

# Distributed Systems

## Intro - Part I

---

Vero Estrada-Galiñanes, PhD

# Internet cables



**‘People think that data is in the cloud, but it’s not. It’s in the ocean.’**

<https://web.archive.org/web/20200107194521/https://www.nytimes.com/interactive/2019/03/10/technology/internet-cables-oceans.html>

How?

Logistics, Policies, Grading...

Where?

Study Resources

Why?

Intro to Distributed Systems (DS)

What?

Syllabus - Main topics

Why do you want to learn  
distributed systems?

## Knowledge

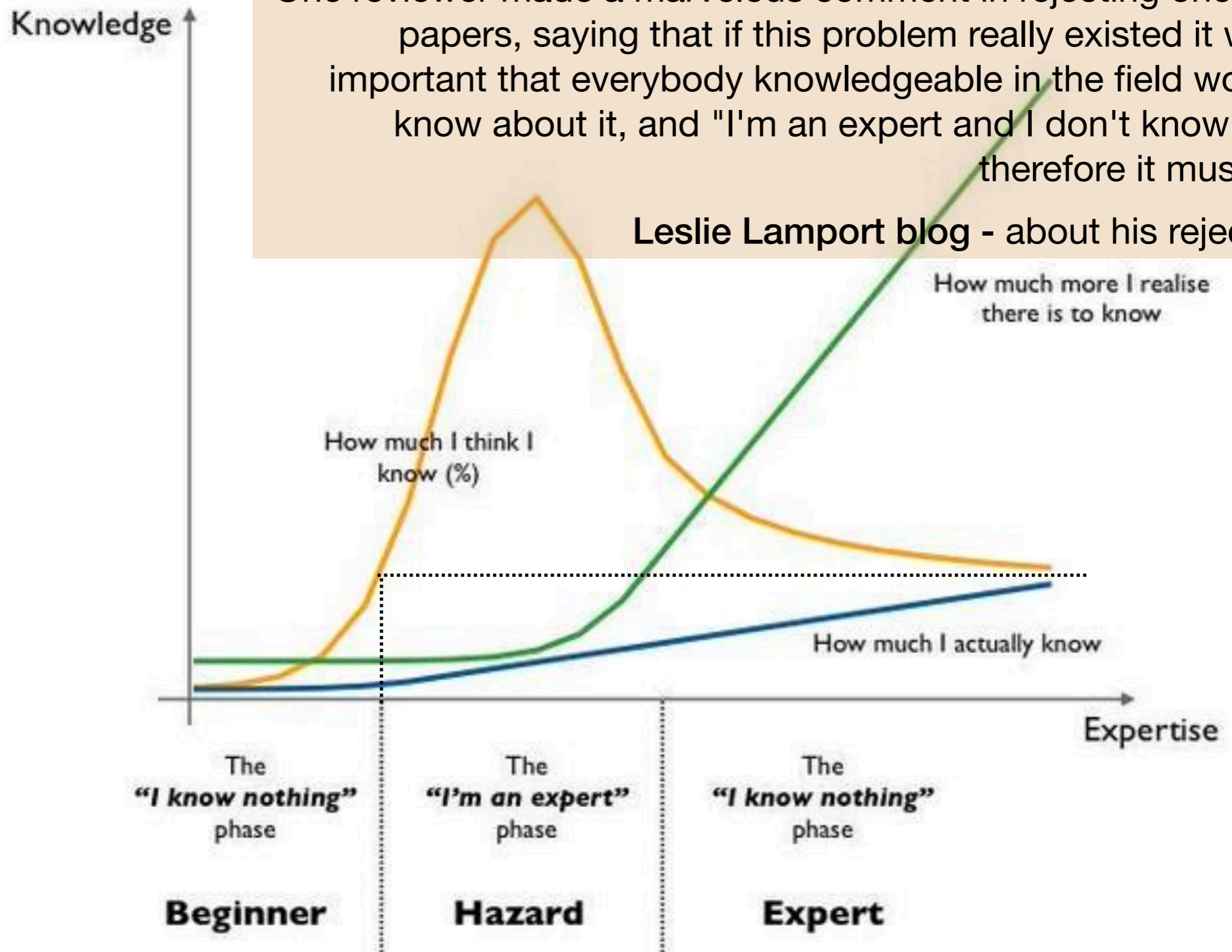
*Be familiar with*

- important principles for design and implementation of distributed systems.
- the most important system architectures for distributed systems.
- important techniques for solving various problems in distributed systems.

# “Real knowledge is to know the extent of one's ignorance.” Confucius

One reviewer made a marvelous comment in rejecting one of the early papers, saying that if this problem really existed it would be so important that everybody knowledgeable in the field would have to know about it, and “I'm an expert and I don't know about it, so therefore it must not exist.”

Leslie Lamport blog - about his rejected papers



## Skills

*Be able to*

- develop advanced distributed applications with fault tolerance properties.
- perform independent research in distributed systems.
- reason about problems that involve distributed components.

**Labs**

**Scientific Activities**

**Lectures hands-on  
Labs**

## General competency

*Know how to*

- develop distributed computer systems.

## Introduction (Part I & II)

1. Abstractions
2. Communication
3. Coordination
4. Availability, consistency and ACID properties
5. Fault tolerance and reliability
6. Consensus
7. Naming
8. Security
9. Design and operations
10. Evaluation

Labs + Research activities



Logistics, Policies,  
Grading...

## Canvas (next week)

- Course information
- Syllabus
- Schedule and Lecture Plan
- Lab project information
- Reading material (links)
- Announcements

## Recommended prerequisites

- Operating Systems
- Computer Networking
- Advance Programming

## Labs

- Golang
- Github
- Docker

**Playground:** <https://play.golang.org/>

**The tour:** <https://tour.golang.org/welcome/4>

**Video 7h without adds:** <https://www.youtube.com/watch?v=YS4e4q9oBaU>

## Communicate effectively (scientific writing)

**Writing skill lessons (45 min each):**

<https://cgi.duke.edu/web/sciwriting/index.php>

## Self-study class/Subject to change

WEEK	TUESDAY	WEDNESDAY	THURSDAY
(2) 7.Jan	NO CLASS	Introduction	
(3) 14.Jan	Architectures	Abstractions	Unsupervised LAB
(4) 21.Jan	Comunication	Coordination	Deadline LAB 1
(5) 28.Jan	Availability, consistency and ACID properties		Deadline LAB 2
(6) 4.Feb	Fault-Tolerance & Reliability		
(7) 11. Feb	Guest lecture: Hein Meling (Paxos - Raft)		
(8) 18.Feb	Consensus (Tuesday go to D-205 room)		Deadline LAB 3
(9) 25. Feb	Guest lecture (not confirmed)	Naming	
(10) 3.Mar	Self-study class	CS Talks   Presentation Tips	
(11) 10.Mar	Student: Paper presentations	Security	
(12) 17.Mar	Miscellaneuous	Students: Lightning talks	Deadline LAB 4
(13) 24.Mar	Student: Paper presentations	Operation	
(14) 31.Mar	Metrics/Evaluations	Self-study class   Deadline Paper	
(15) 7.Apr	NO CLASS (pre Easter)		
(16) 14.Apr	Self-study class	REVIEW FOR FINAL EXAM	Deadline LAB 5
(17) 21.Apr	NO CLASS	NO CLASS	Lab Exam
Apr/May??	FINAL EXAM		

