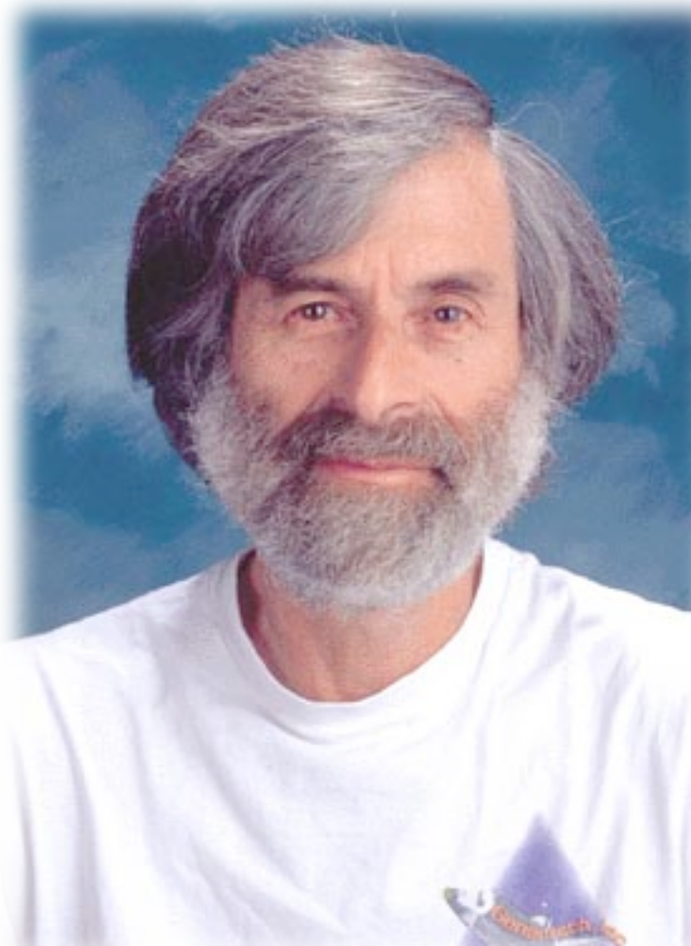
Distributed Systems

Chapter 1: **Introduction**

Prof. Dr. Markus Esch
htw saar

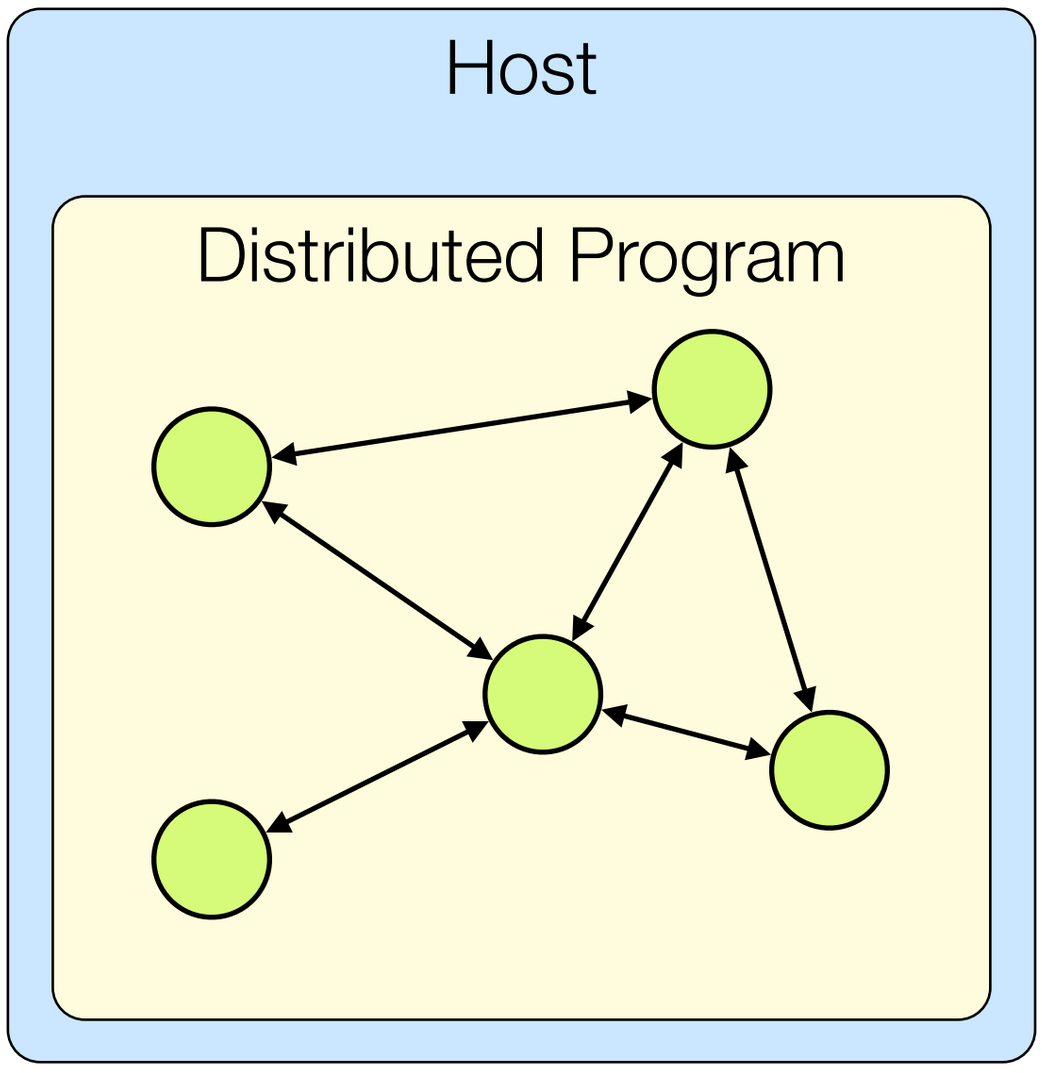
A distributed system is one on which I cannot get any work done because some machine I have never heard of has crashed.

Leslie Lamport



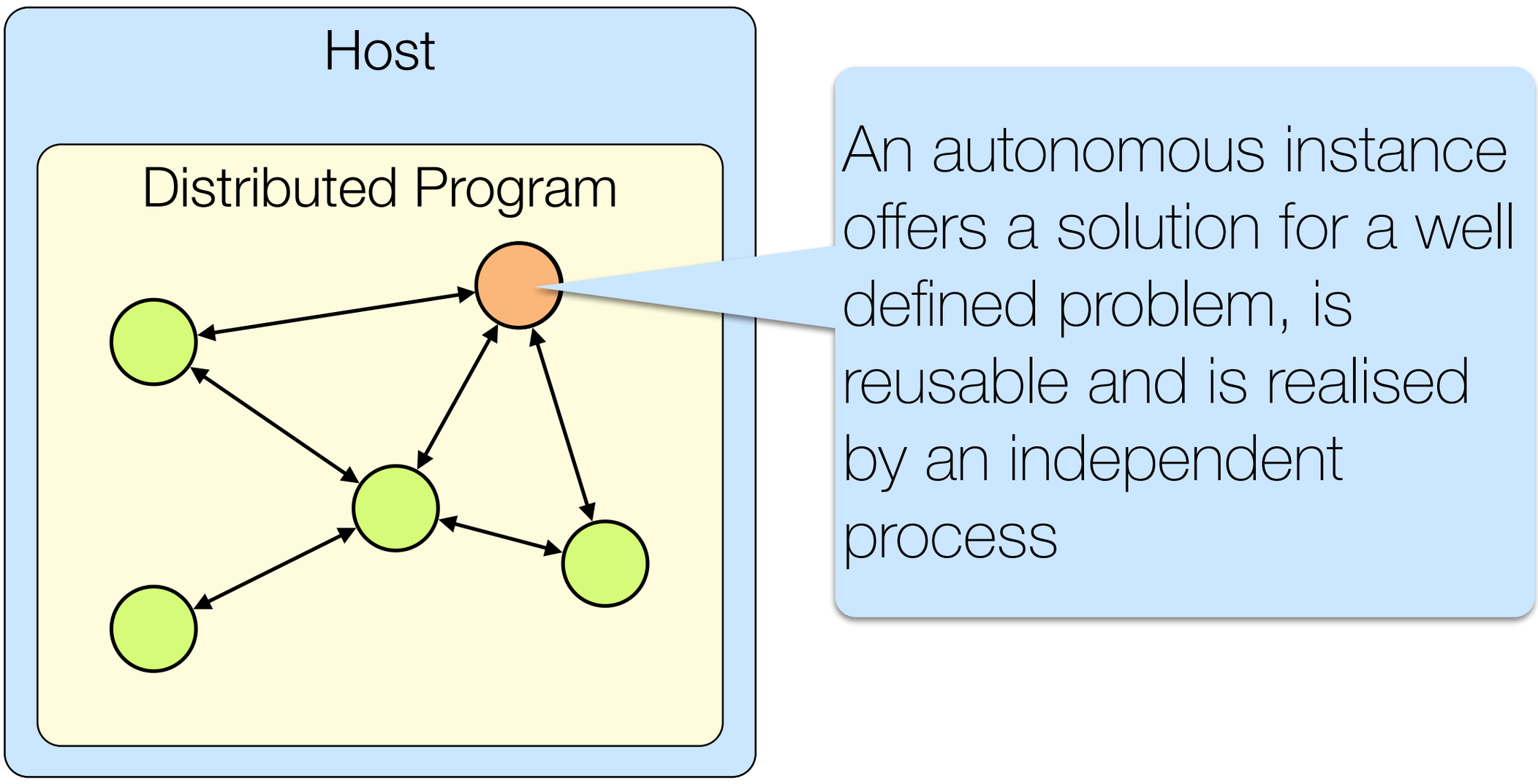
Distributed Program

- Definition: A distributed program consists of **autonomous** software instances communicating via **messages**.



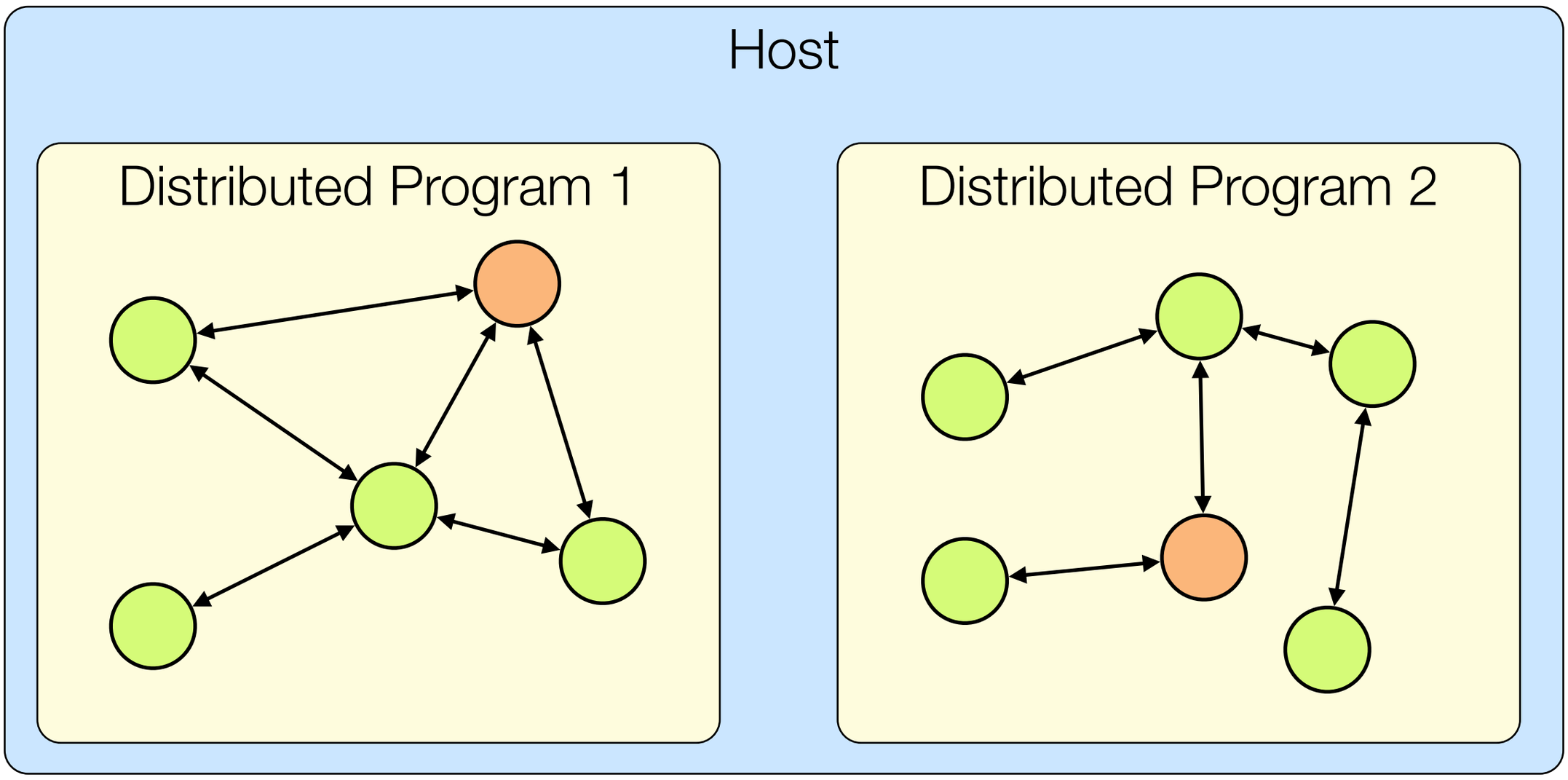
Distributed Program

- Definition: A distributed program consists of **autonomous** software instances communicating via **messages**.



