'REILLY®
オライリー・ジャパン



Collective Wisdom
from the Experts

97 Things Every Programmer Should Know

O'REILLY®

Edited by Kevlin Henney

Thinking in States

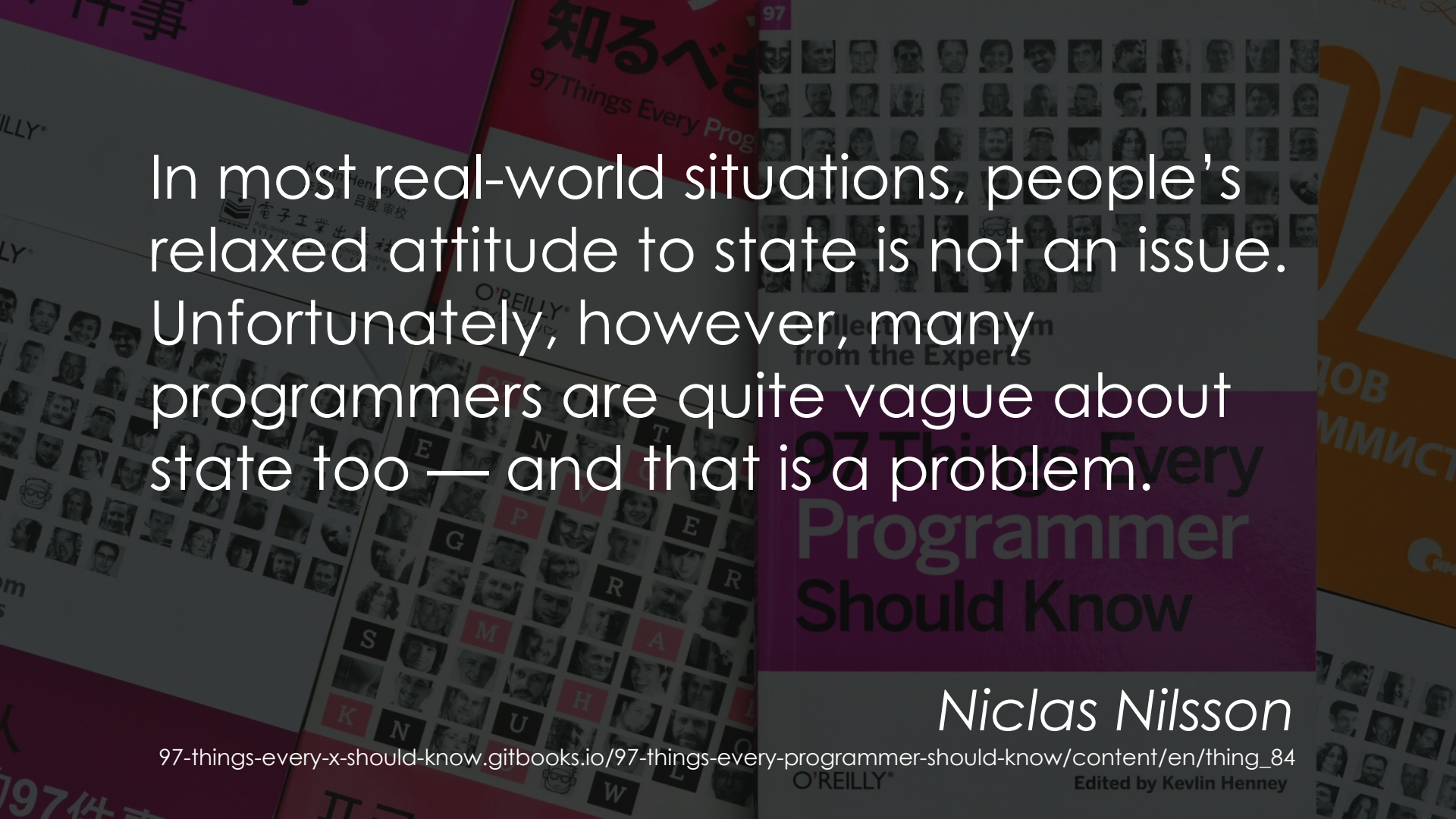
97 Things Every
Programmer
Should Know

Niclas Nilsson

97-things-every-x-should-know.gitbooks.io/97-things-every-programmer-should-know/content/en/thing_84

O'REILLY®

Edited by Kevlin Henney



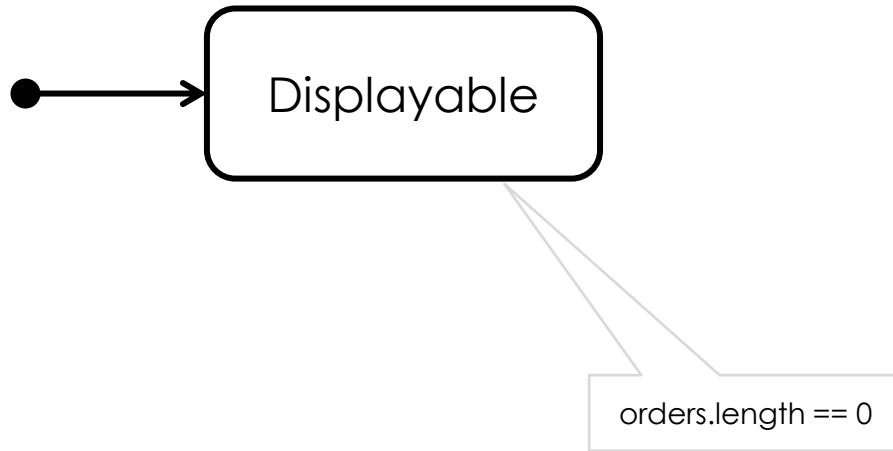
In most real-world situations, people's relaxed attitude to state is not an issue. Unfortunately, however, many programmers are quite vague about state too — and that is a problem.

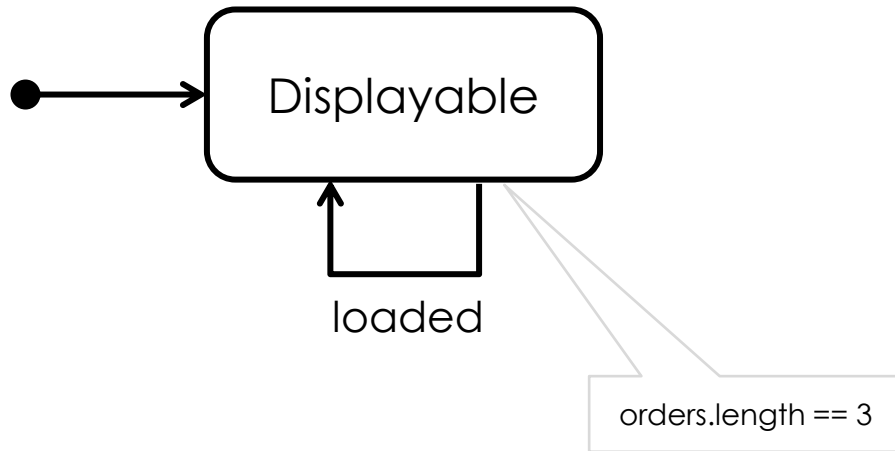
Niclas Nilsson

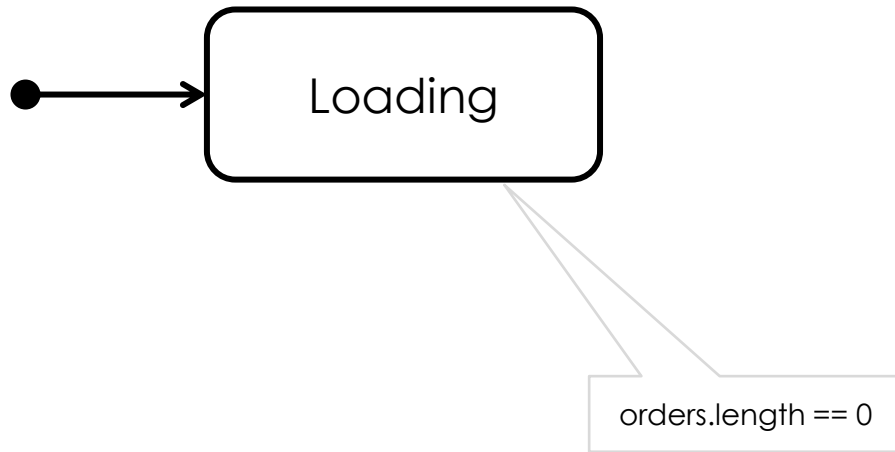
97-things-every-x-should-know.gitbooks.io/97-things-every-programmer-should-know/content/en/thing_84