Lab exercises are on [github.com](https://github.com)

Getting started (10–15h)

Network programming in Go (10–15h)

**INDIVIDUAL  
TASKS**

**fail/pass**

Failure Detector and Leader Election (20h)

Single-decree Paxos (30h)

Multi-Paxos (40h)

**GROUP  
TASKS**

**0-100**

Oral examinations for labs

**INDIVIDUAL  
TASK**

## Learning through enquiry and discovery

*Please, do not decentralize the Internet with (permissionless) blockchains!*

Pedro Garcia Lopez , Alberto Montresor, Anwitaman Datta

<https://arxiv.org/abs/1904.13093>

Discovery: Become familiar with this paper and literature related to one of the four mentioned problems (minimize scope)

- read
- read actively
- make connections
- organize ideas

### GROUP TASKS

**give a presentation**  
*“show your engagement and engage us”*

Enquiry about the selected problem

- recognize subtle and challenging aspects of decentralized applications in the past
- recognize subtle and challenging aspects of recent decentralized applications

### INDIVIDUAL TASKS

**lightning talk + 4-page writing**  
*“agree or refute”*

## Collaboration

- Talk to each other, TA, instructors or anyone else about any of the assignment. Assistance is limited to general discussion of the problem. Each programming project group must write out their own solutions.
- Consulting another student's/group's solution is prohibited. Submitted solutions must not be copied from any source.
- You shall not supply your work to other students in future instances of this course.

## Deadlines & Slip days

If you cannot make a deadline, e.g. due to illness, resit exams, or other conflicting deadlines, you can use up to a total of **five slip days** throughout the semester for free. Weekends and holidays **are included** in your slip day budget.

Slip days beyond the five free days will count negatively towards your lab grade. For each extra slip day used, your grade is reduced by 5 points.




## Partner problems

Try to avoid having partner problems

1. Hopes
2. Concerns
3. Problems
4. Crises

If you find in a situation which you can't resolve, ask for help: TA, instructor

Act early



Don't know.  
I copied it from stack overflow

Any form of cheating, plagiarism, i.e. copying of another student's text or source code, will result in the grade F, and may be reported to the university for administrative processing.

Committing acts that violate Student Conduct policies that result in course disruption are cause for suspension or dismissal from UiS.

Don't cheat. It's not worth it!

- Oral examination for labs
- Mini-research project
- Final examination

## Overview

- All exercises must be passed.
- For the graded part of the lab, you will be given a grade in the range 0-100 points. Passing is  $> 60$  points
- The lab grade accounts for 30 % of the final grade.
- The mini-research project accounts for 10% of the final grade
- The final exam accounts for the remaining 60 %





Source: Rethinking schooling for the 21st century: the state of education for peace, sustainable development and global citizenship in Asia

## Anonymous feedback

- End of Jan / Beginning of Feb
- Mid Apr

## General information

Serious and critical matters

- You can report serious incidents such as
- Bullying/harassment
- Unwanted sexual attention
- Breach of research ethical guidelines, plagiarism etc.
- Financial misconduct

### Call first:

Fire brigade 110

Police 112

Ambulance 113



### Tell them:

Who you are

What has happened

The place of the incident/address

### Start first aid

### Inform UiS

417 75 566

UiS emergency number

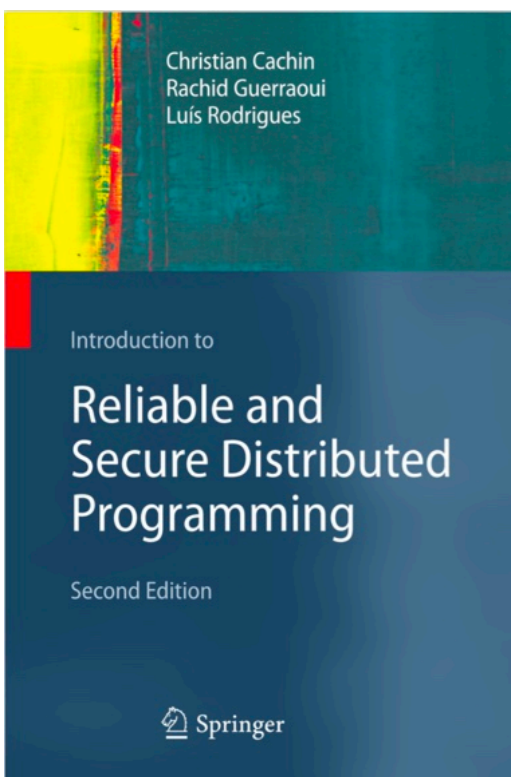
**ONLY FOR UIS**

# Study Resources

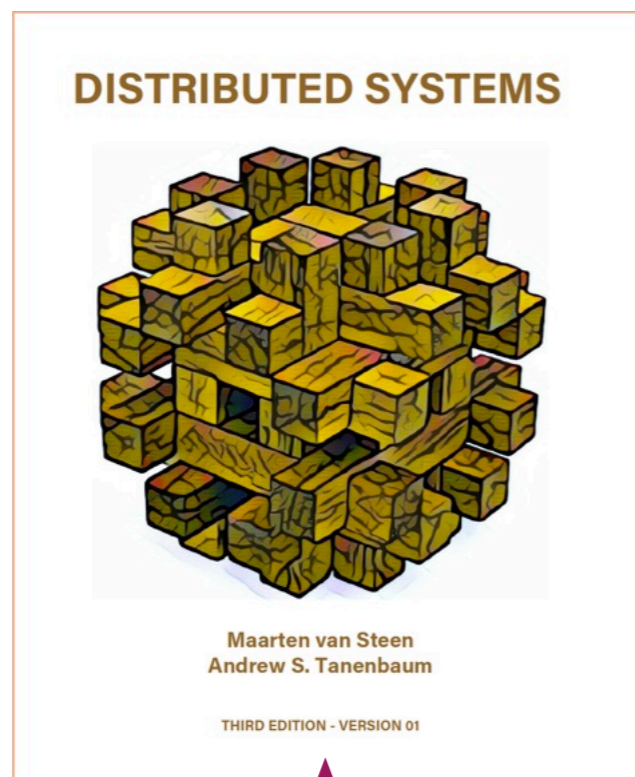
# Course bibliography and more

Topics will be covered during in-class lectures,  
and through collaborative course notes