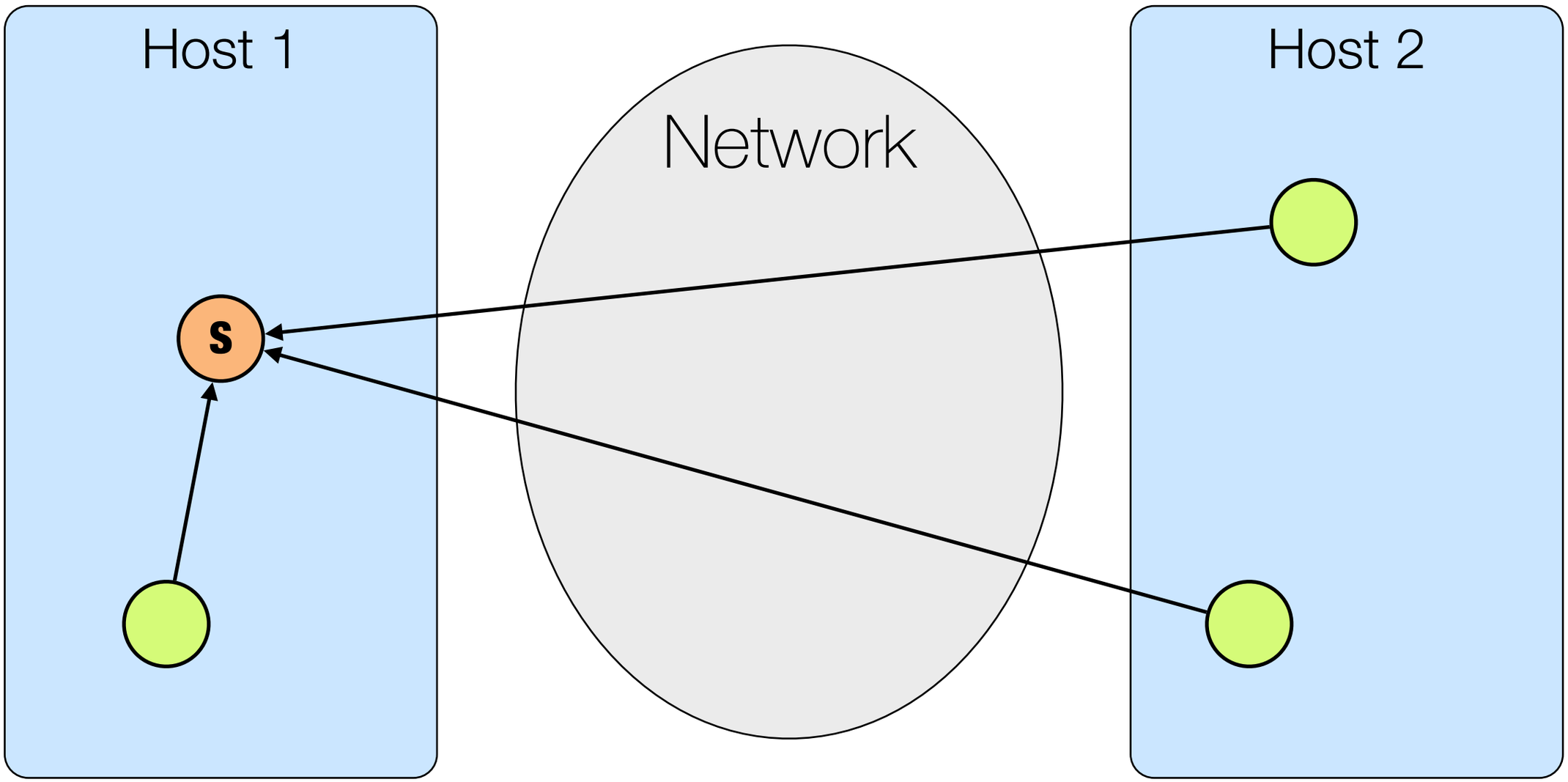
Distributed Program

- Definition: A distributed program consists of **autonomous** software instances communicating via **messages**.



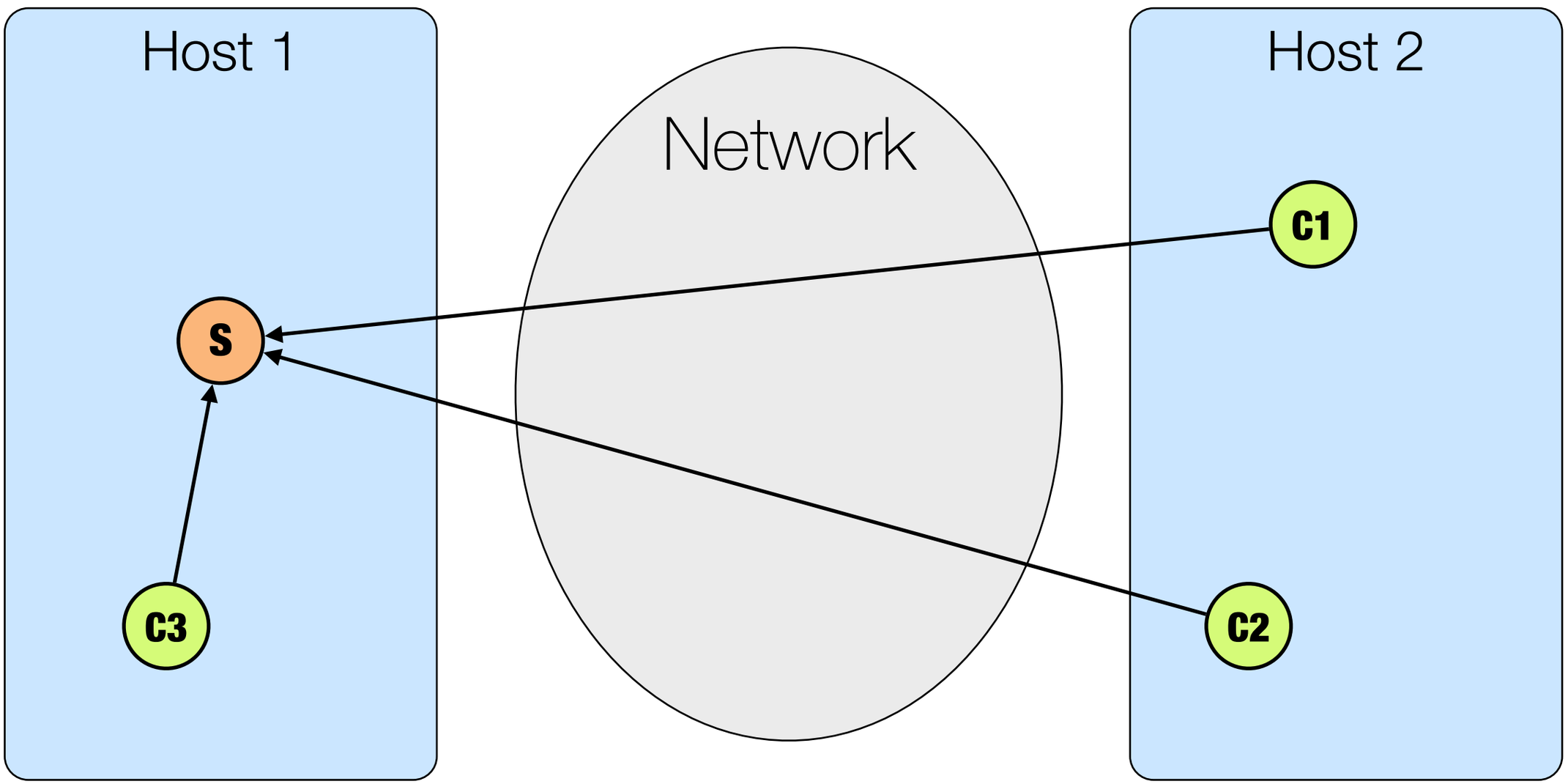
Service

- Definition: A **service** is an autonomous software instance accessible via network.



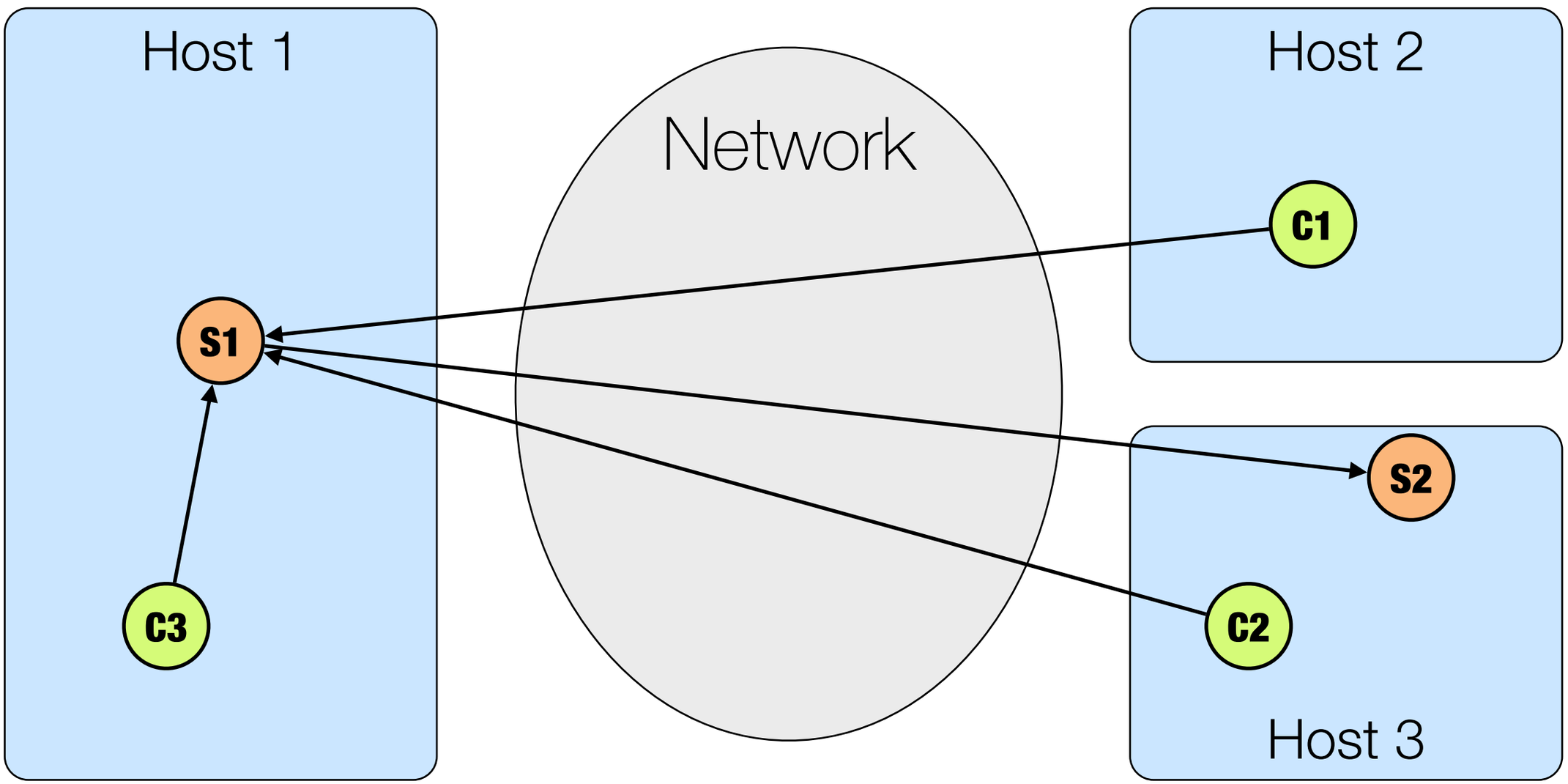
Client

- Definition: A software instance using a service is called a **client**.