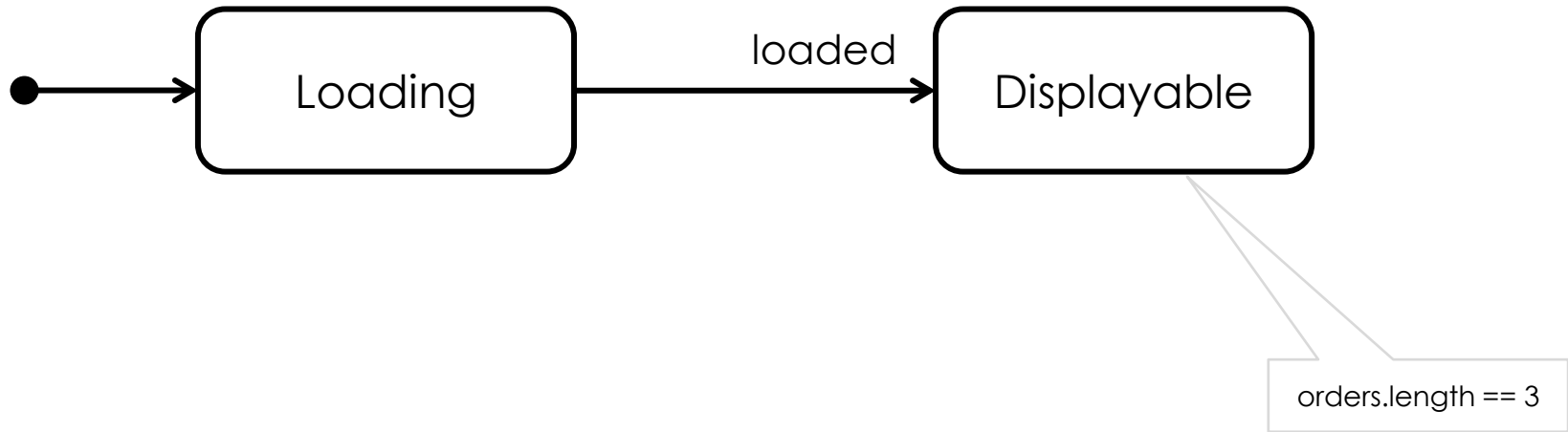
O'REILLY®

Edited by Kevlin Henney









Please wait...

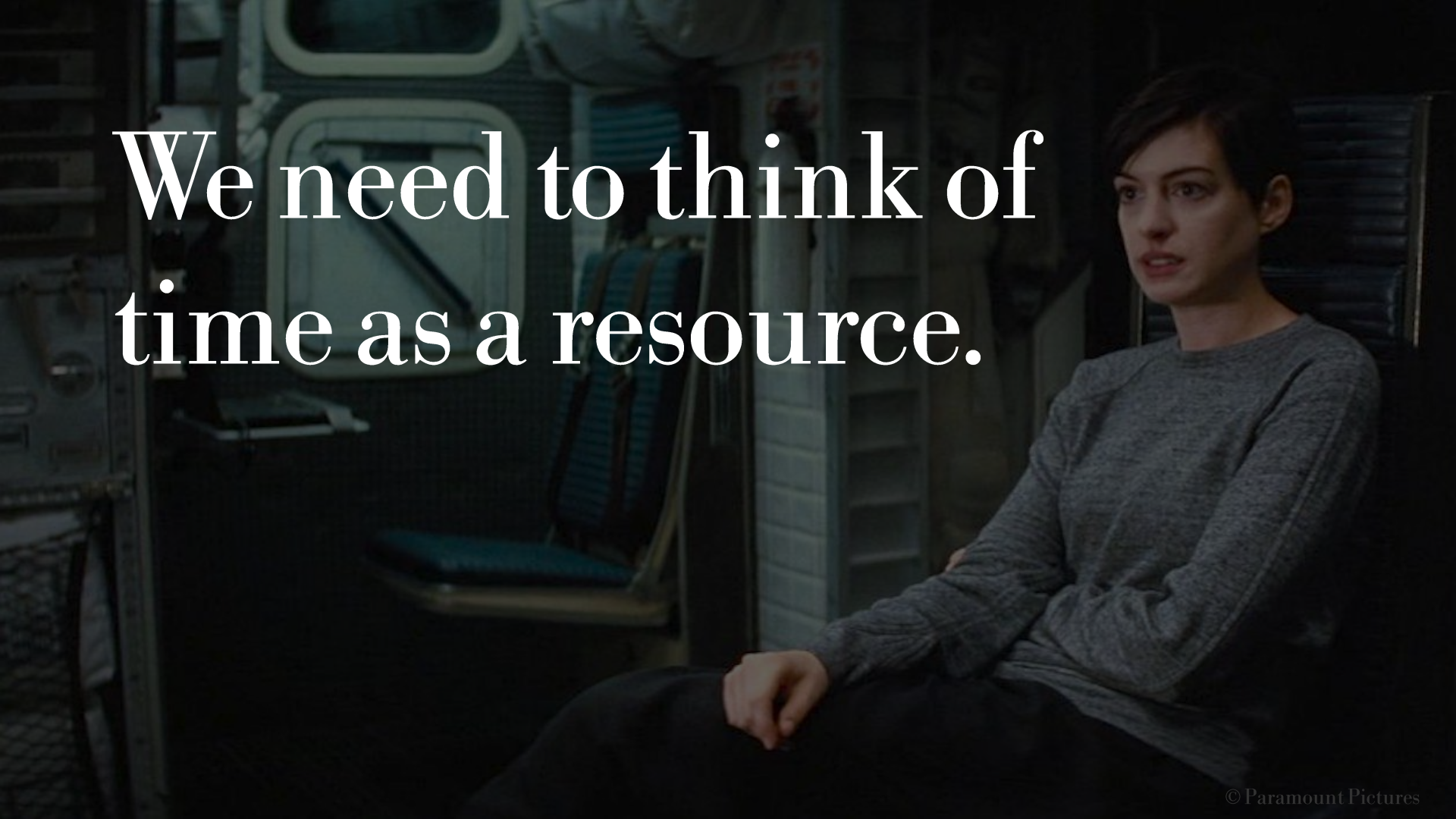
You have 3 orders

Order 1: 10 items, \$100.00

Order 2: 5 items, \$50.00

Order 3: 15 items, \$150.00



A still from the movie Interstellar showing Anne Hathaway as Amelia Brand sitting in a spacecraft. She is wearing a grey sweater and looking off to the side with a serious expression. The background shows the interior of the ship with various panels and equipment.

We need to think of
time as a resource.