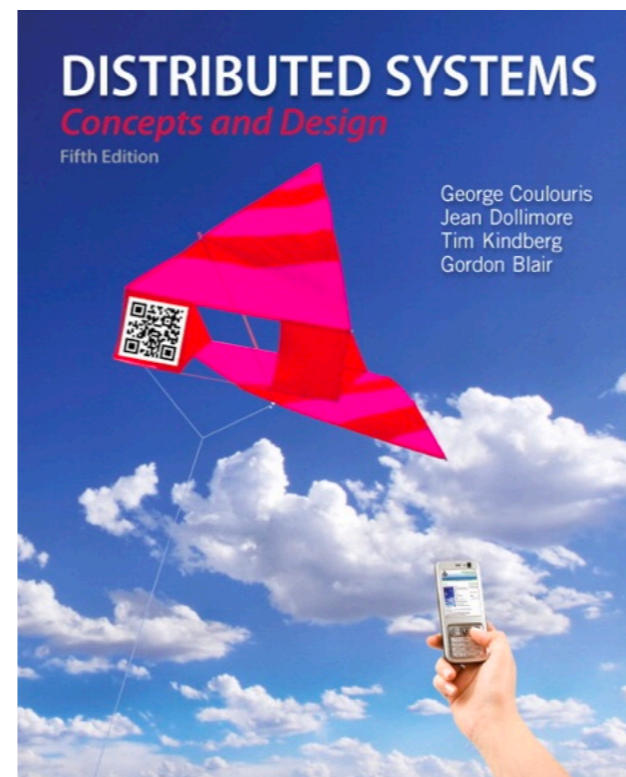
Selection of DS books used to prepare lectures:



Available online  
at UiS library



Available online



Copy available at  
UiS library

Attending these lectures

Attending labs

Doing Labs

Participate in hand-on activities

Reading papers

Skimming papers

Present other's work

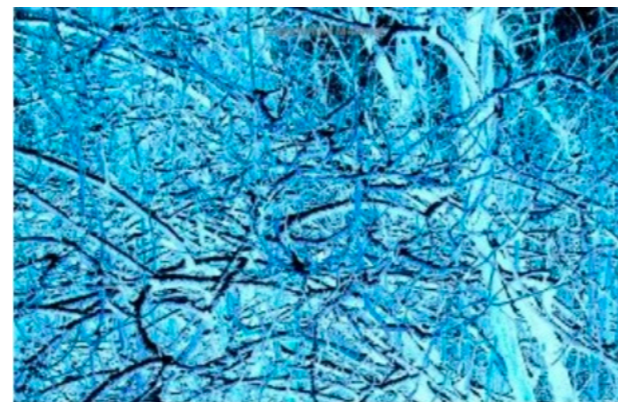
Watch videos conferences, ...)

Write a paper (collaborate)

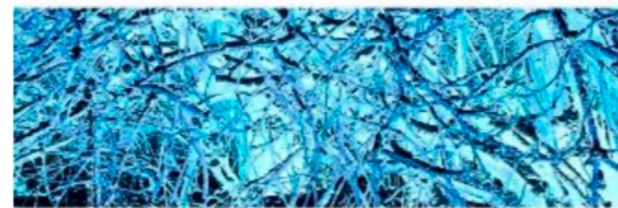


## The extra mile

- Graph theory
- Complexity



Graph Theory and Complex Networks  
An Introduction



Maarten van Steen



Contributing to open source projects

Become a Google SoC student

Participate in Hackatons

Real-world experiments

System prototyping

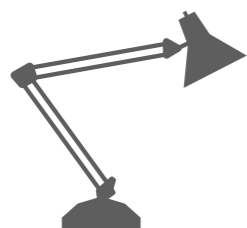
<https://www.distributed-systems.net/index.php/books/gtcn/>



GOOD READ



MUST READ



SELF-STUDY



EXERCISE

# Collaborate

The image shows a screenshot of the Microsoft OneNote application interface. At the top, the title bar displays 'OneNote | Veronica Estrada Galinanes ▶ ClassNotes' and 'ClassNotes'. Below this is the ribbon menu with tabs for 'File', 'Home', 'Insert', 'Draw', 'View', and 'Help'. The 'Home' tab is active, showing a rich text editor toolbar with options for undo, redo, font face (Calibri Light), font size (20), bold (B), italic (I), underline (U), highlight, text color (A), background color, and bullet points. The left sidebar shows a tree view of the notebook 'ClassNotes' with a search icon. The tree view includes the following items: 'Intro', 'Architectures', 'Abstractions', 'Coordination', 'Communication', 'Consistency-Replication', 'FT-Reliability', 'Consensus', 'Naming', 'StateMachineReplicati...', 'Security', 'Operation', and 'Metrics and Evaluation'. The 'Intro' item is selected, and its sub-items are visible: 'General Info', 'Why I should study DS?', and 'Untitled Page'. The main editing area on the right shows a blank page with a timestamp: 'Monday, December 9, 2019 4:33 PM'.

## The three-pass approach

### FIRST-PASS: BIRD'S EYE VIEW



5-10 minutes

- title, abstract, introduction
- section & subsection headings
- mathematical content
- conclusions
- references



<http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/07/paper-reading.pdf>

### How to Read a Paper

Version of February 17, 2016

S. Keshav  
David R. Cheriton School of Computer Science, University of Waterloo  
Waterloo, ON, Canada  
keshav@uwaterloo.ca

#### ABSTRACT

Researchers spend a great deal of time reading research papers. However, this skill is rarely taught, leading to much wasted effort. This article outlines a practical and efficient *three-pass method* for reading research papers. I also describe how to use this method to do a literature survey.

#### 1. INTRODUCTION

Researchers must read papers for several reasons: to review them for a conference or a class, to keep current in their field, or for a literature survey of a new field. A typical researcher will likely spend hundreds of hours every year reading papers.

Learning to efficiently read a paper is a critical but rarely taught skill. Beginning graduate students, therefore, must learn on their own using trial and error. Students waste much effort in the process and are frequently driven to frustration.

For many years I have used a simple 'three-pass' approach to prevent me from drowning in the details of a paper before getting a bird's-eye-view. It allows me to estimate the amount of time required to review a set of papers. Moreover, I can adjust the depth of paper evaluation depending on my needs and how much time I have. This paper describes the approach and its use in doing a literature survey.

#### 2. THE THREE-PASS APPROACH

The key idea is that you should read the paper in up to three passes, instead of starting at the beginning and plowing your way to the end. Each pass accomplishes specific goals and builds upon the previous pass: The *first* pass gives you a general idea about the paper. The *second* pass lets you grasp the paper's content, but not its details. The *third* pass helps you understand the paper in depth.