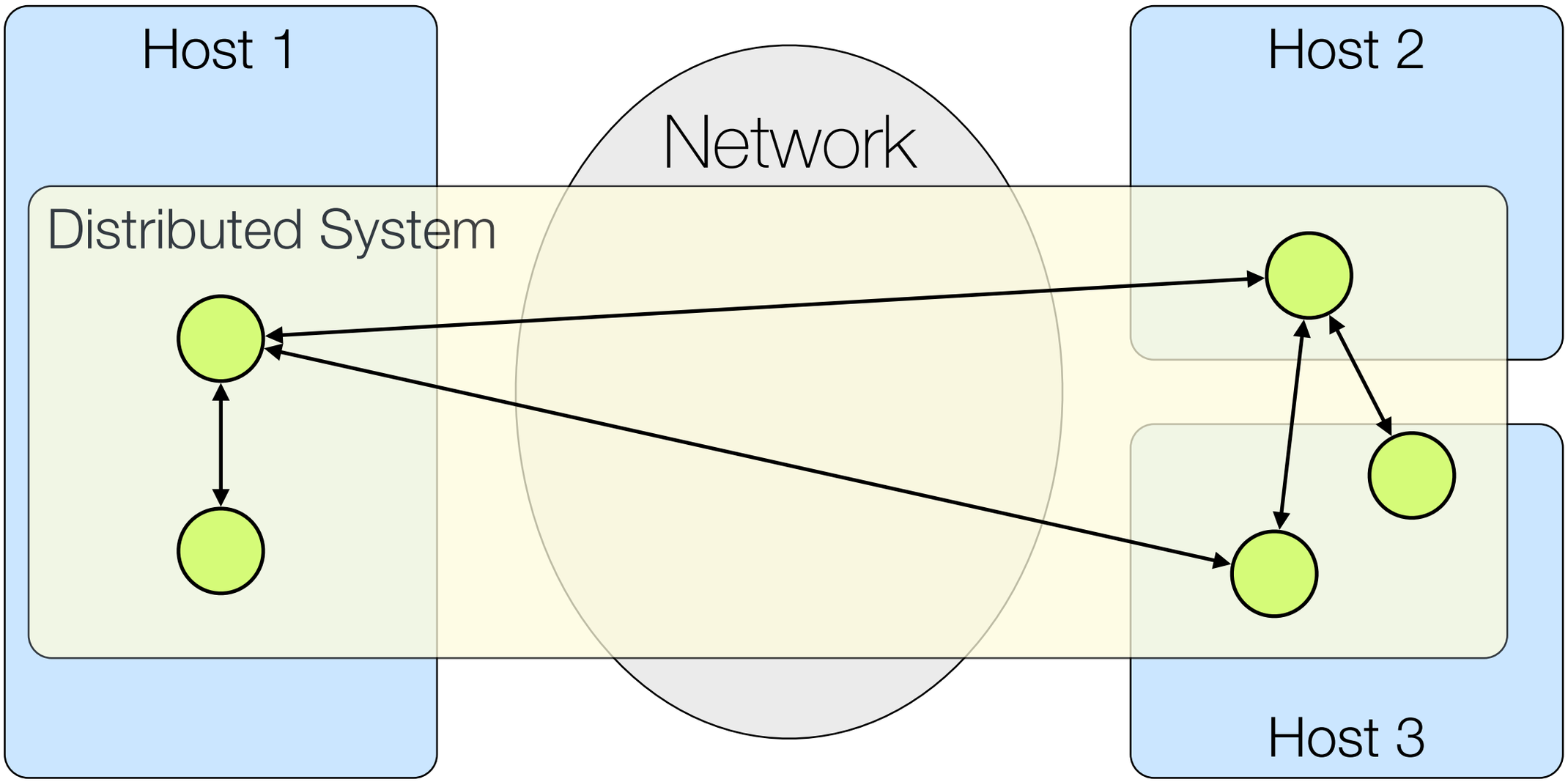
Client

- Definition: A software instance using a service is called a **client**.

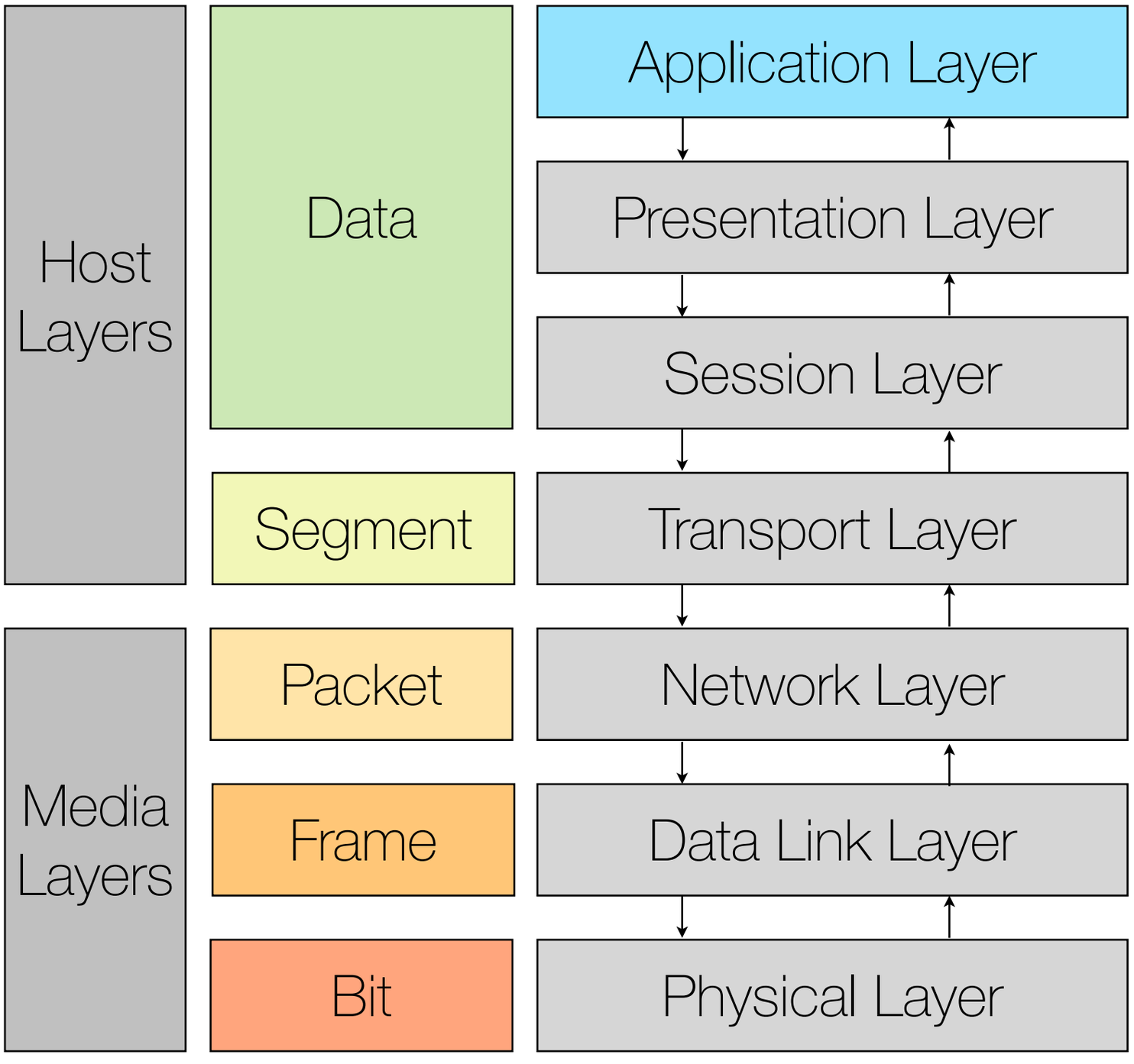


Distributed System

- Definition: A distributed system is a distributed program whose instances are distributed in a network

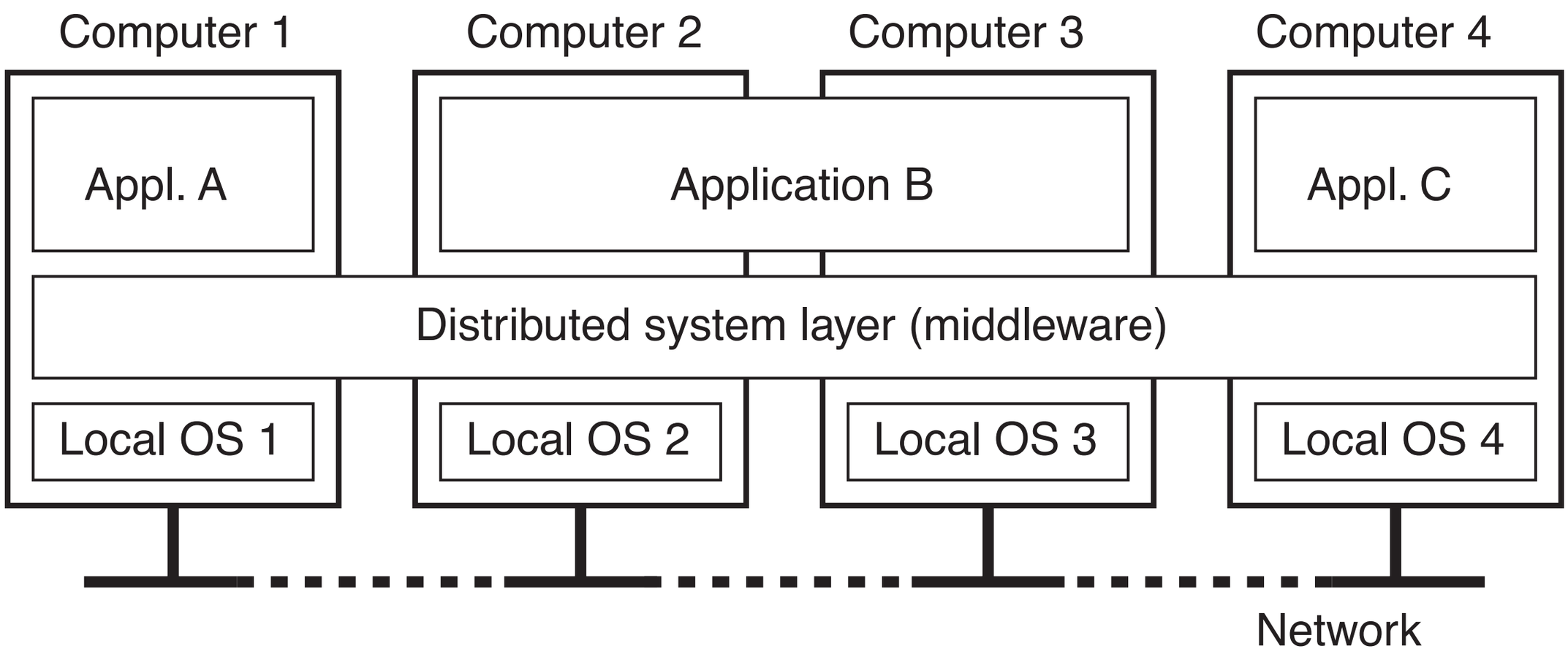


Application Layer



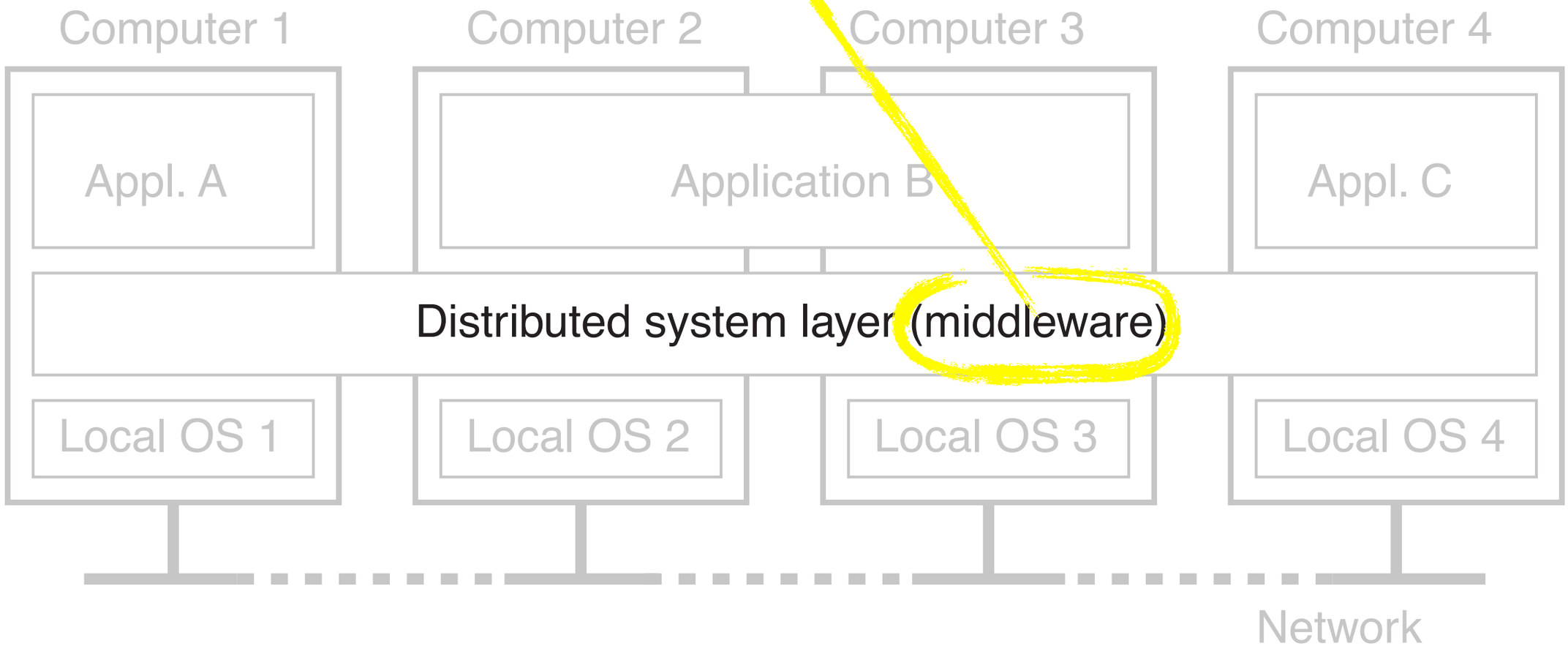
- protocols used in process-to-process communication
  - www, VoIP, imap, smtp, DNS…
- **distributed systems are implemented on the application layer**