The warp drive

The warp drive: hyper-fast travel within general relativity

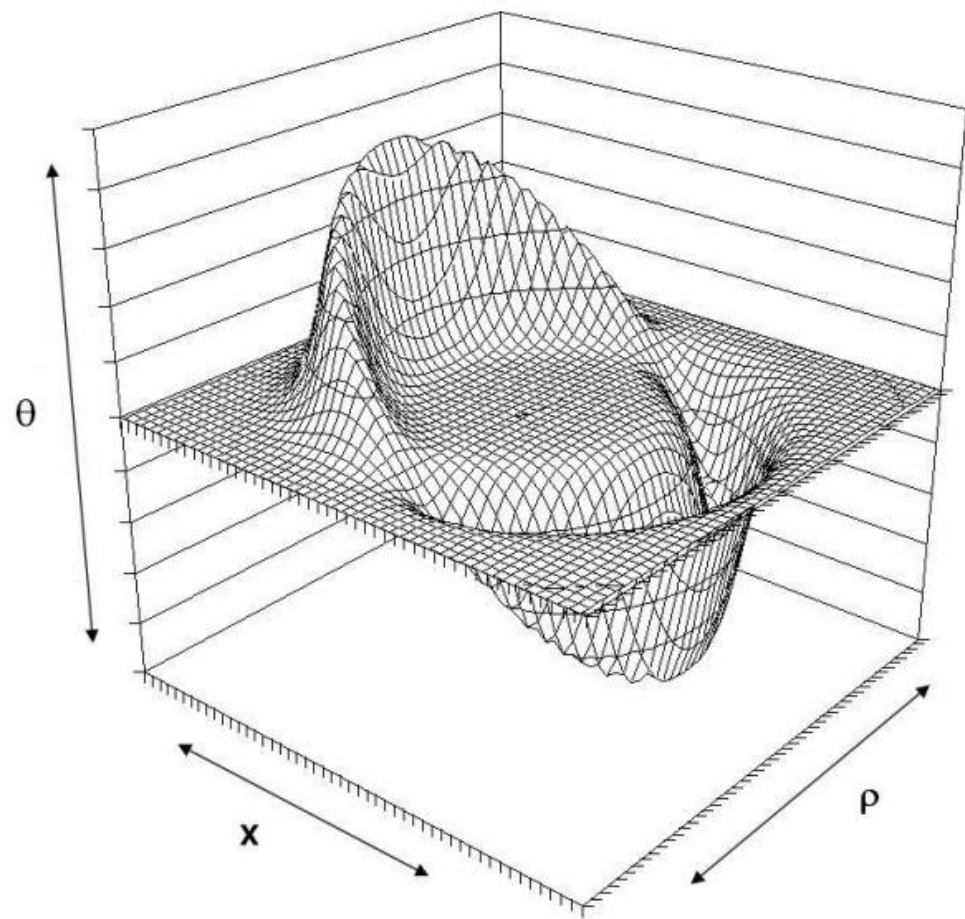
The warp drive: hyper-fast travel within general relativity

Miguel Alcubierre

Department of Physics and Astronomy, University of Wales,
College of Cardiff, PO Box 913, Cardiff CF1 3YB, UK

Received 19 January 1994, in final form 24 February 1994

Abstract. It is shown how, within the framework of general relativity and without the introduction of wormholes, it is possible to modify a spacetime in a way that allows a spaceship to travel with an arbitrarily large speed. By a purely local expansion of spacetime behind the spaceship and an opposite contraction in front of it, motion faster than the speed of light as seen by observers outside the disturbed region is possible. The resulting distortion is reminiscent of the 'warp drive' of science fiction. However, just as happens with wormholes, exotic matter will be needed in order to generate a distortion of spacetime like the one discussed here.







WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A Pattern Language for
Distributed Computing



Volume 4

Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

A distributed system is a computing system in which a number of components cooperate by communicating over a network.



WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

**PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE**

**A Practical Language for
Distributed Computing**

Volume 4

Frank Buschmann

Kevin Henney

Stefan Schmidt

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages.

Operating Systems
Leslie Lamport
Massachusetts Computer Associates, Inc.

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

Leslie Lamport
Massachusetts Computer Associates, Inc.

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A network of interconnected computers, such as the ARPANET, is a distributed system. A single computer can also be viewed as a distributed system in which the central control unit, the memory units, and the input-output channels are separate processes. A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process.

We will concern ourselves primarily with systems of spatially separated computers. However, many of our remarks will apply more generally. In particular, a multiprocess system on a single computer involves problems similar to those of a distributed system because of the unpredictable order in which certain events can occur.

In a distributed system it is sometimes impossible to say that one of two events occurred first. The relation "happened before" is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

In this paper, we discuss the partial ordering defined by the "happened before" relation, and give a distributed algorithm for extending it to a consistent total ordering of all the events. This algorithm can provide a useful mechanism for implementing a distributed system. We illustrate its use with a simple method for solving synchronization problems. Unexpected, anomalous behavior can occur if the ordering obtained by this algorithm

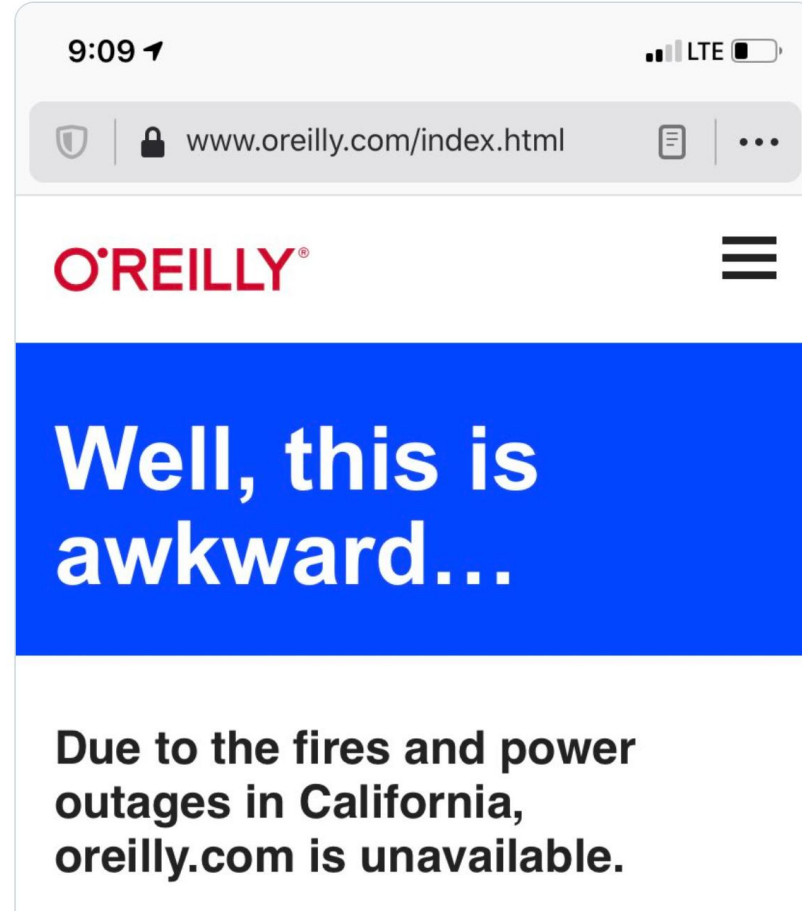
A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

Leslie Lamport



Charlie Morris
@cdmo

Fire in California, can't read your ebook in Pennsylvania



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

Seth Gilbert*

Nancy Lynch*

Abstract

When designing distributed web services, there are three properties that are commonly desired: consistency, availability, and partition tolerance. It is impossible to achieve all three. In this note, we prove this conjecture in the asynchronous network model, and then discuss solutions to this dilemma in the partially synchronous model.