##### 2.1 The first pass

The first pass is a quick scan to get a bird's-eye view of the paper. You can also decide whether you need to do any more passes. This pass should take about five to ten minutes and consists of the following steps:

1. Carefully read the title, abstract, and introduction
2. Read the section and sub-section headings, but ignore everything else
3. Glance at the mathematical content (if any) to determine the underlying theoretical foundations
4. Read the conclusions
5. Glance over the references, mentally ticking off the ones you've already read

At the end of the first pass, you should be able to answer the *five Cs*:

1. *Category*: What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?
2. *Context*: Which other papers is it related to? Which theoretical bases were used to analyze the problem?
3. *Correctness*: Do the assumptions appear to be valid?
4. *Contributions*: What are the paper's main contributions?
5. *Clarity*: Is the paper well written?

Using this information, you may choose not to read further (and not print it out, thus saving trees). This could be because the paper doesn't interest you, or you don't know enough about the area to understand the paper, or that the authors make invalid assumptions. The first pass is adequate for papers that aren't in your research area, but may someday prove relevant.

Incidentally, when you write a paper, you can expect most reviewers (and readers) to make only one pass over it. Take care to choose coherent section and sub-section titles and to write concise and comprehensive abstracts. If a reviewer cannot understand the gist after one pass, the paper will likely be rejected; if a reader cannot understand the highlights of the paper after five minutes, the paper will likely never be read. For these reasons, a 'graphical abstract' that summarizes a paper with a single well-chosen figure is an excellent idea and can be increasingly found in scientific journals.

##### 2.2 The second pass

In the second pass, read the paper with greater care, but ignore details such as proofs. It helps to jot down the key points, or to make comments in the margins, as you read. Dominik Grusemann from Uni Augsburg suggests that you "note down terms you didn't understand, or questions you may want to ask the author." If you are acting as a paper referee, these comments will help you when you are writing your review, and to back up your review during the program committee meeting.

## The three-pass approach

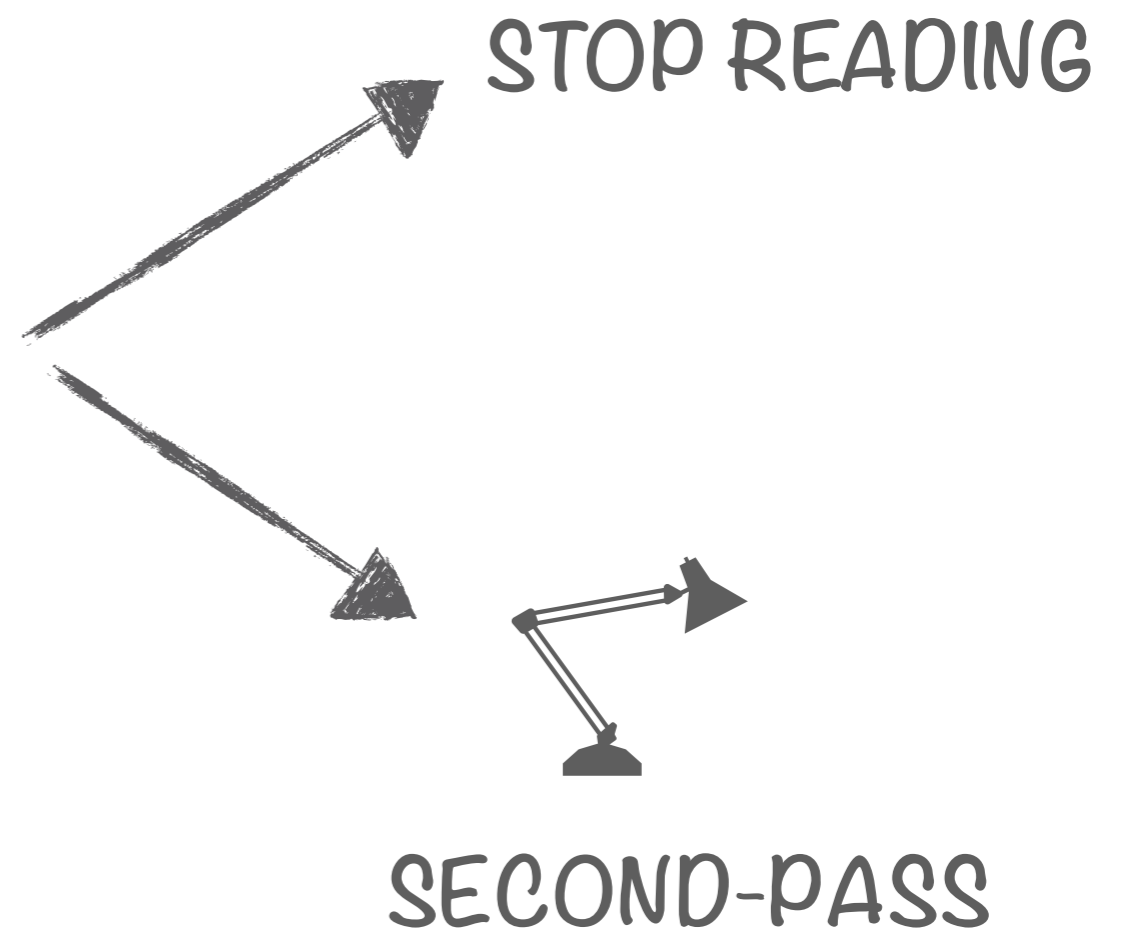
**Category:** What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?