Distributed System

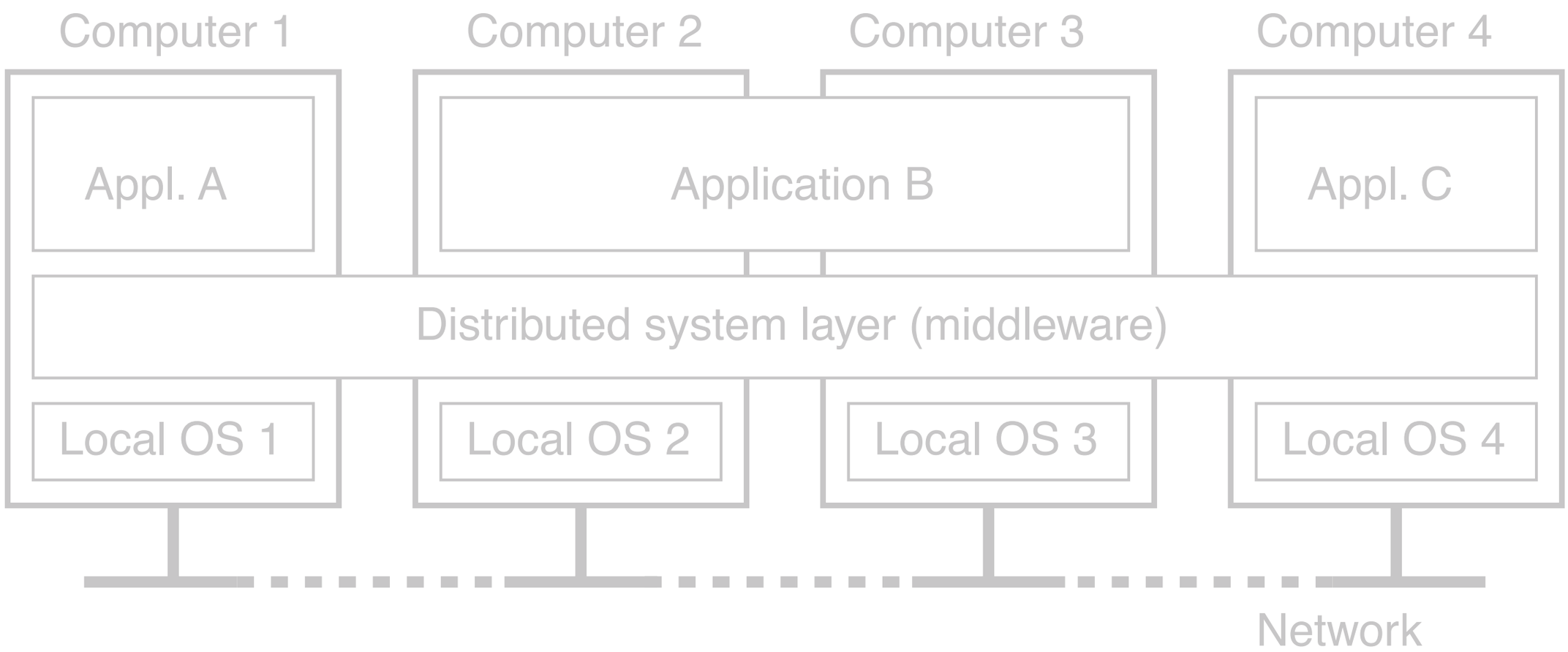


independent computers appearing as **single coherent system**

- Support distributed systems development
- Tackle heterogeneity
- Cope with different data representations



independent computers appearing as **single coherent system**



independent computers appearing as **single coherent system**

→ single system image

single system image

Transparency!

Transparency!

access	location	migration
relocation	replication	concurrency
performance	scaling	failure

Transparency!

Access transparency is about hiding differences in data representation and how a resource is accessed

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency!

Location transparency is about hiding where a resource is located

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency!

Migration transparency is about hiding that a resource might move to another location

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency!

Relocation transparency is about hiding that a resource may be moved while in use

access	location	migration
relocation	replication	concurrency
performance	scaling	failure

Transparency!

Replication transparency is about hiding that a resource is replicated

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency!

Concurrency transparency is about hiding that a resource may be used concurrently by multiple users

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency!

Performance transparency is about allowing the system to be reconfigured to improve performance as loads vary

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency!

Scaling transparency is about the system to expand in scale without change to the system structure or the application algorithms

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency!

Failure transparency finally is about hiding failures and recovery of a resource

access	**location**	**migration**
relocation	**replication**	**concurrency**
performance	**scaling**	**failure**

Transparency

?

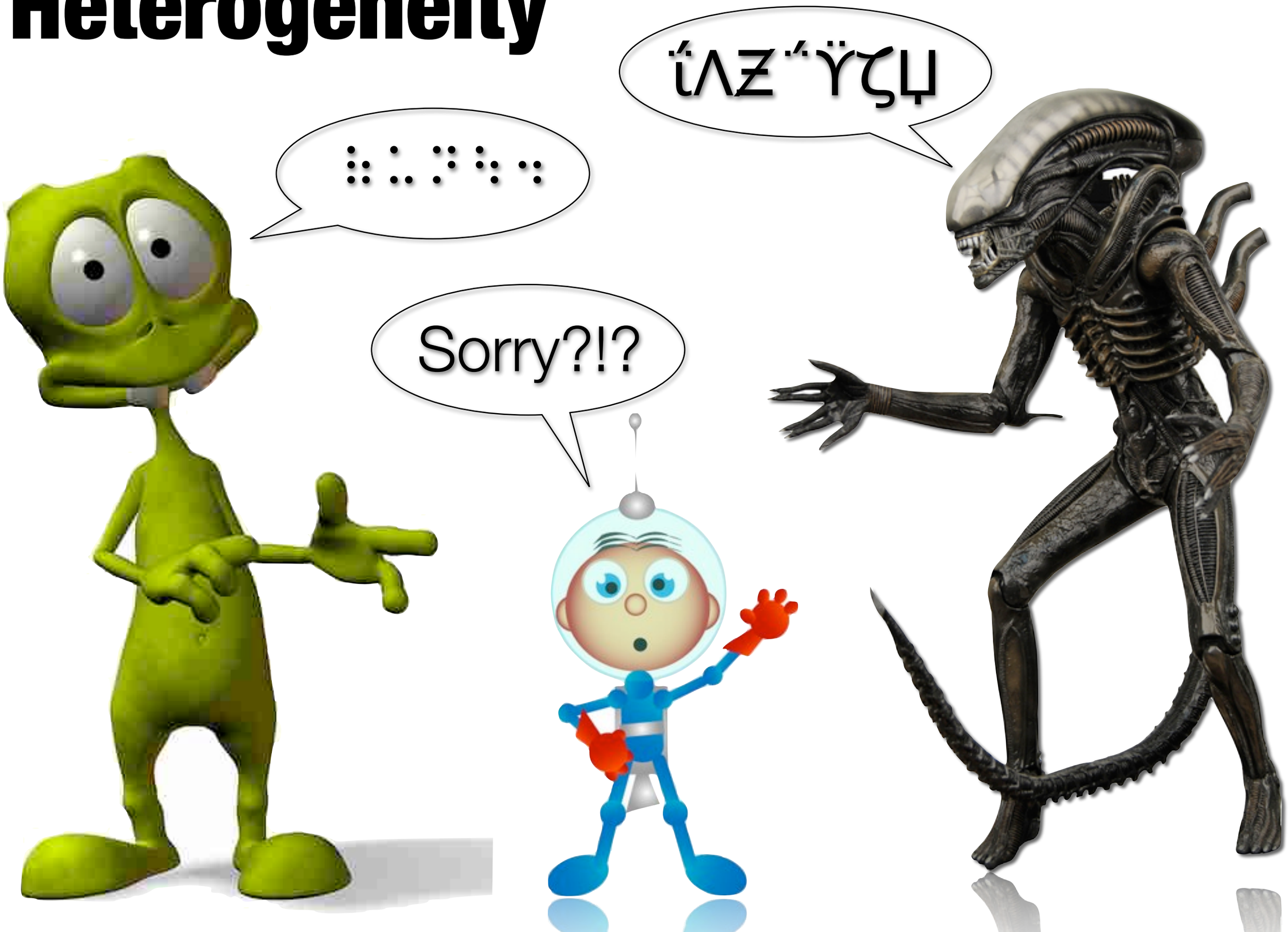
access	location	migration
relocation	replication	concurrency
performance	scaling	failure

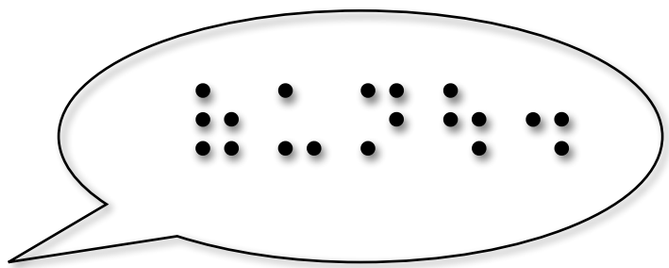
Hiding Failures
Virtually **Impossible**

Hiding Failures
Virtually **Impossible**

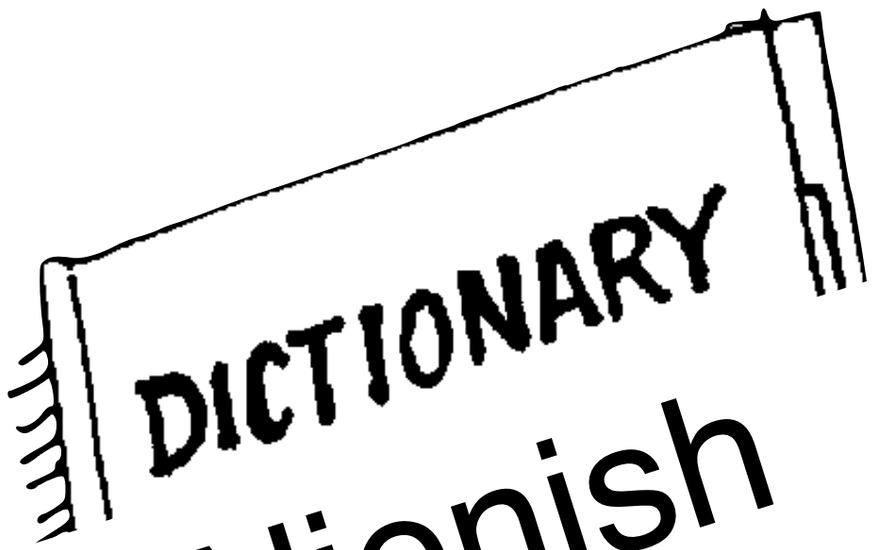
remote machine
just **slow**
or **crashed**
?