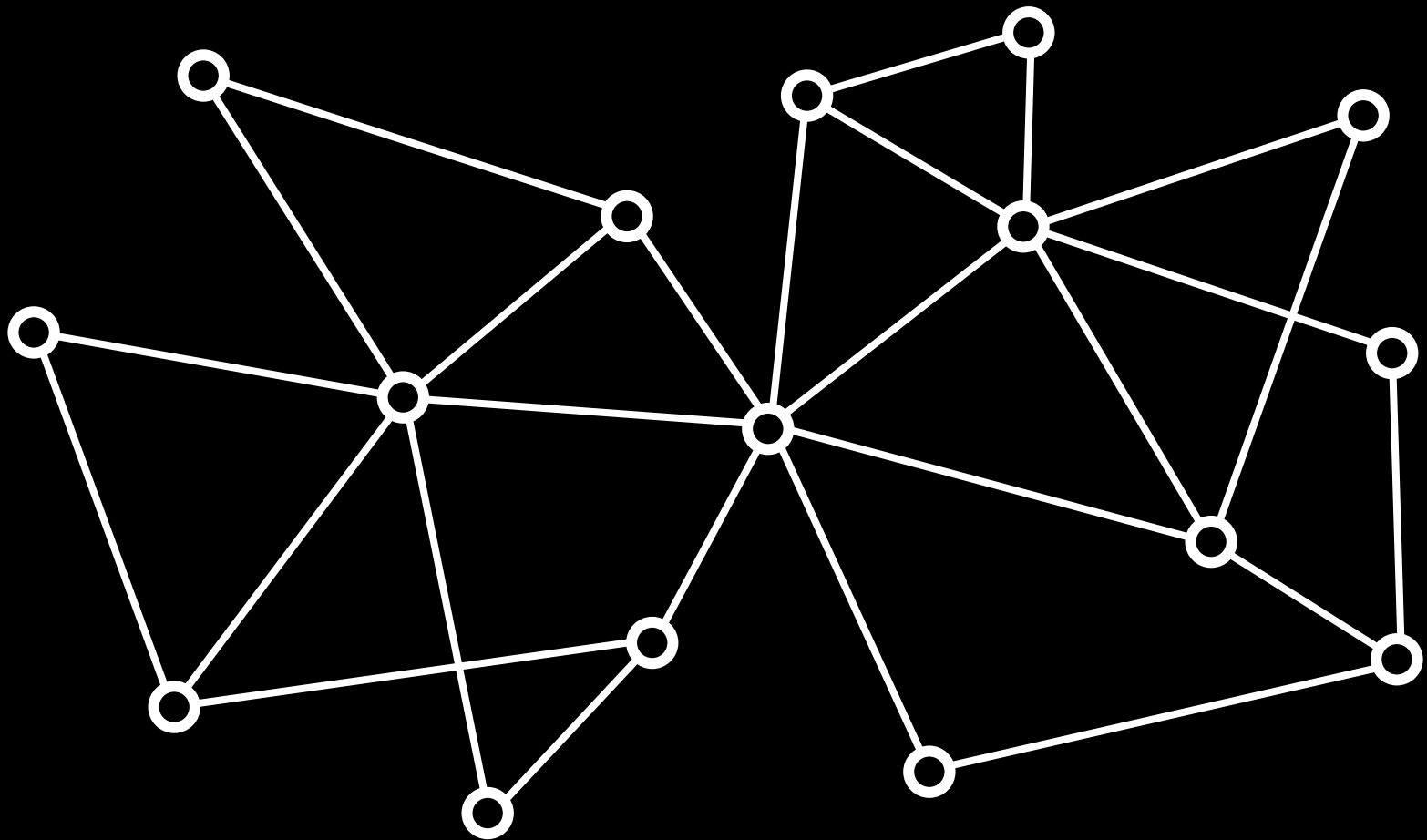
Brewer's Conjecture and the Feasibility of
Consistent, Available, Partition-Tolerant Web

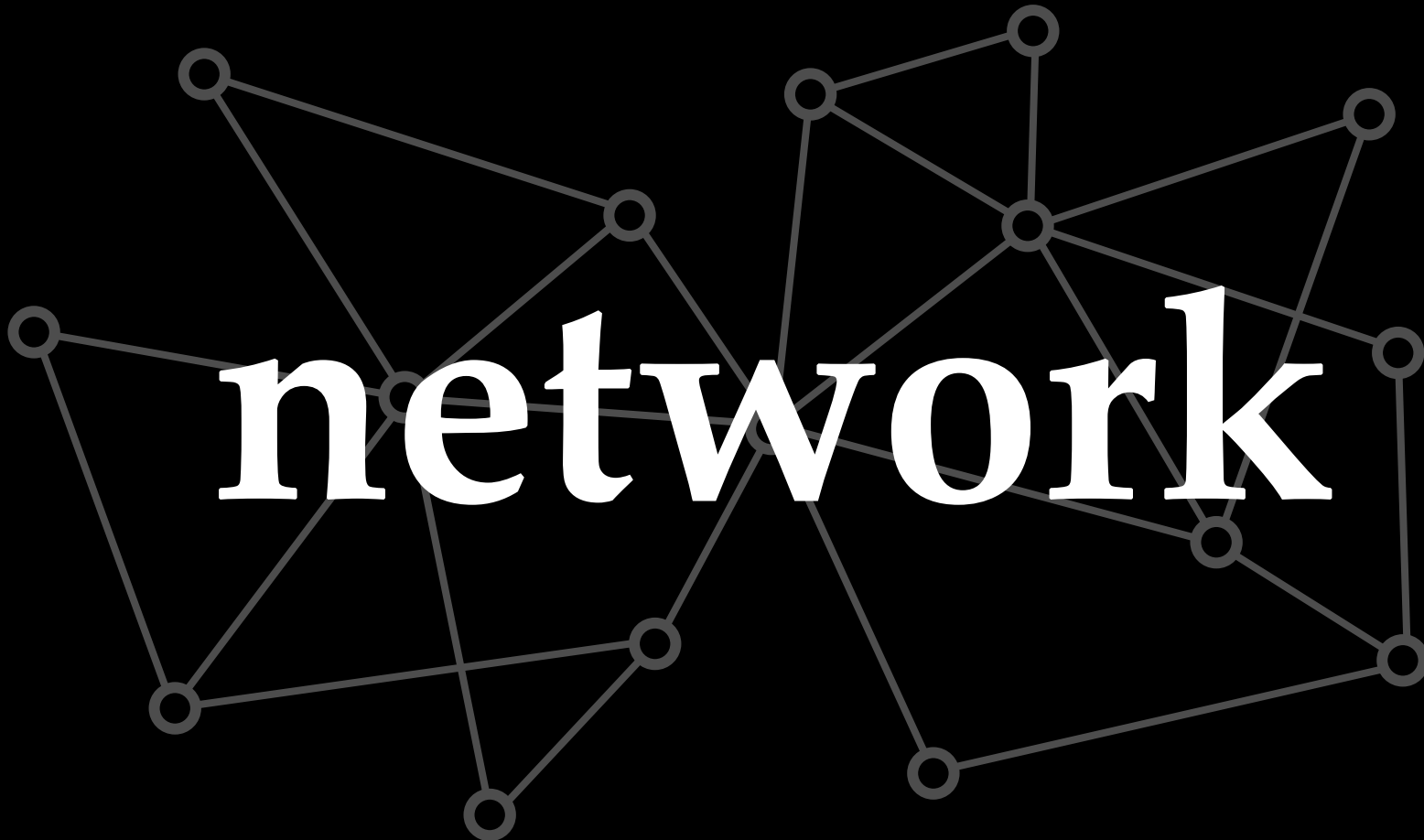
Services
When designing distributed web
services, there are three properties that
are commonly desired: consistency,
availability, and partition tolerance.
Abstract

It is impossible to achieve all three.

When designing distributed web services, there are three
properties that are commonly desired: consistency, avail-
ability, and partition tolerance. It is impossible to achieve
all three. In this note, we prove this conjecture in the asyn-
chronous network model, and then discuss solutions to this
dilemma in the partially synchronous model.