**Context:** Which other papers is it related to? Which theoretical bases were used to analyze the problem?

**Correctness:** Do the assumptions appear to be valid?

**Contributions:** What are the paper's main contributions?

**Clarity:** Is the paper well written?



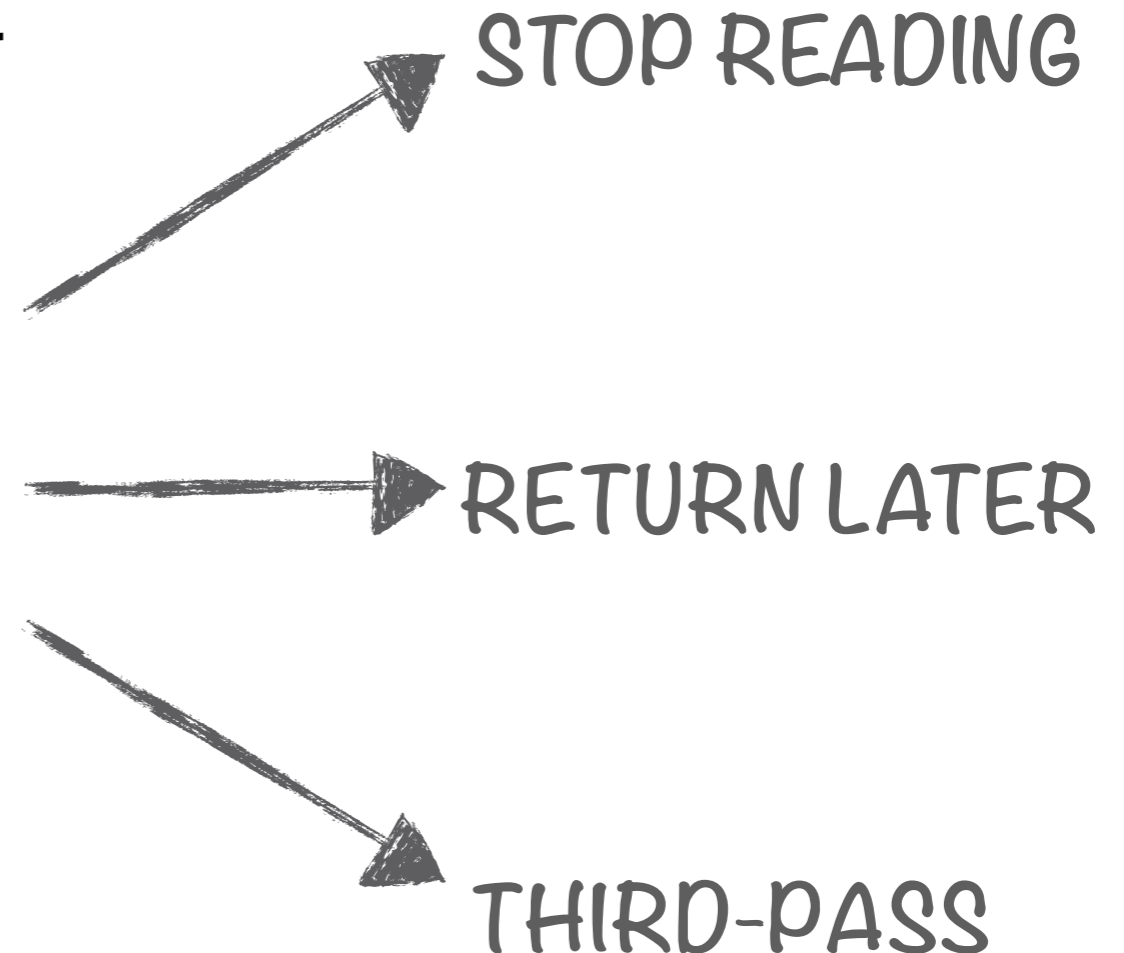
## The three-pass approach

### SECOND-PASS TO GRASP THE CONTENT



up to 1 hour for expert reader

- Read paper carefully avoid proofs, annex,...
- Look carefully at the figures, diagrams and other illustrations in the paper. Pay special attention to graphs.
- Remember to mark relevant unread references for further reading (this is a good way to learn more about the background of the paper).



## The three-pass approach

### THIRD-PASS TO UNDERSTAND THE PAPER



"many hours" for the beginner

- Virtually re-create the paper
- Identify and challenge every assumption
- How would you present it?

RECONSTRUCT THE PAPER BY MEMORY

IDENTIFY STRONG AND WEAK POINTS

PINPOINT ANY ISSUE

# Reference Management Programs

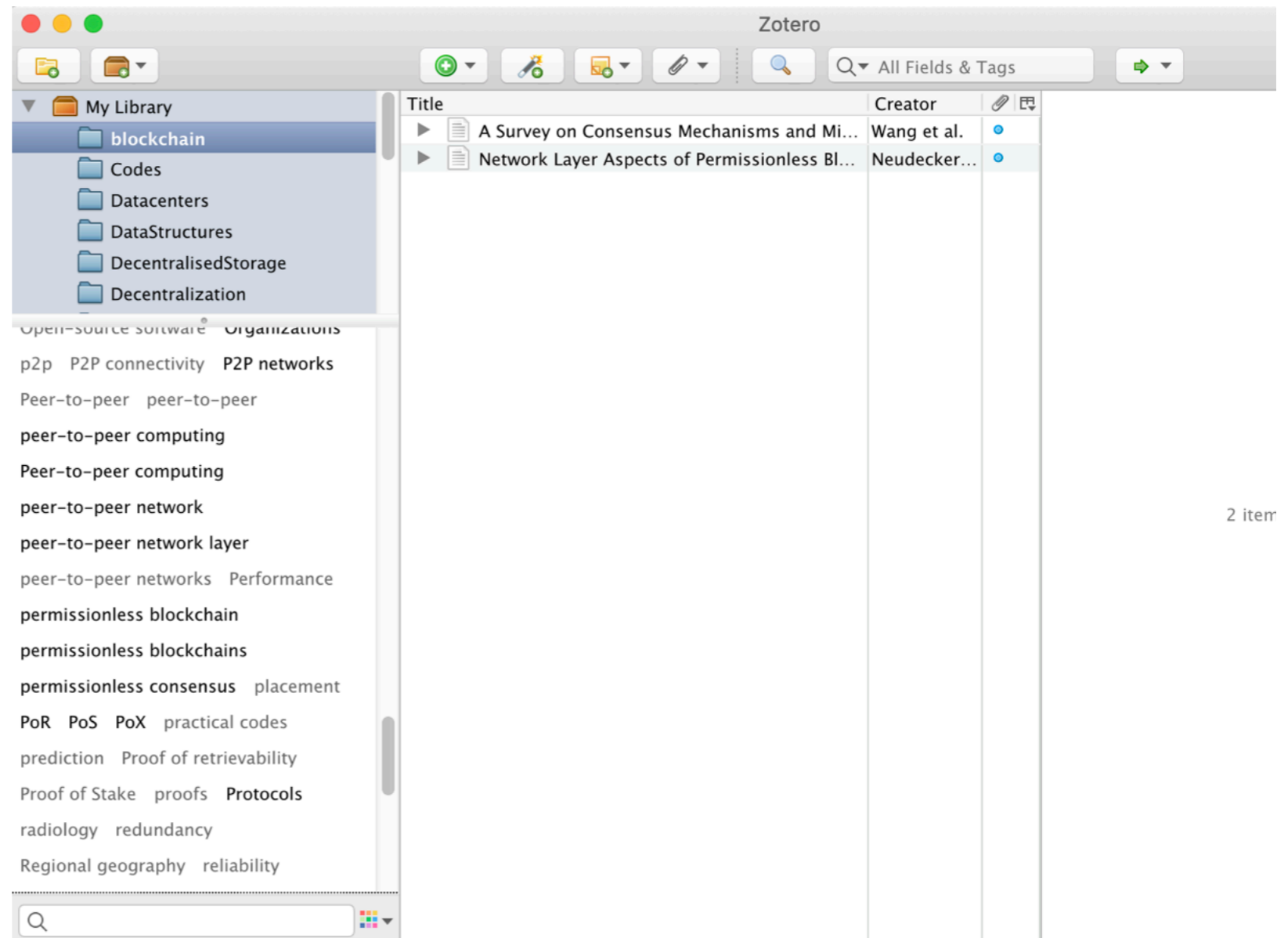
## zotero

Zotero is a program for managing references. Zotero helps you collect, organize, cite and share research.