Heterogeneity



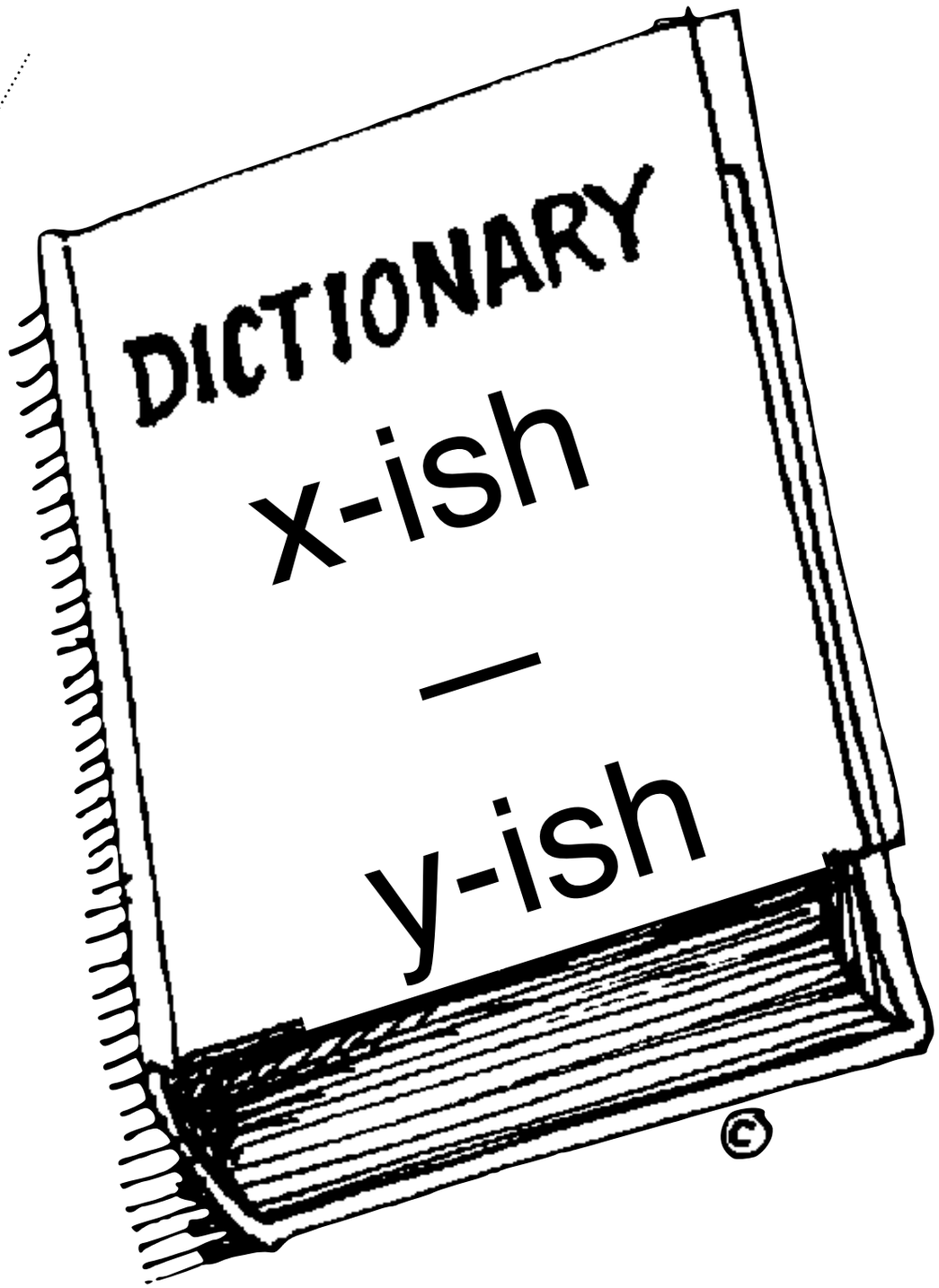


Sorry?!?



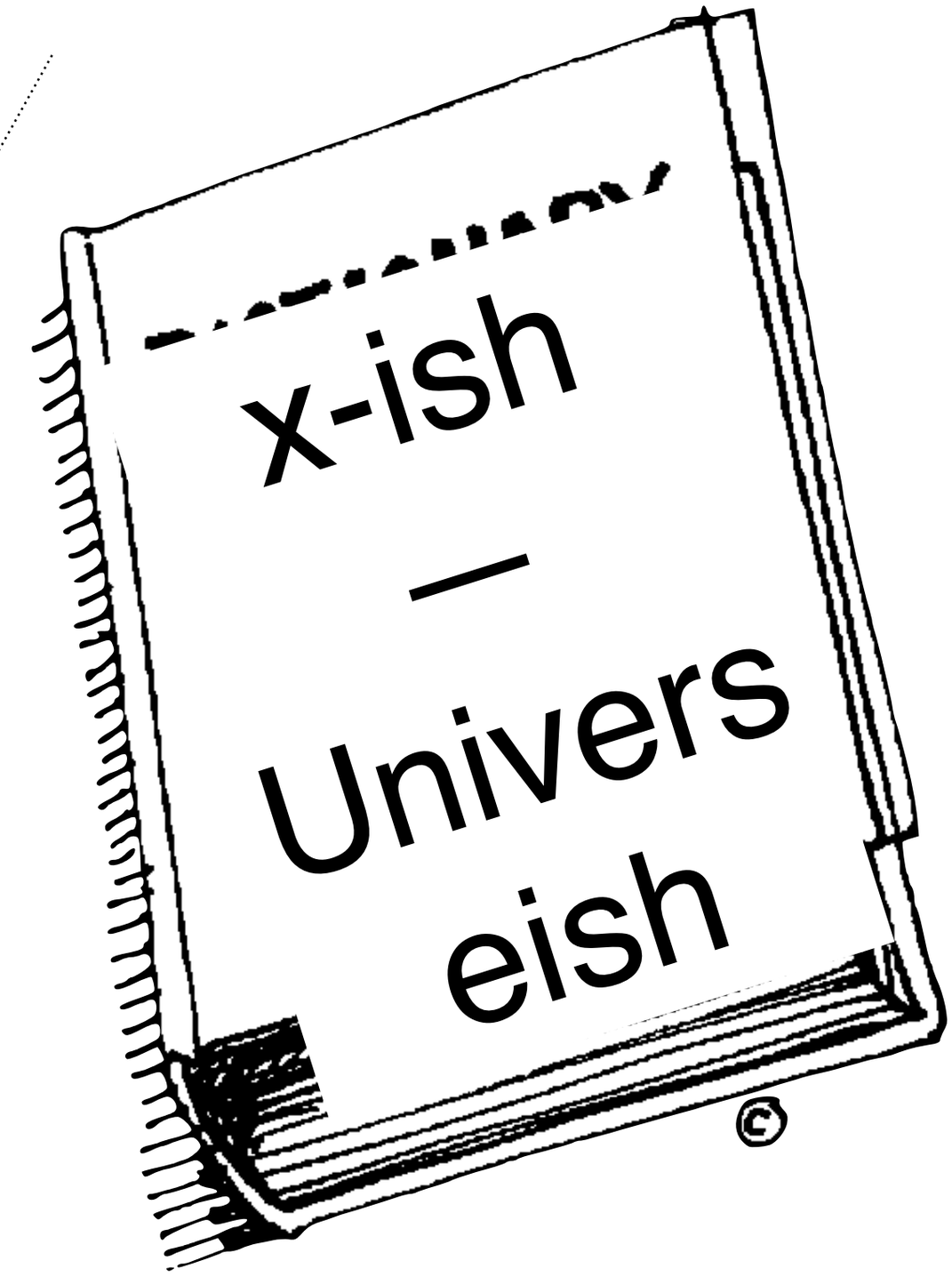
Openness

	Saturnish	Marsianish	Uranusish	Venusish	Jupiterish	...
Saturnish	·	✓	✓	✓	✓	✓
Marsianish	✓	·	✓	✓	✓	✓
Uranusish	✓	✓	·	✓	✓	✓
Venusish	✓	✓	✓	·	✓	✓
Jupiterish	✓	✓	✓	✓	·	✓
...	✓	✓	✓	✓	✓	·



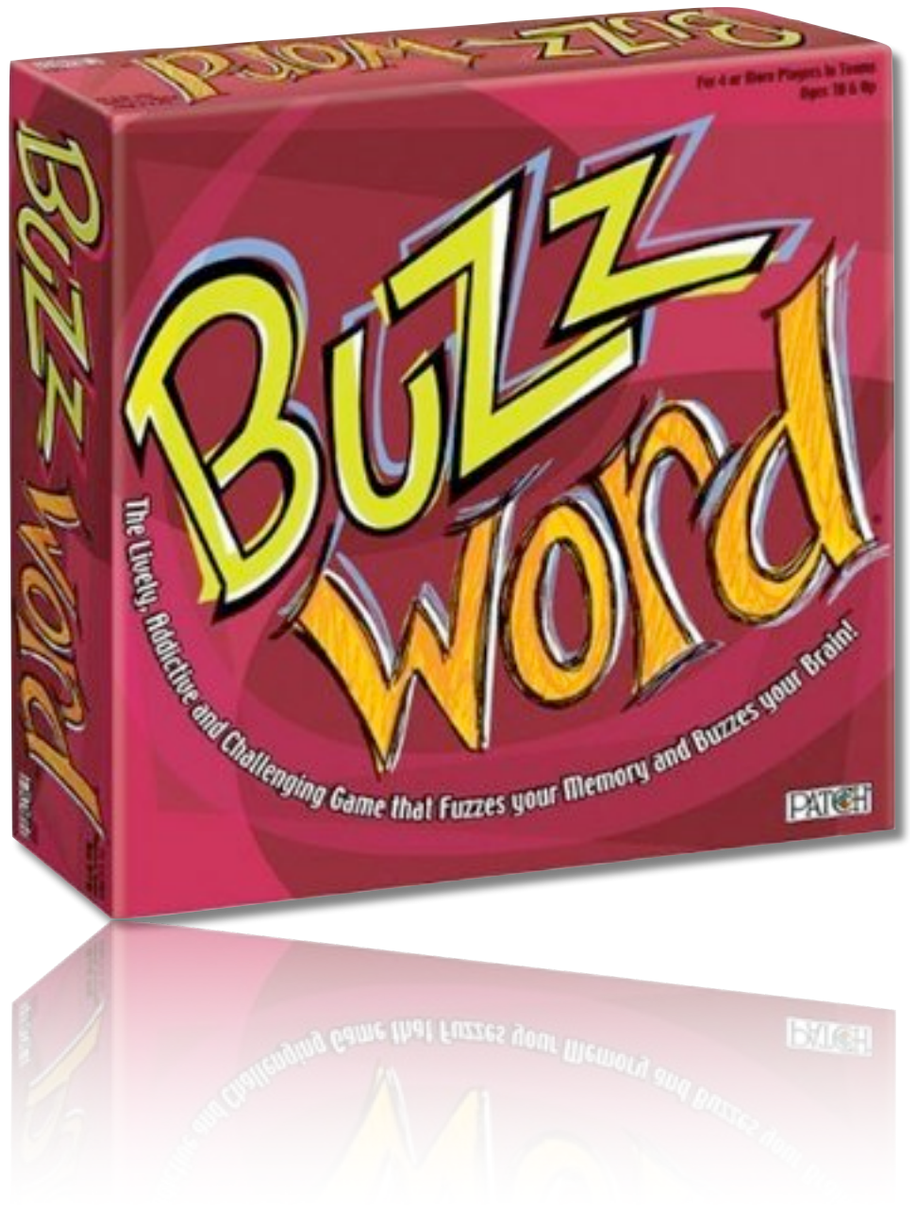
$O(n^2)$?!? Interoperability

	Saturnish	Marsianish	Uranusish	Venusish	Jupiterish	...
Universeish	✓	✓	✓	✓	✓	✓



O(n)

Marshalling



Scalability

size

geographical

administrative

Scalability

size

Size scalability deals with the ability of (easily) adding additional users and resources to the system

geographical

administrative

Scalability

size

In a **geographical scalable** system, users and resources might lie far apart

geographical

administrative

Scalability

size

geographical

administrative

Administrative scalability
is given if a system can be (easily) managed even if it spans multiple administrative domains

Scalability

Addressing Scalability . . .

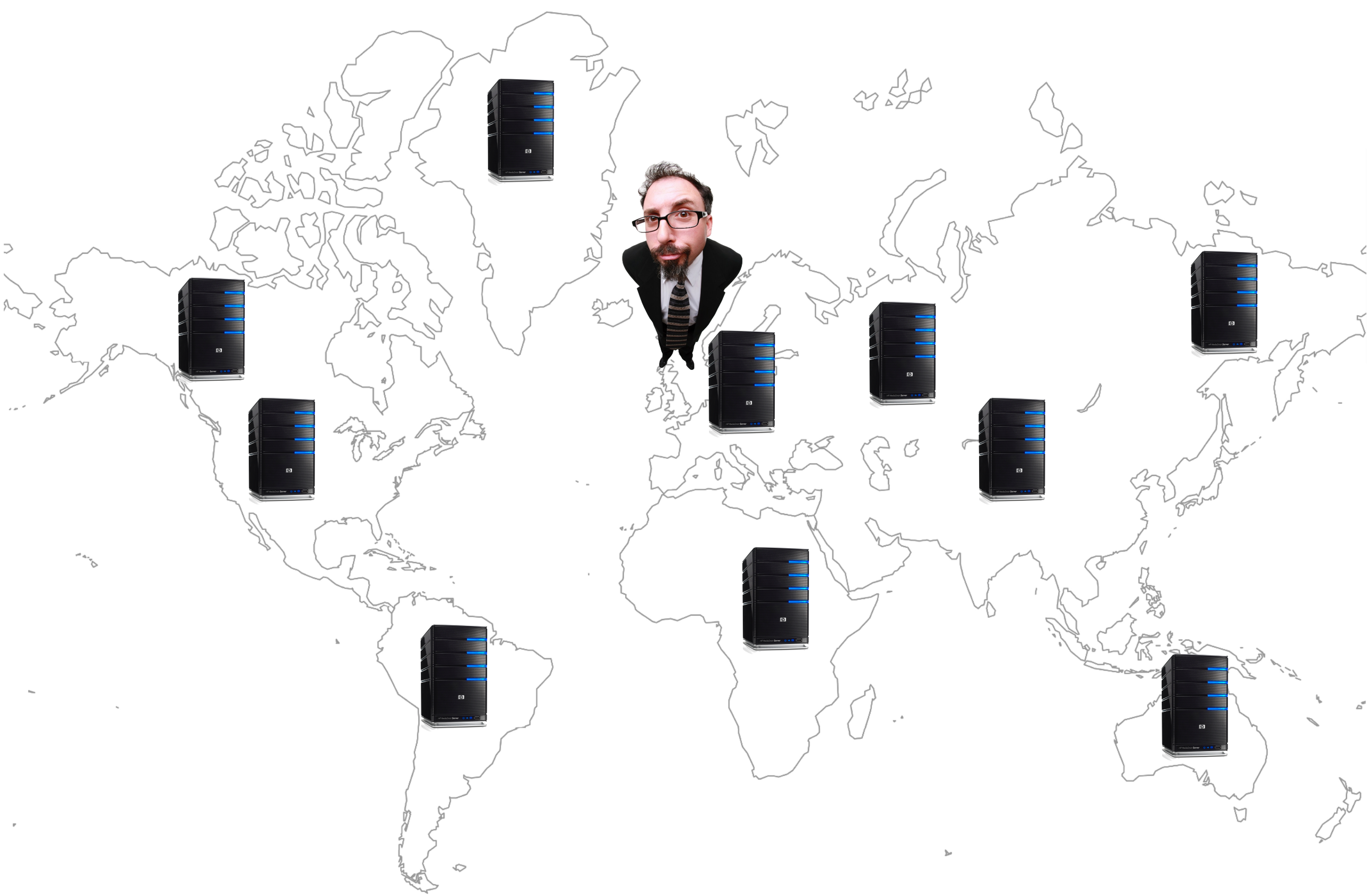
Scalability **Non-Solution**

How to Address Scalability?