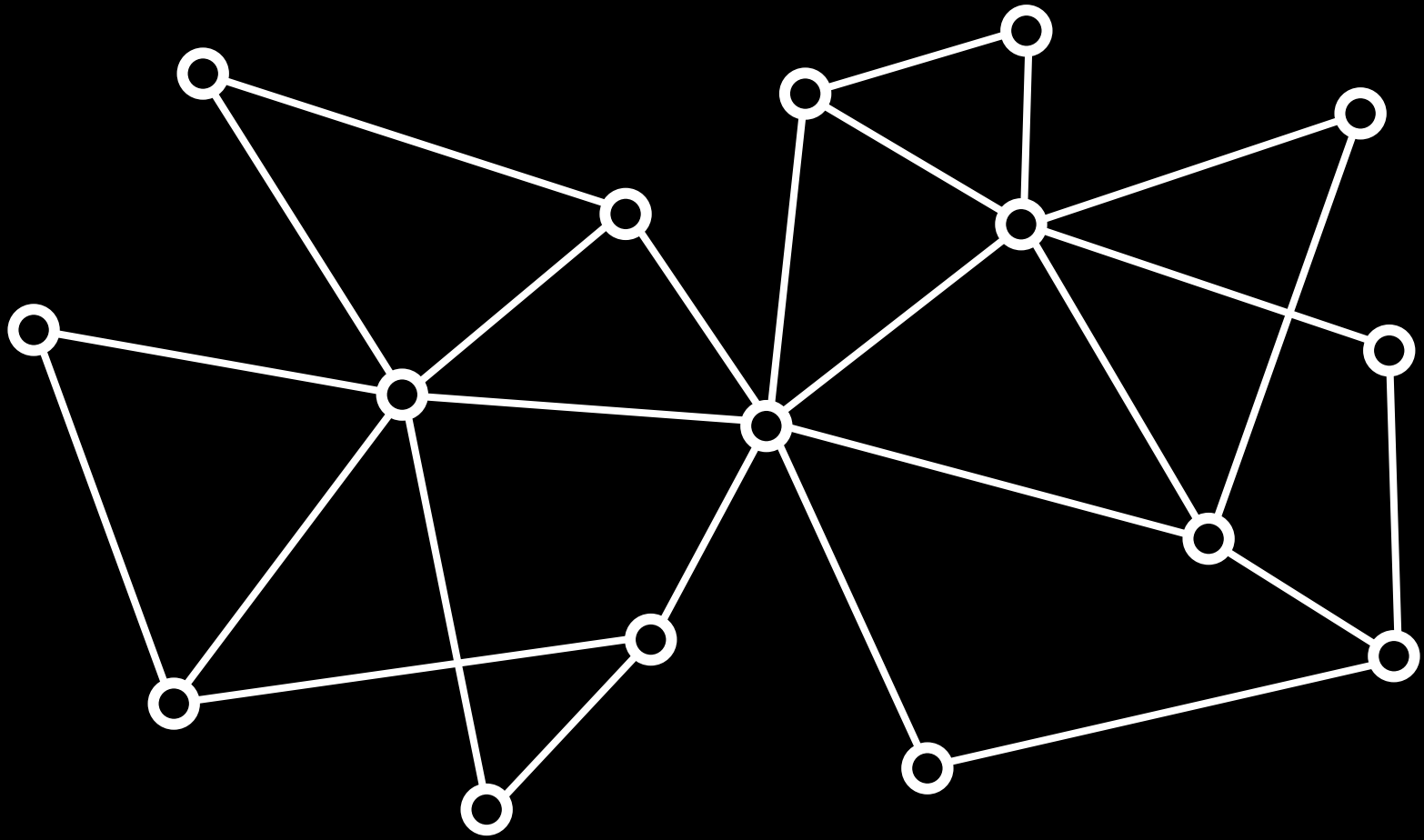
partition tolerance

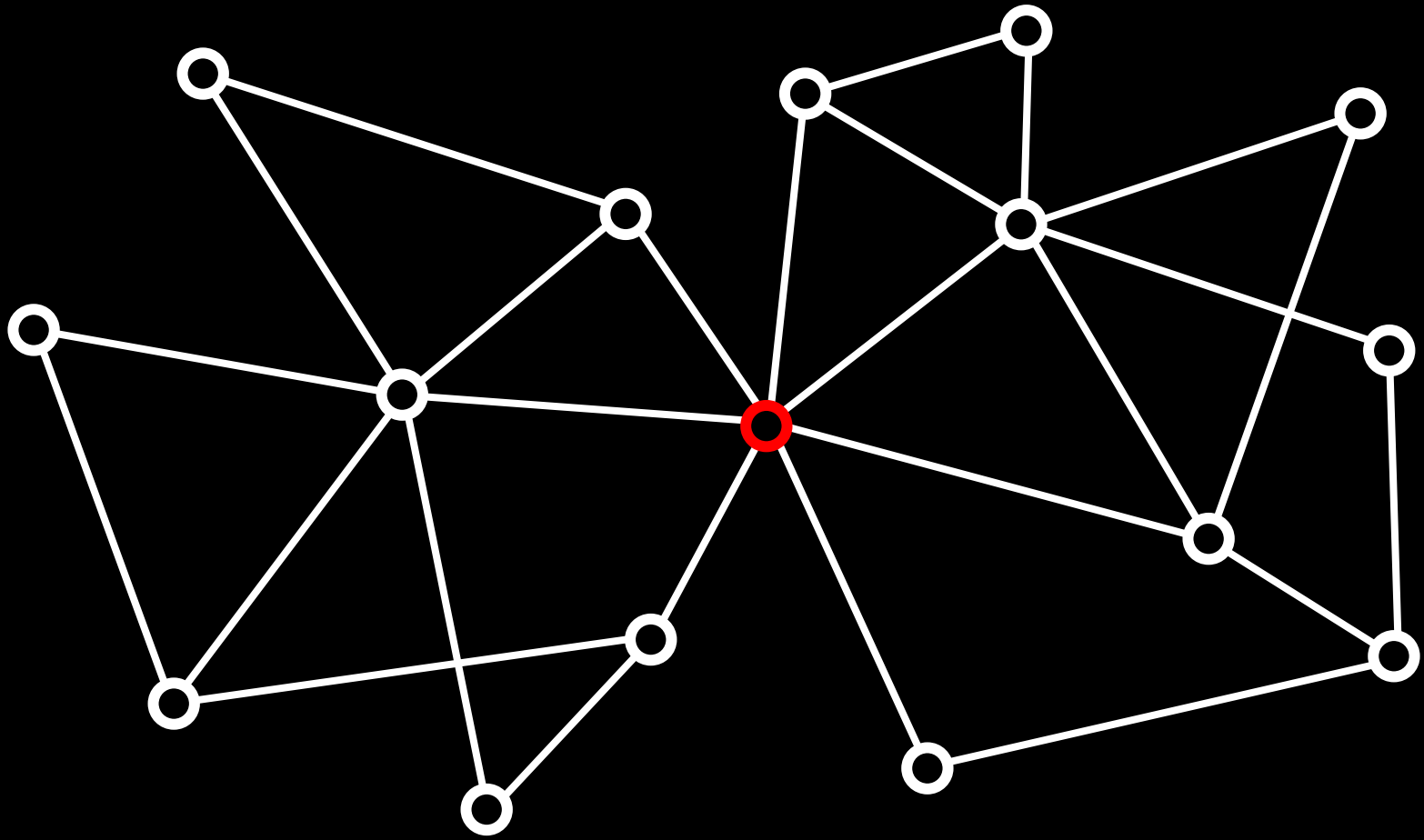


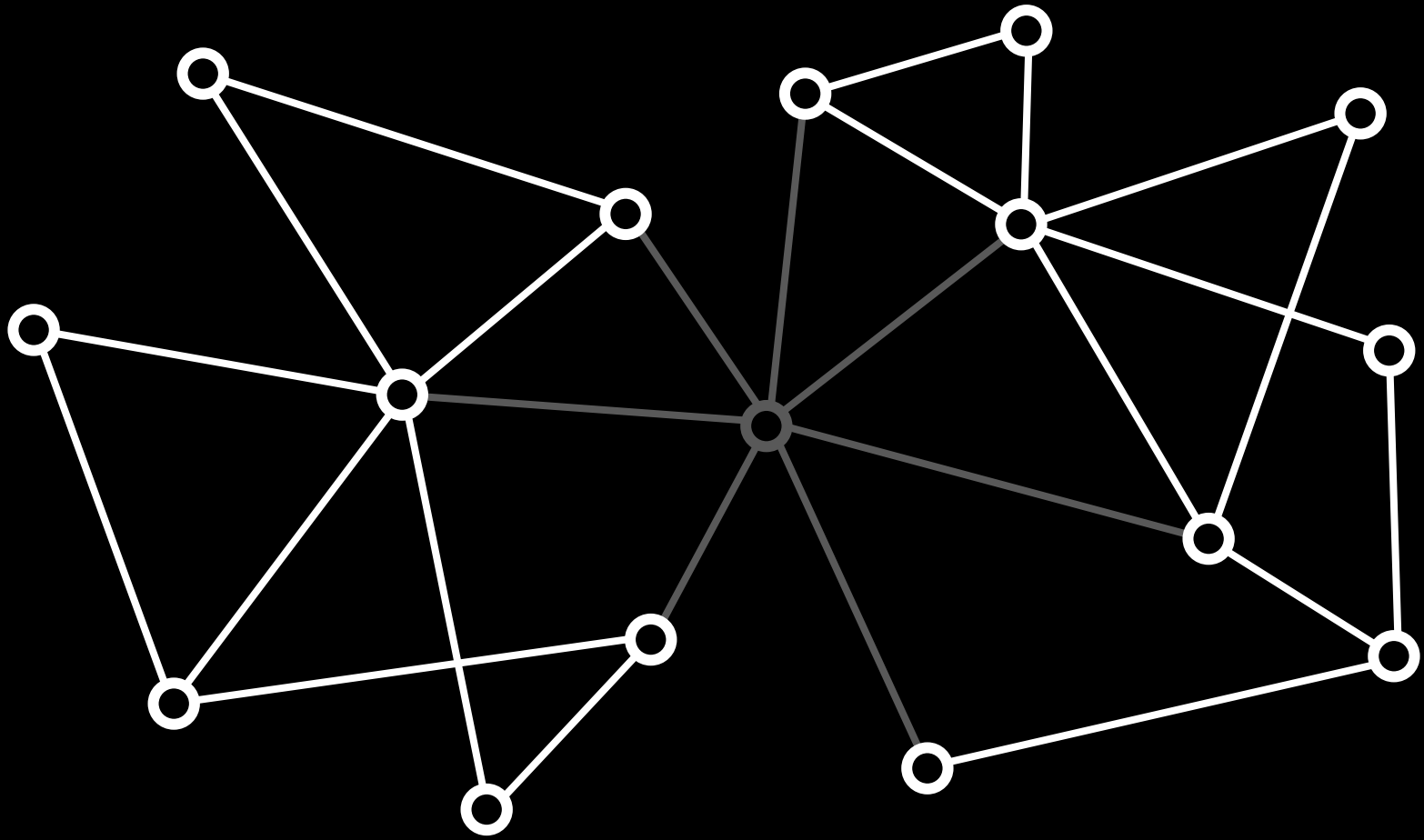


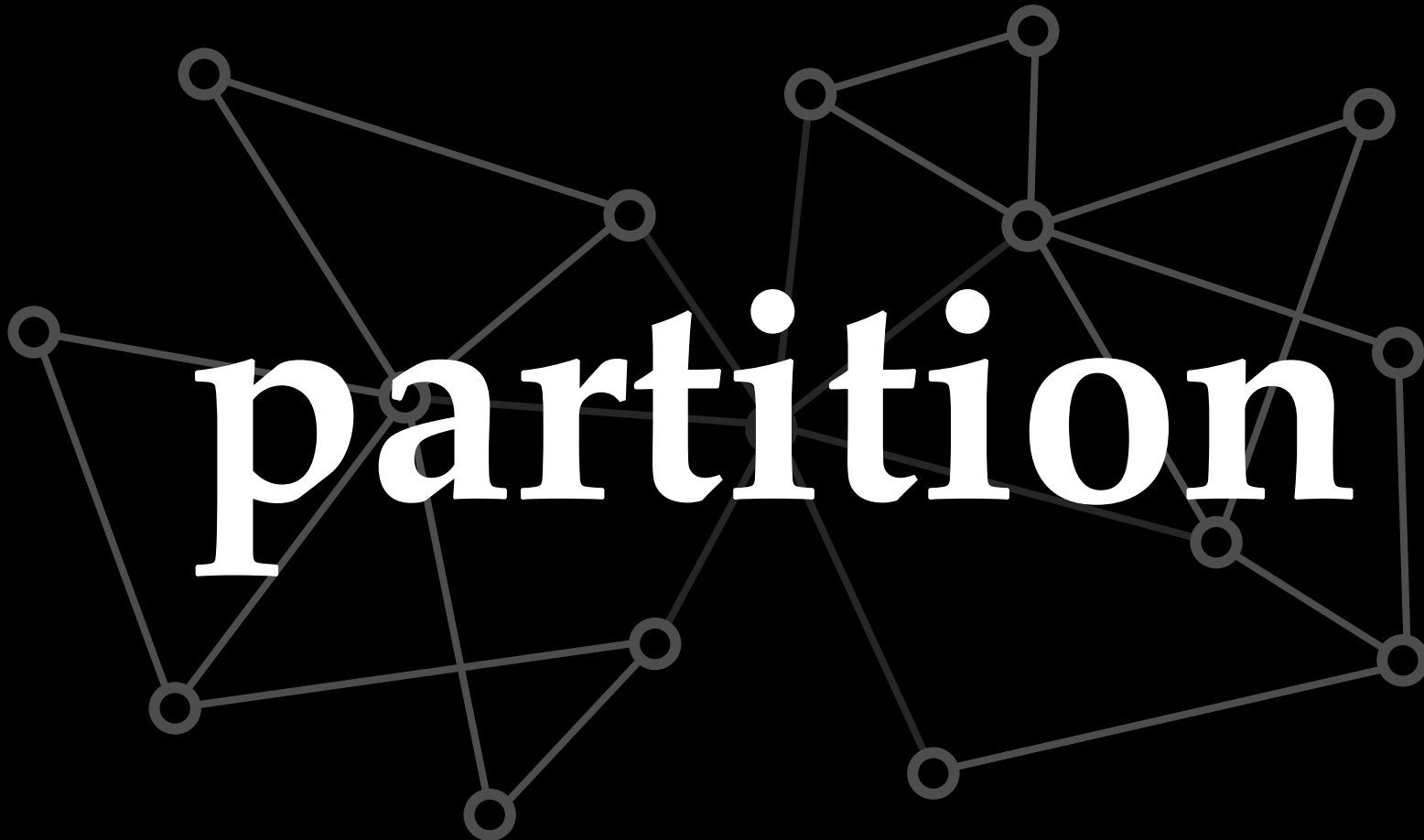
A network diagram consisting of 15 nodes (small circles) and 25 edges (lines connecting the nodes). The nodes are arranged in a complex, interconnected pattern. The word "network" is written in a large, white, serif font across the center of the diagram, partially obscuring some of the edges and nodes.