<https://student.uis.no/library/classes/master/zotero/>

## BIBTEX

The word „[BibTeX](#)“ stands for a tool and a file format which are used to describe and process lists of references, mostly in conjunction with LaTeX documents.





## Learning through enquiry and discovery

*Please, do not decentralize the Internet with (permissionless) blockchains!*

Pedro Garcia Lopez , Alberto Montresor, Anwitaman Datta

For next Tuesday (**homework**):

- read the paper (two-pass approach)
- form a group of 2 students
- install a reference management program
- choose one of the four problems mentioned in the paper (first option & second option)

# Questions?

# Distributed Systems

**DISTRIBUTED COMPUTING**    **DECENTRALIZED COMPUTING**  
**CENTRALIZED COMPUTING**    **AUTONOMOUS COMPUTING**  
**PARALLEL COMPUTING**  
**VOLUNTEER COMPUTING**  
**CLUSTER COMPUTING**    **TRUSTWORTHY COMPUTING**  
**GRID COMPUTING**  
**SERVICE-ORIENTED COMPUTING**  
**UTILITY COMPUTING**  
**PERVASIVE COMPUTING**    **CLOUD COMPUTING**  
**EDGE COMPUTING**  
**UBIQUITOUS COMPUTING**    **FOG COMPUTING**  
**SERVERLESS COMPUTING**

# What is this?



**HINT: It's not Mad Hatter's hat**



What is this?



# Cray-1 supercomputer (1975)

160 MFLOPS

# What is this?



**Interlock puzzle**

**Solid burr**  
**Holey burr**

**HOMEWORK: Find out the reason  
why they appear in the book**



# Interlock puzzle

Bill Cutler's wrote puzzle programs during various decades

1965...first program that never run in a computer

1986 he bought 8MHZ IBM PC AT 80286 chip - a mainframe was 40 times faster

**6-piece burr**

**35.5 billion possibilities**

**original estimate 400 years!**



Cray computers at Lawrence Livermore Labs (when idle)

The introduction of 80386 chip and friend's computers

**400 → 62.5 > 2.5 years**

## How do we increase speed?

4 GHz processor performs 4,000,000,000 clock cycles per second

### High Performance Computing (HPC)

#### Parallel Computing

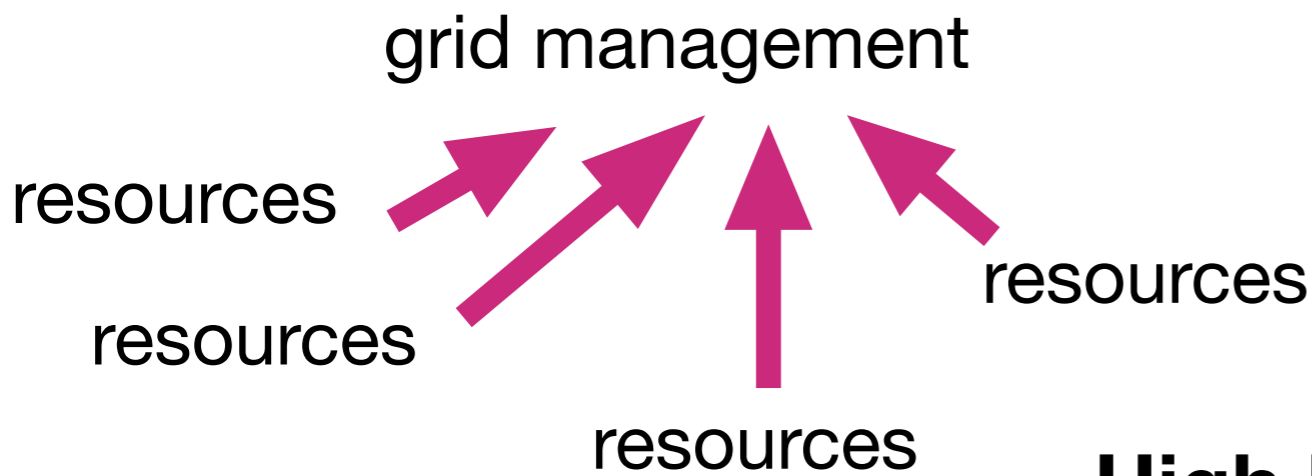
70s

Massively parallel processors are made by chaining together hundreds or thousands of inexpensive commercial microprocessors.



**Electric Power Grid ~ Grid / Utility computing**

**How do we reduce the cost of computing, increase reliability, and increase flexibility?**



- Open Science Grid (US national-funded grid)
- European Grid Infrastructure

## High Performance Computing (HPC)

### Grid Computing

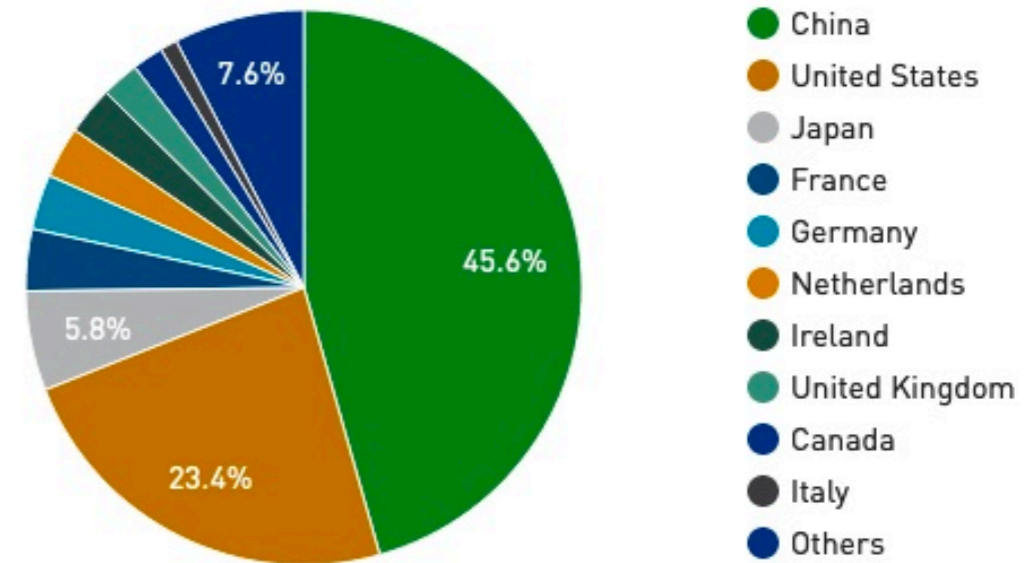
90s

The grid combines various computing resources, databases, and measuring devices that comprise a pool of resources for coordinated, integrated and flexible shared use.