Distribute the **Load**

Who Serves Me?

Watch Out for Latency!

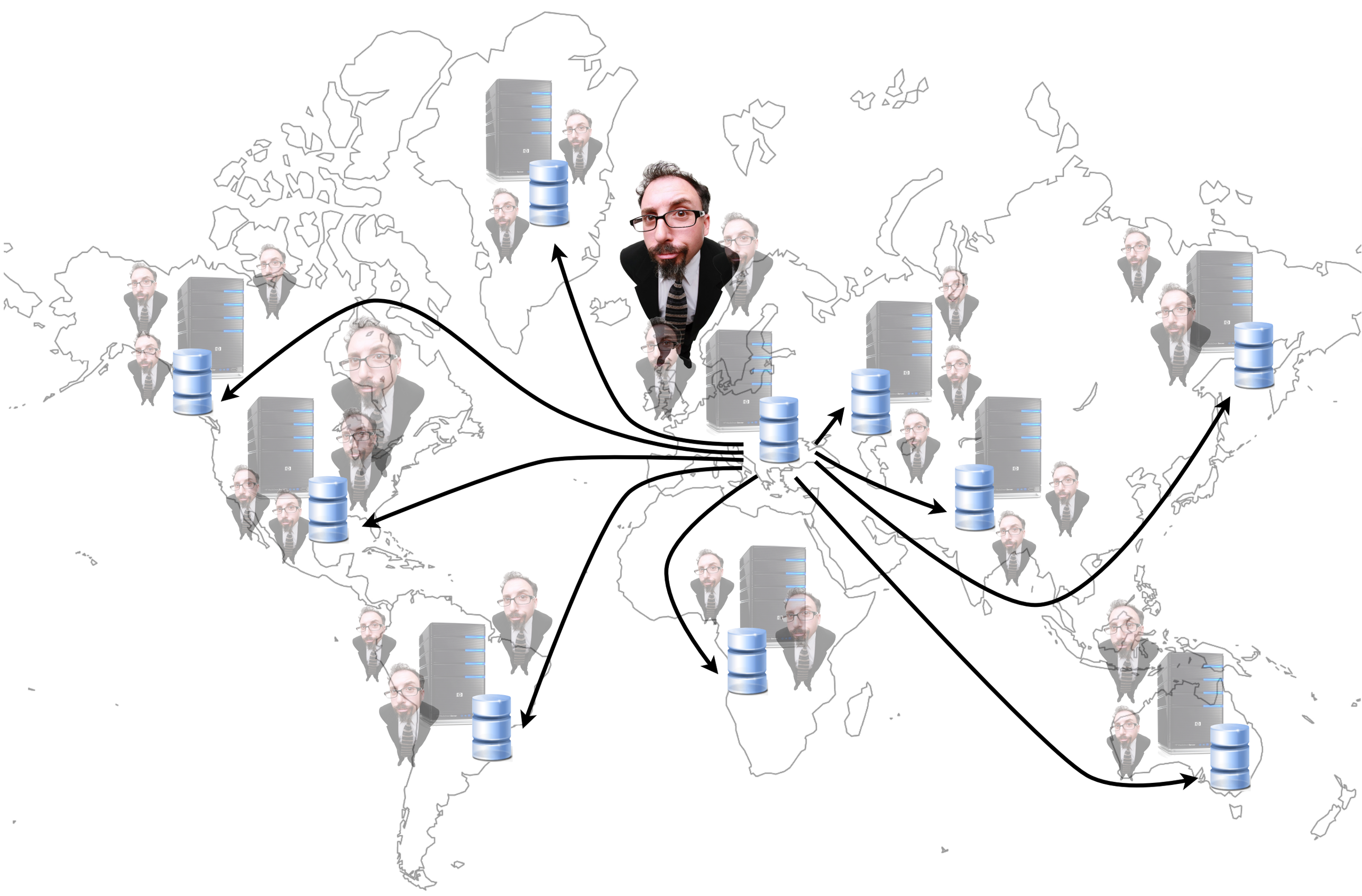


Topology-wise **Closest!?**

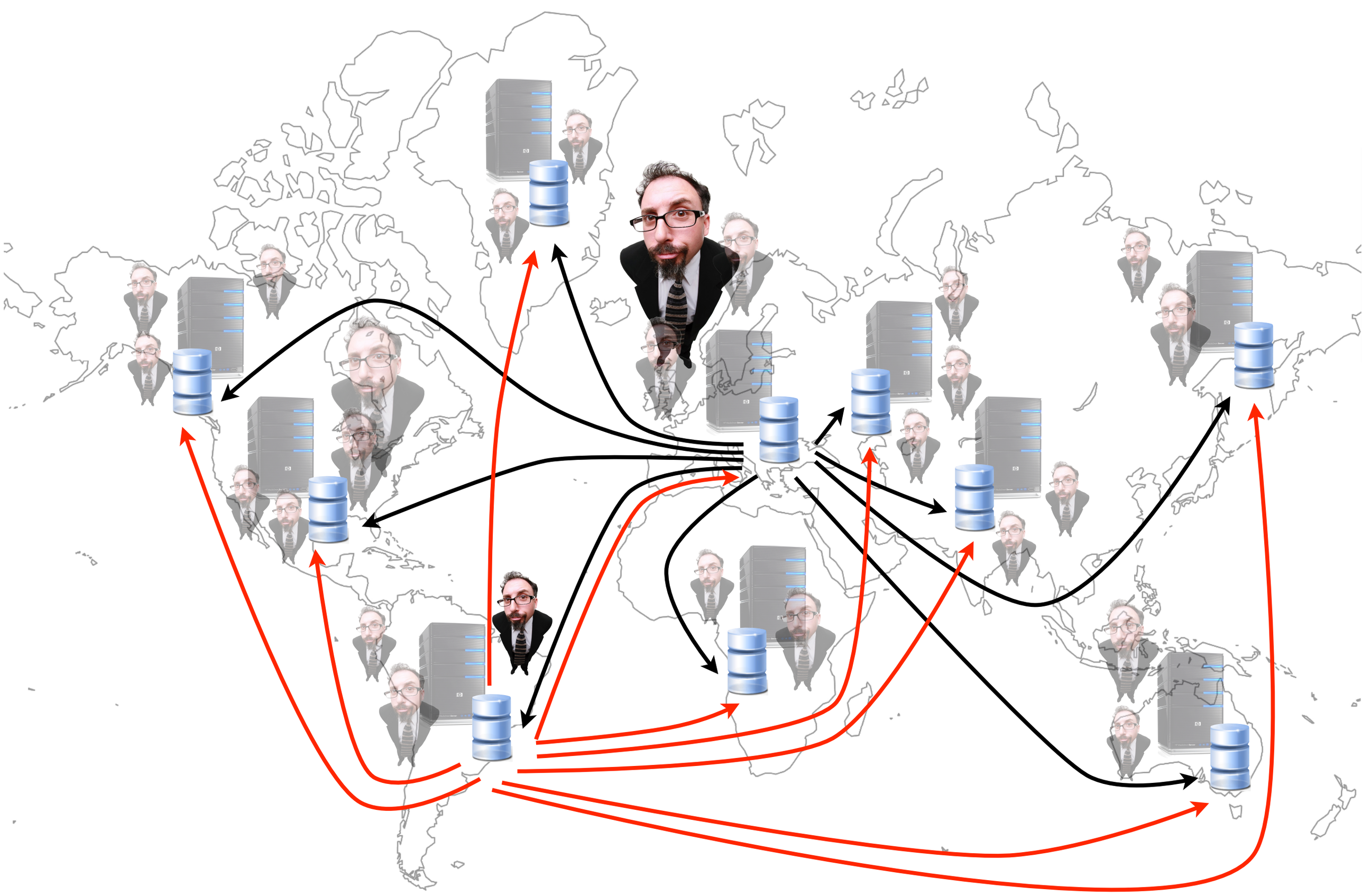
Maybe **Not** Just **Closest!**

Load-Balancing!

Load-Balancing Requires **Replication**

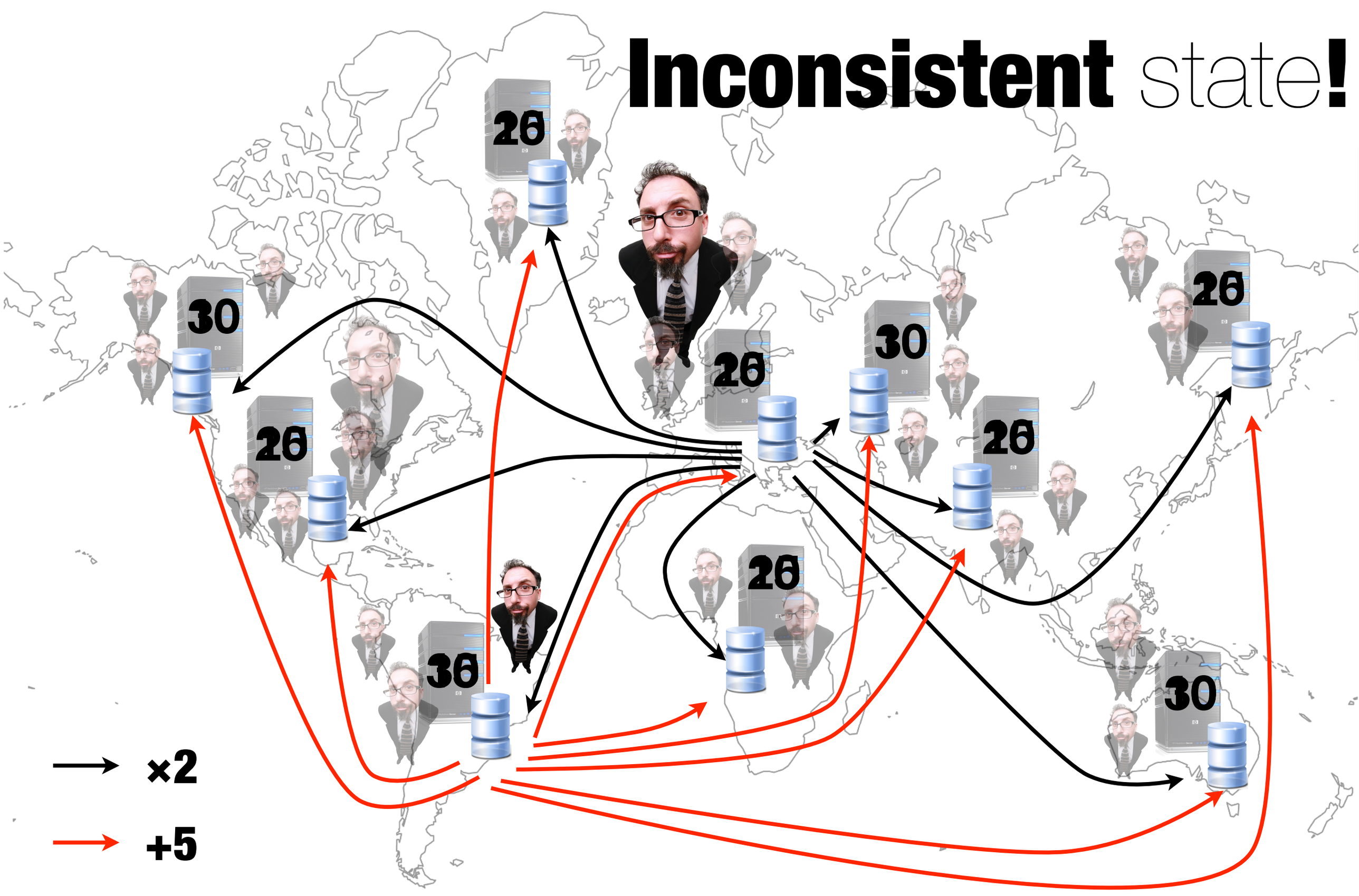


Replication Hampers **Scalability**



What about **Concurrent Updates?**

Inconsistent state!