network

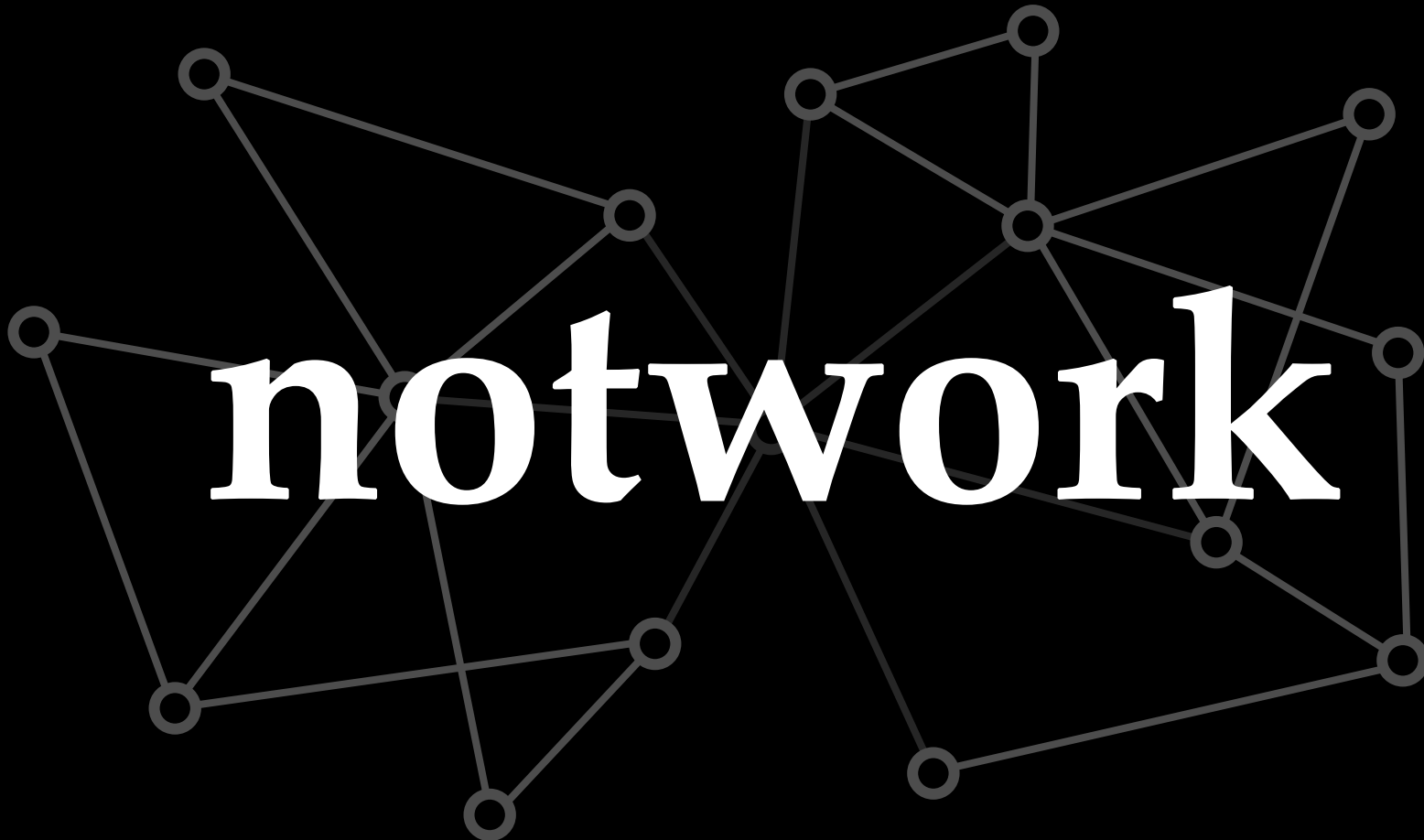




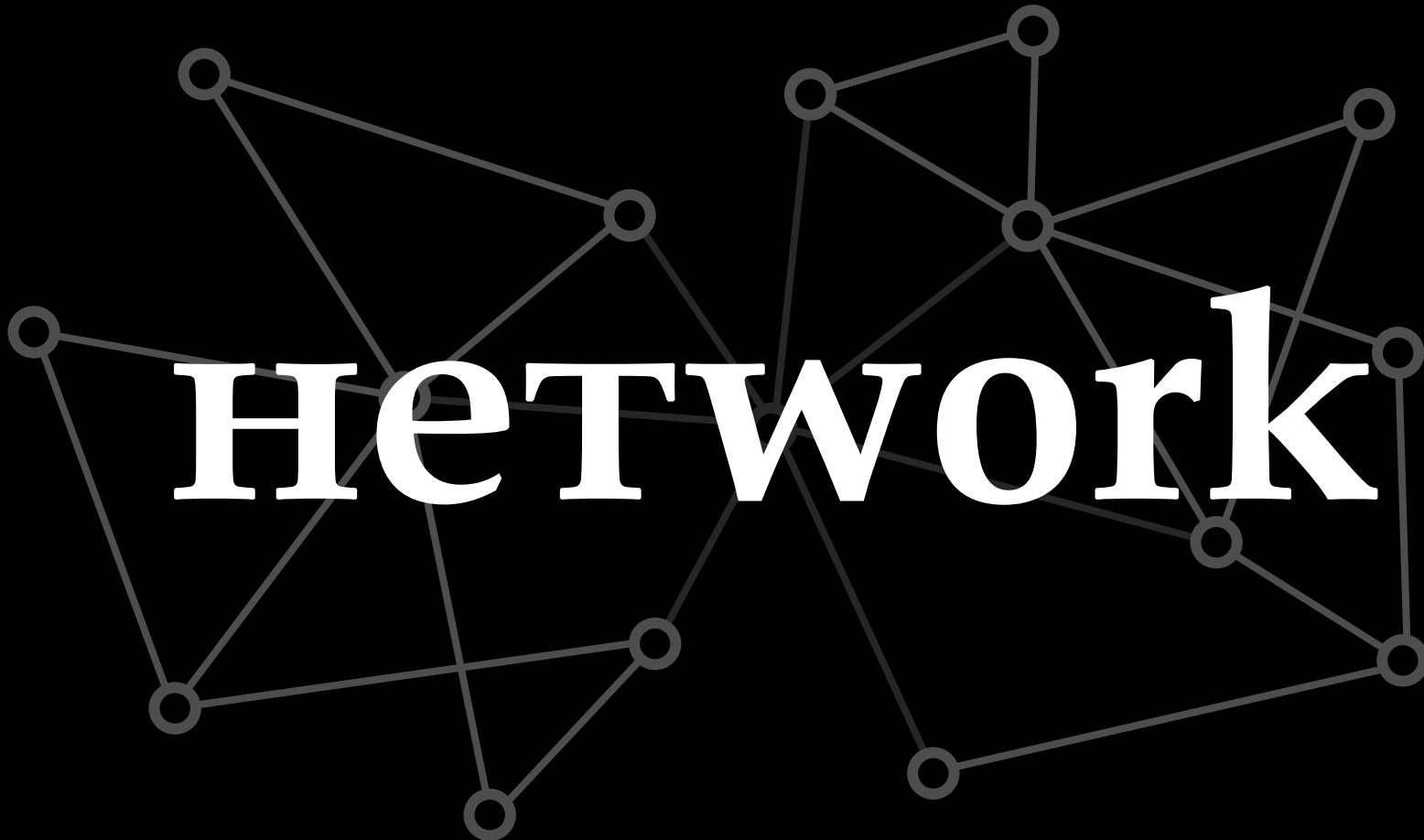




partition



network



Network

Consistency
Availability
Partition tolerance

Consistency

Availability

Partition tolerance

Consistency

Availability

Partition tolerance

Consistency

Availability

Partition tolerance

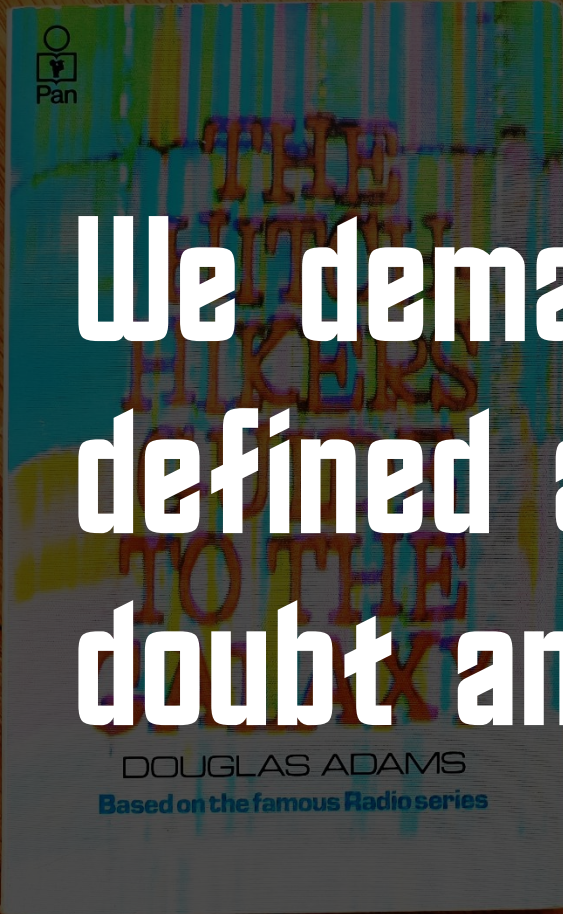
$$\Delta x \Delta p \geq \frac{\hbar}{2}$$



THE HITCH- HIKERS GUIDE TO THE GALAXY