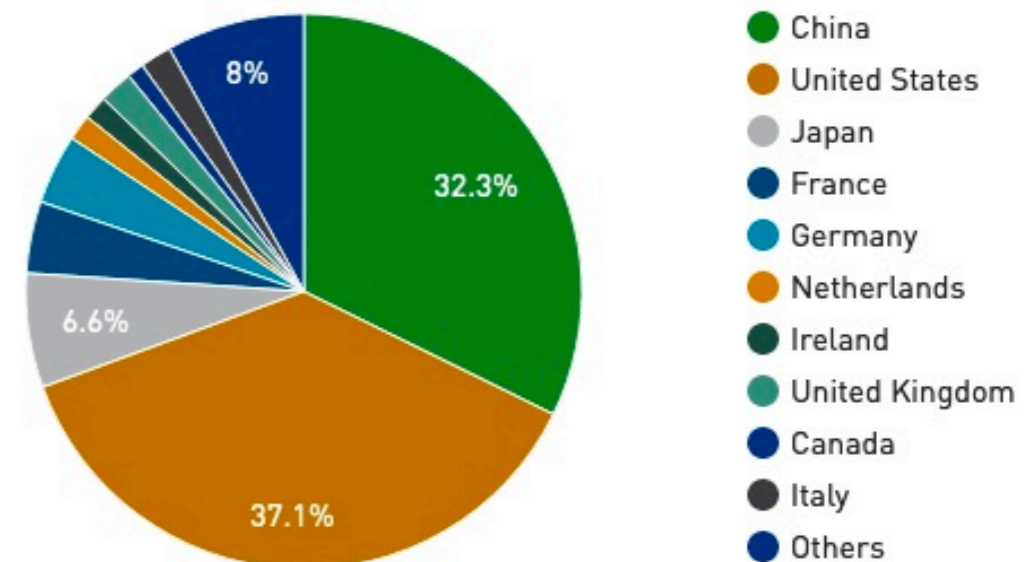
## Top 500 Supercomputers

- Since June 2019 only Petaflop systems have been able to make the list.
- The total aggregate performance of all 500 system has now risen to **1.65 Exaflops**.

Countries System Share



Countries Performance Share



<https://www.top500.org/>

## How do we scale?

- Infrastructure as a service (IaaS)      **Amazon EC2 / S3**
- Platform as a service (PaaS)      **Google App Engine**
- Software as a service (SaaS)      **Salesforce**

## BIG DATA

## Cloud Computing

A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.



Foster, Ian, et al. "Cloud computing and grid computing 360-degree compared." *arXiv preprint arXiv:0901.0131* (2008).

# SETI HOME

SETI@home is a scientific experiment, based at UC Berkeley, that uses Internet-connected computers in the Search for Extraterrestrial Intelligence (SETI).

You can participate by running a free program that downloads and analyzes radio telescope data.

# Paradigms in Computing

10 or 100 of thousands  
*unreliable* and  
*heterogeneous*  
machines

Example: SETI@home using the  
BOINC infrastructure (UC Berkeley)

## Volunteer Computing

Internet-connected computers,  
volunteered by their owners, as a source  
of computing power and storage.

Anderson, David P. "Boinc: A system for public-resource computing and storage." *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*. IEEE Computer Society, 2004.

Mengistu, TESSEMA M., and Dunren Che. "Survey and taxonomy of volunteer computing." *ACM J. Comput. Surv* (2019): 1-35.





# Distributed communication networks

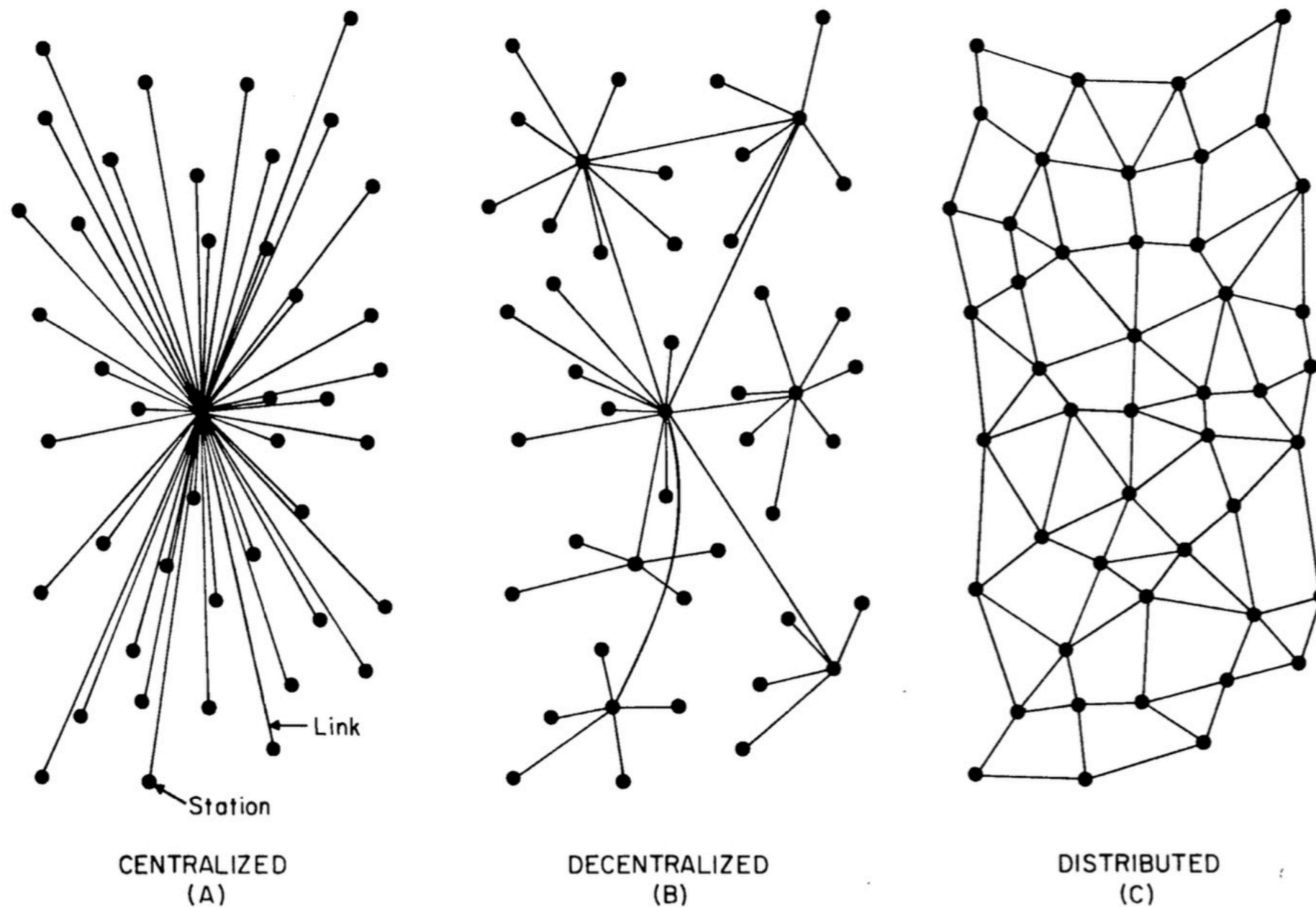


FIG. 1 — Centralized, Decentralized and Distributed Networks



Baran, Paul. "On distributed communications networks." *IEEE transactions on Communications Systems* 12.1 (1964): 1-9.