Example

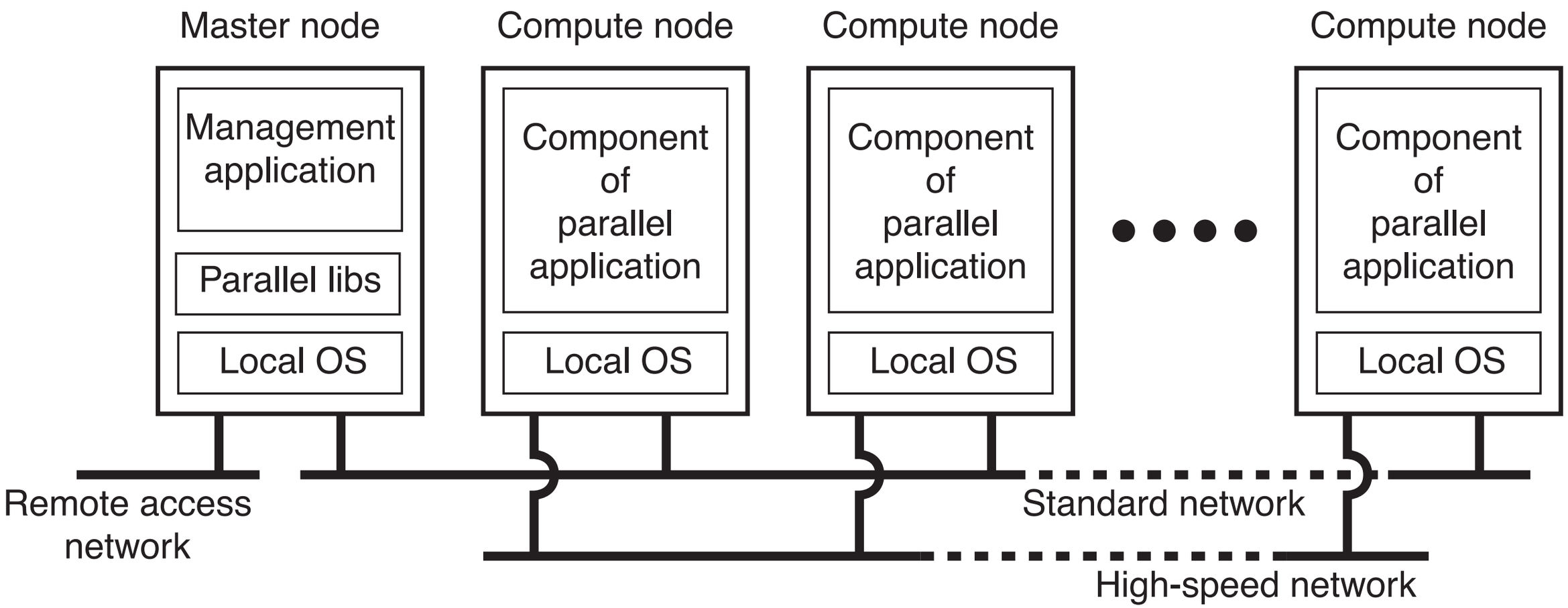
Network is **reliable**, **secure** & **homogeneous**
Topology does **not change**
Latency is **zero**
Bandwidth is **infinite**
Transport cost is **zero**
There is only a **single administrator**
and many more **false assumptions!**

Typical **Pitfalls**

Types of Distributed Systems

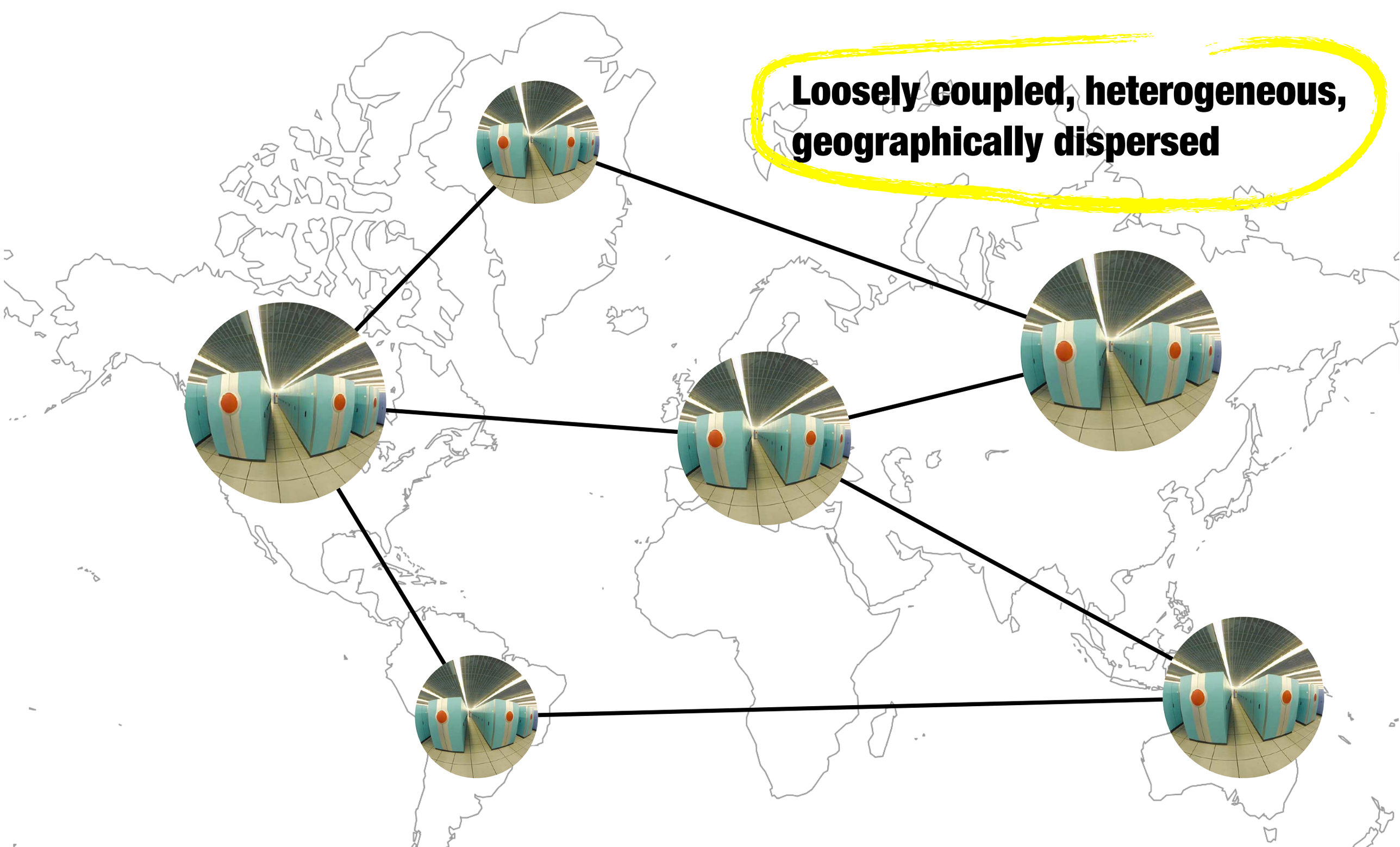
- Distributed ***Computing*** Systems
  - Distributed computation load across multiple machines
- Distributed ***Information*** Systems
  - Provide information from various, heterogenous information sources
- Distributed ***Pervasive/Embedded*** Systems
  - Ad-hoc interconnection of mobile devices with wireless communication facilities

Distributed Computing Systems



Clusters are pretty useful when it comes to **parallel programming** and **high-performance computing**

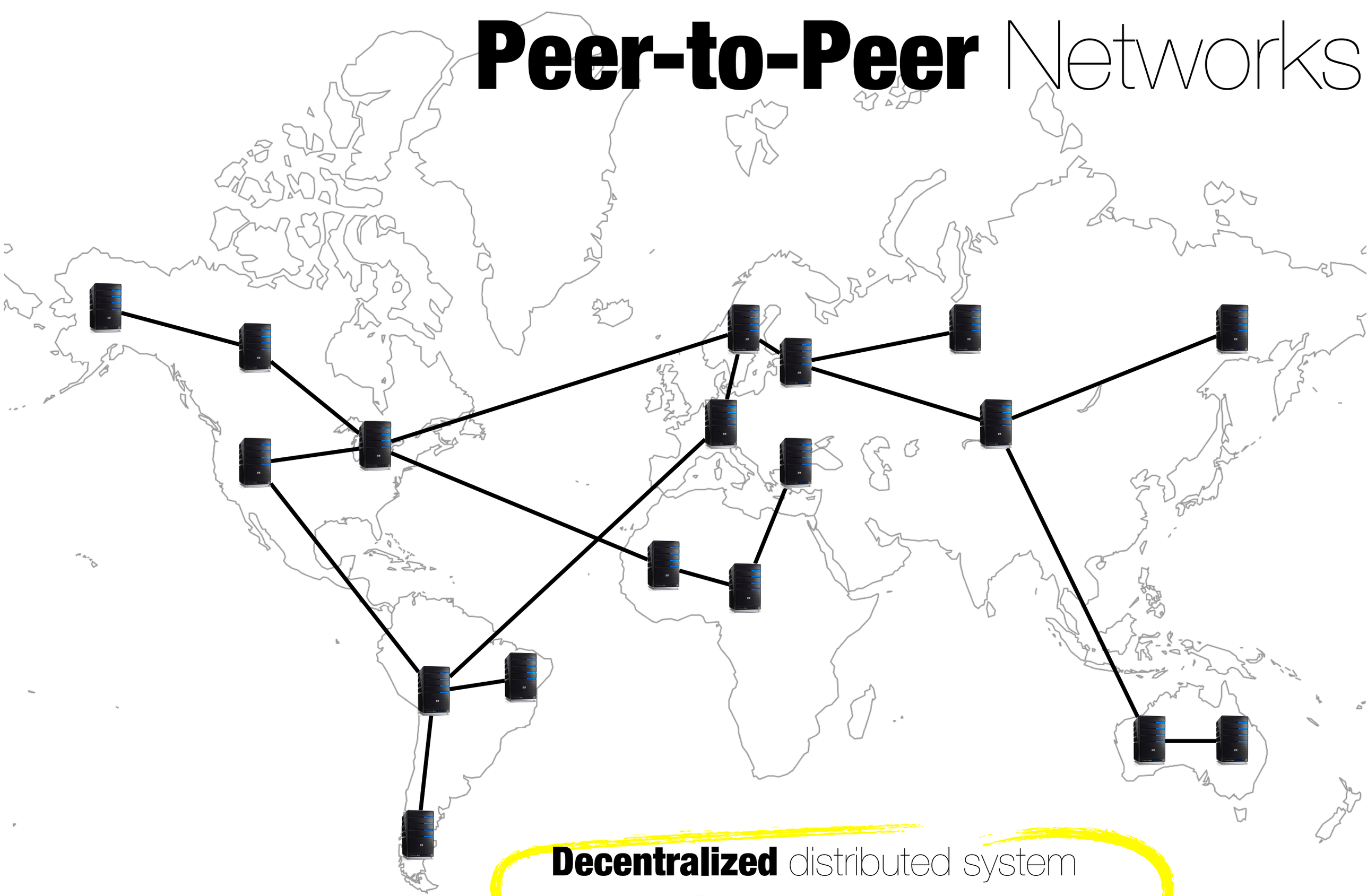
High Performance Computing — **Clusters**



HPC at Global Scale Spanning

Multiple Administrative Domains — **Grids**

Peer-to-Peer Networks



Decentralized distributed system
in which **Peers** act as resource
supplier and **consumer**