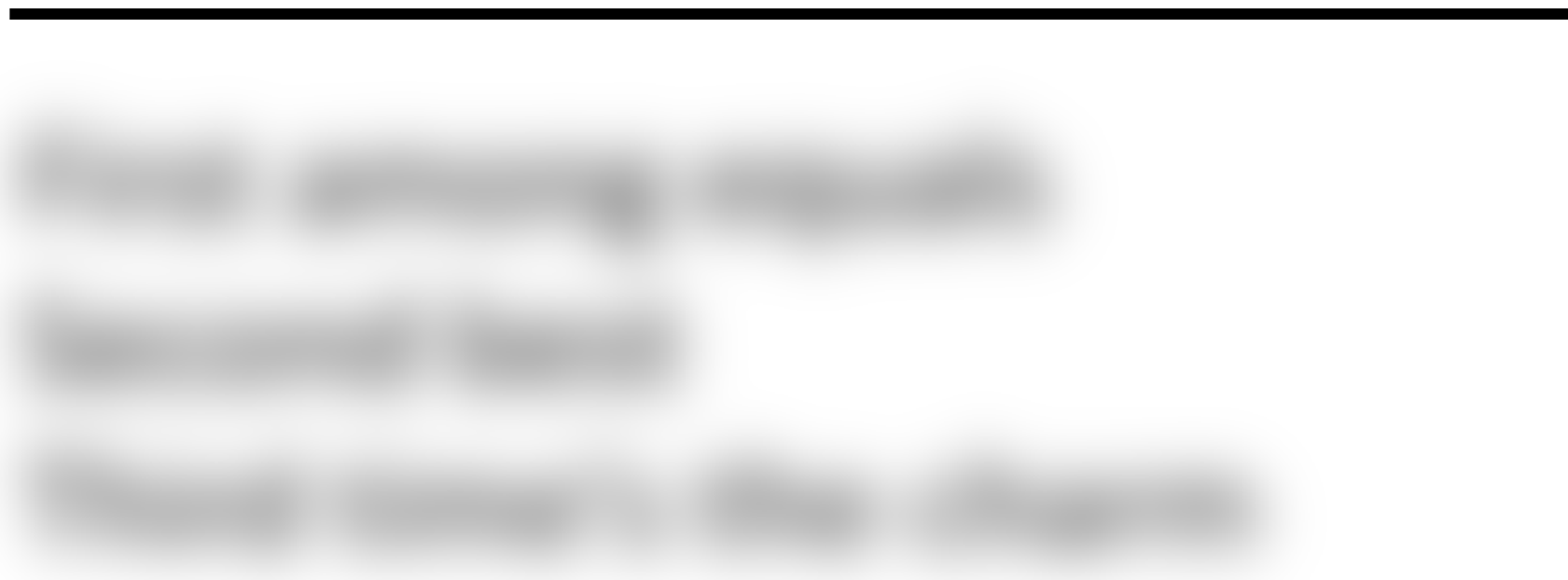
DOUGLAS ADAMS

Based on the famous Radio series

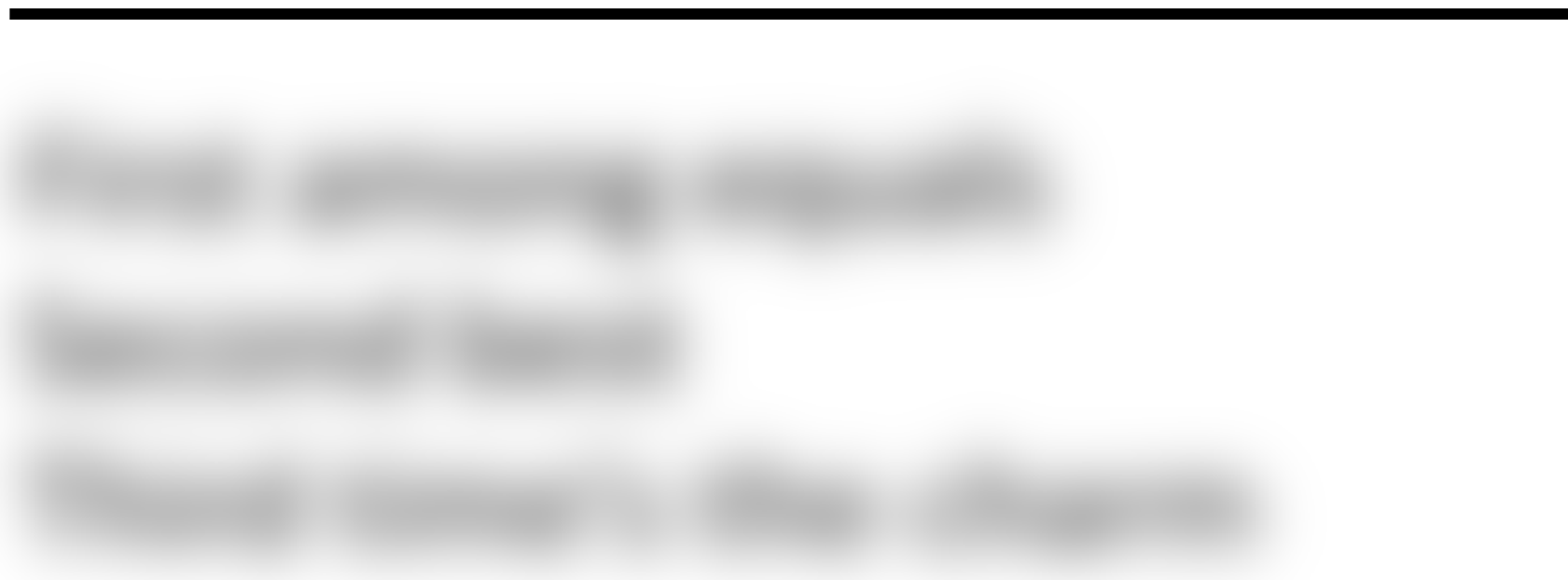


**We demand rigidly
defined areas of
doubt and uncertainty!**

You have 2 orders



You have 3 orders



It is a feature of a distributed system that it may not be in a consistent state, but it is a bug for a client to contradict itself.

x.com/KevlinHenney/status/1351956942877552646

**Reality cannot be ignored
except at a price.**

Aldous Huxley