Cloud **Computing**

- **Definition**: on demand provision of compute resources via internet

User

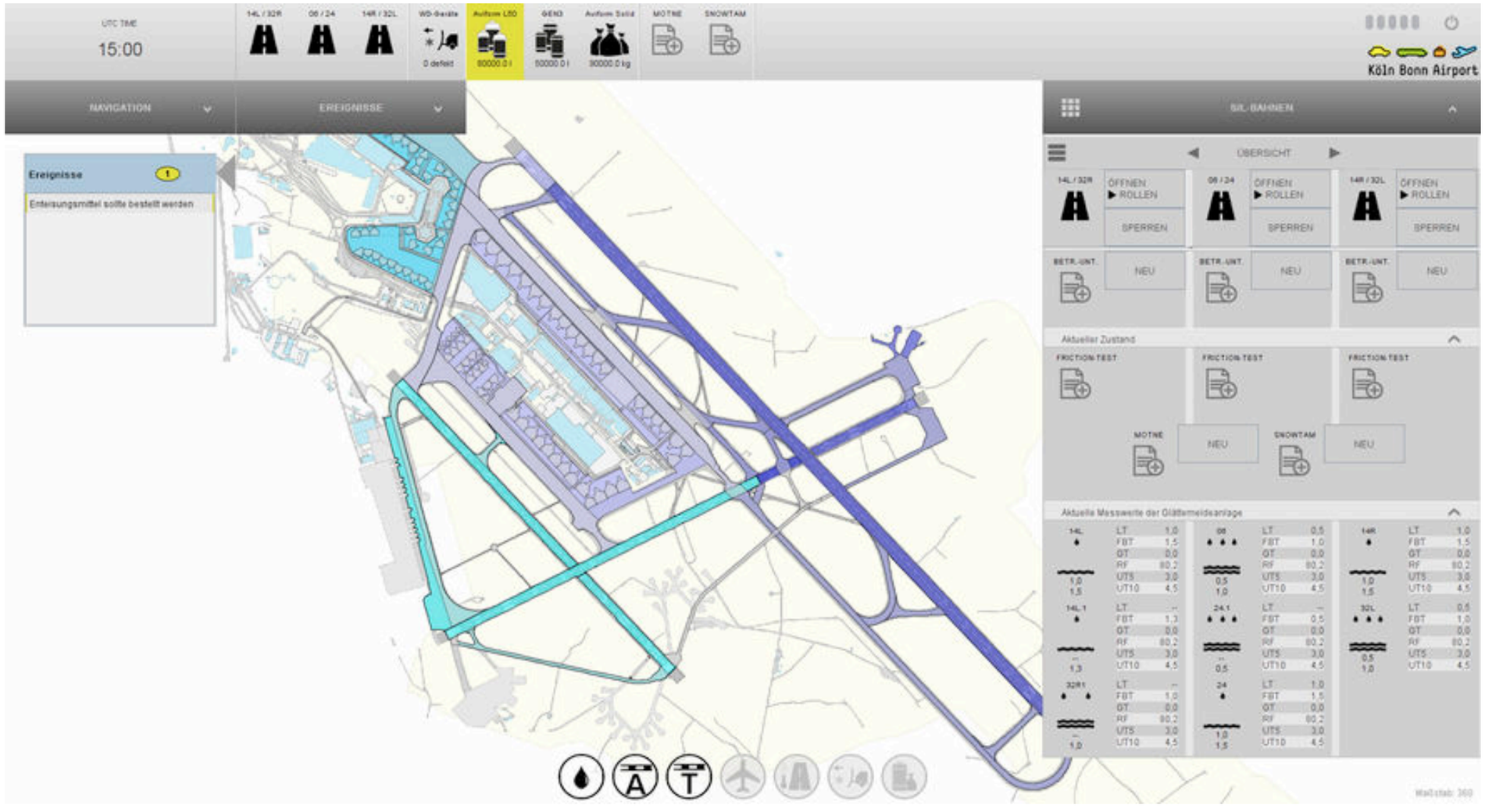
Internet

Cloud-Provider

Distributed **Information** Systems

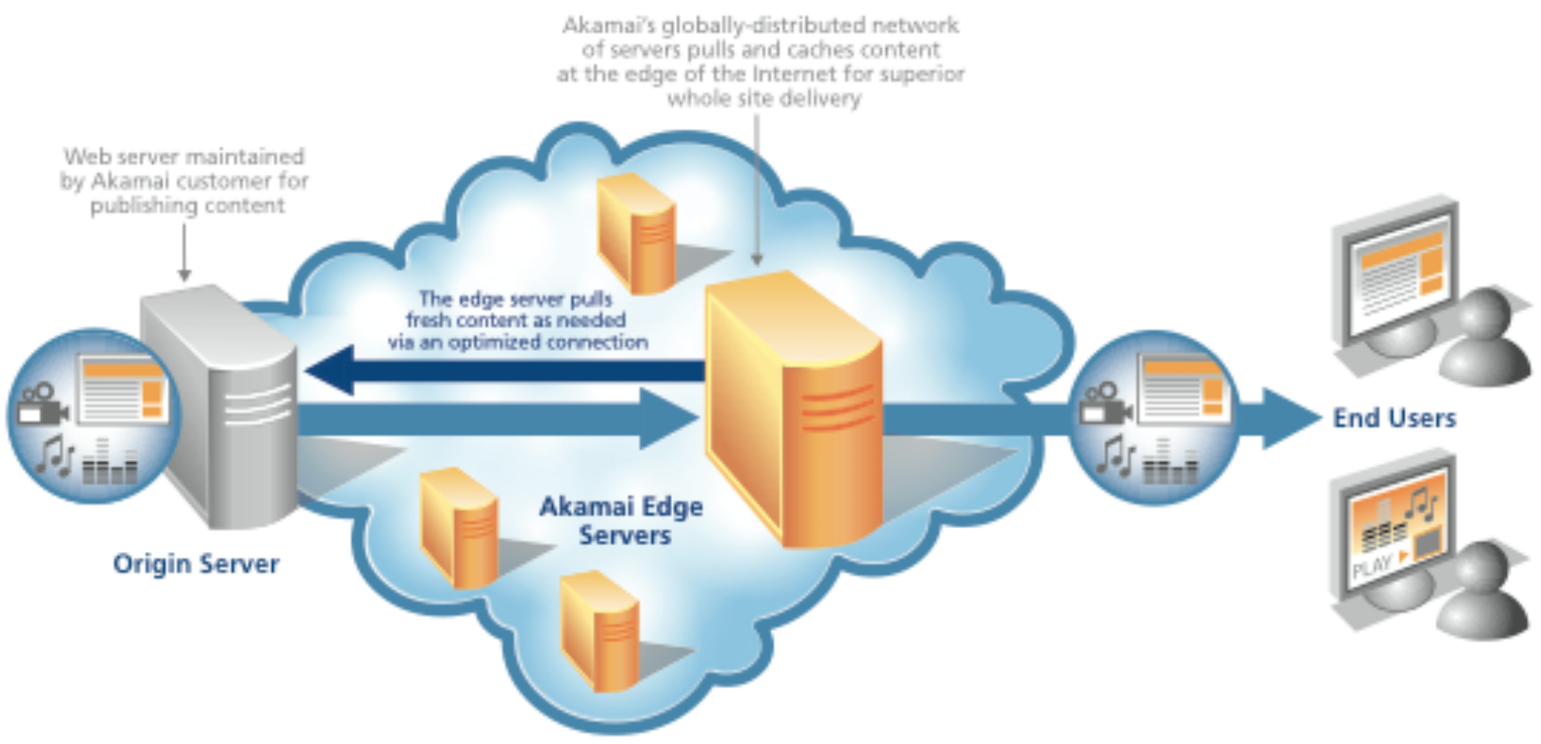
Distributed **Information** Systems

- Examples:
  - distributed situational awareness system for airport winter service



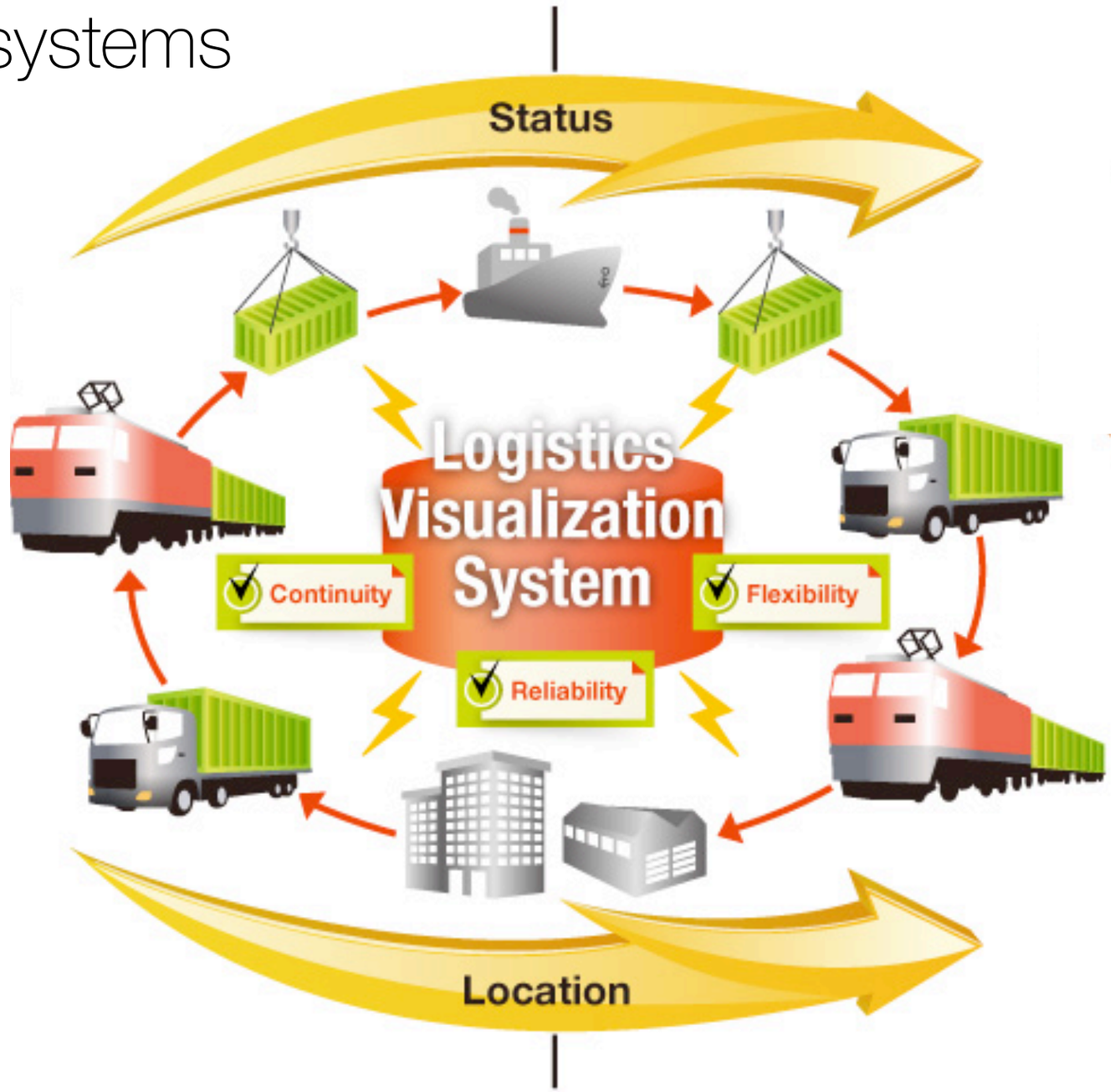
Distributed **Information** Systems

- Examples:
  - Akamai



Distributed **Information** Systems

- Examples:
  - logistics information systems



Distributed **Information** Systems

- Examples:
  - enterprise resource planing systems
    - integration of various enterprise information systems

Distributed Pervasive Systems

Mobile Computing

Jane – Default Simulation

Simulation Visualization Screenshot

Search

Views Stats

Default/Plan

Perspective

Zoom

Rotate

Move

time 8.0000

1/1024x 1/256x 1/64x 1/16x 1/4x 1x 2x 4x 8x 16x 32x 64x 128x 256x

Console

Mobile **Ad-Hoc** Networks

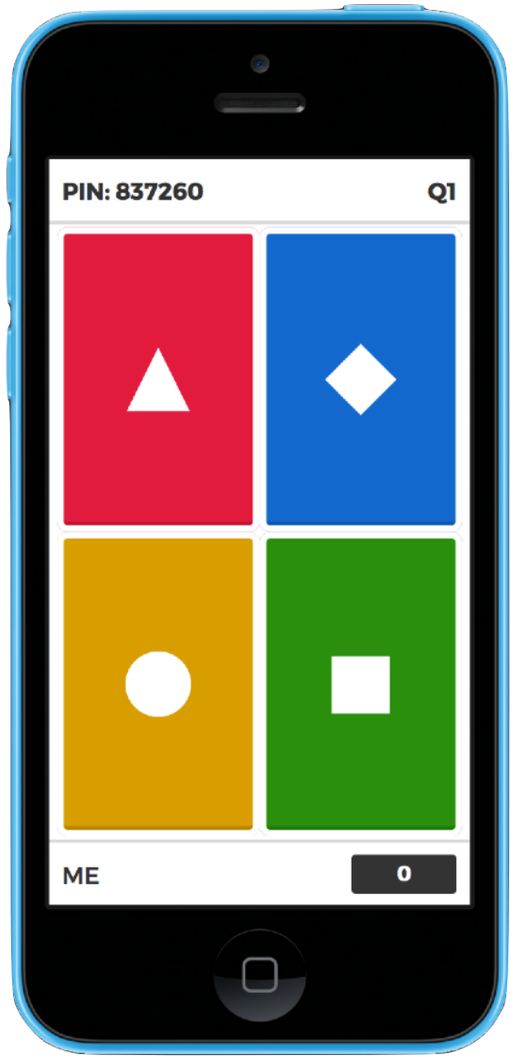
Kahoot!

go to
kahoot.it

PIN
eingeben

Nickname
eingeben

mitspielen



1 2 3 4