**Computer Networks:** the autonomous computers are explicitly visible and must be specifically addressed

**Distributed systems:** the multiple autonomous computers and components are transparent to the user

**Parallel computers:** single system view without physical separation

Many problems are however in common:

- Scheduling
- Load balancing
- Resource and data sharing/distribution

Networks are in some sense also distributed systems (e.g. name services) and every distributed system relies on services provided by a network

Distributed systems may serve the same purpose as parallel computers: high performance

# Distributed Systems - Definitions

“A system in which **hardware or software components** located at networked computers communicate and coordinate their actions only by message passing.” [Coulouris]

# Distributed Systems - Definitions

“A system that consists of a collection of two or more independent computers which **coordinate their processing** through the exchange of **synchronous or asynchronous** message passing.”

# Distributed Systems - Definitions

“A distributed system is a collection of independent computers that **appear to the users of the system as a single computer.**” [Tanenbaum]

# Distributed Systems - Definitions

“A distributed system is a collection of autonomous computers **linked by a network** with software designed to produce an integrated computing facility.”

# Distributed Systems - Definitions

“A distributed system is one in which **the failure of a machine you have never heard of can cause your own machine to become unusable**”. [Lamport]

# Distributed Systems - Definitions

**They have in common: Distributed hardware and/or distributed data and/or distributed control and computers connected through some network**



# Distributed Systems - Definitions

“A collection of autonomous computing elements that appears to its users as a **single coherent system.**” [M. van Steen/Tanembaum]

# DS - Application Domains

- Finance and commerce
- The information society
- Creative industries and entertainment
- Healthcare
- Education
- Transport and logistics
- Science
- Environmental management

# Distributed Systems - Examples

## Web Search

Search engines need to index the entire contents of the World Wide Web.

Google's index contains **hundreds of billions of webpages**.

That makes an index of more than **100,000,000 gigabytes**

- physical infrastructure of networked computers
- distributed file system that supports very large files
- distributed storage system with very large datasets
- distributed locking and agreement
- a programming model for very large parallel and distributed computation

Google

YAHOO!

Yandex

Aol.

Ask.com

Baidu 百度

Bing



DuckDuckGo

Search engine focused on online privacy

~900 millions/month processed requests

# Distributed Systems - Examples

## A modern telescope: The Pierre Auger Project, Colorado, USA

- The radio telescope itself may be a wireless distributed system developed as a grid of a few thousand sensor nodes
- The central point needs to be a reasonably powerful system, capable of storing and processing the events sent to it by the sensor nodes.
- Most partners have local distributed systems (often in the form of a cluster of computers) that they use to further process the data collected by the telescope.



<https://www.auger.org/>

# Distributed Systems - Examples

## 150 years of transatlantic communication advancements enabled the first telesurgery

- 1792: telecommunication network in France based on Chappe telegraph
- 1851: submarine telegraph link between France (Calais) and Britain (Dover)
- 1858: transatlantic telegraph cable - theoretical speed of 2.5 words/s 🚢 ✉️
- 1875: duplex technology application (80 words/s)
- 1928: radio mobile links (100 meters)
- 1956: intercontinental submarine telephone cable (TAT-1)
- 1962: intercontinental 🌐 links bring 📺
- 1998: fiberoptic transatlantic cable, TAT-8 is build, with a capacity of 40,000 simultaneous 📞
- 2000: transatlantic self-healing ring network with seven segments linking Europe and the United States. TAT-14 offers 64x more capacity. Some 80 % of this capacity will be dedicated to Internet and multimedia traffic, enabling routing of the equivalent of eight million simultaneous calls



“OPERATION LINDBERGH” A World First in TeleSurgery: The Surgical Act Crosses the Atlantic! New York – Strasbourg - 2001



Charles Lindbergh, New York to Paris in 33-hour flight - 1927

# Distributed Systems: Why?

A distributed system should

- make resources easily accessible;
- hide the fact that resources are distributed across a network;
- be open;
- be scalable.

## Make resources easily accessible

Typical examples include peripherals, storage facilities, data, files, services, and networks, to name just a few.

There are many reasons for wanting to share resources. One obvious reason is that of economics.

## Hide the fact that resources are distributed across a network

Transparency	Description
--------------	-------------

Access	Hide differences in data representation and how an object is accessed
--------	---

Location	Hide where an object is located    ← <b>Naming</b>
----------	--

Relocation	Hide that an object may be moved to another location while in use
------------	---

Migration	Hide that an object may move to another location
-----------	--

Replication	Hide that an object is replicated
-------------	-----------------------------------

Concurrency	Hide that an object may be shared by several independent users
-------------	--

Failure	Hide the failure and recovery of an object
---------	--



## Be open

### Interoperability, composability, and extensibility

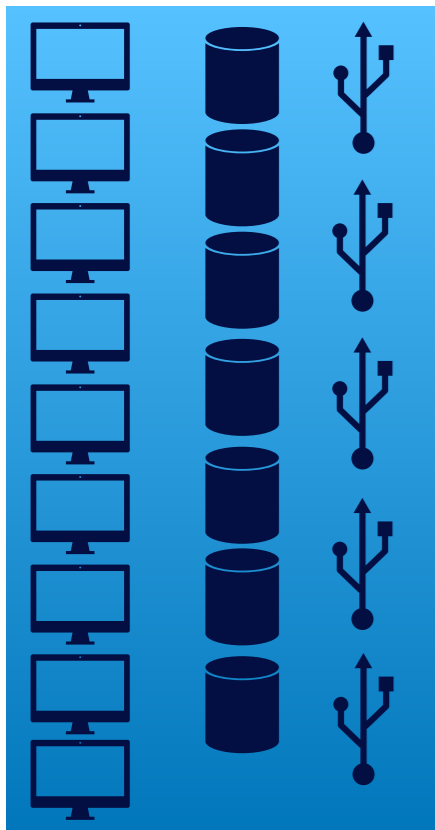
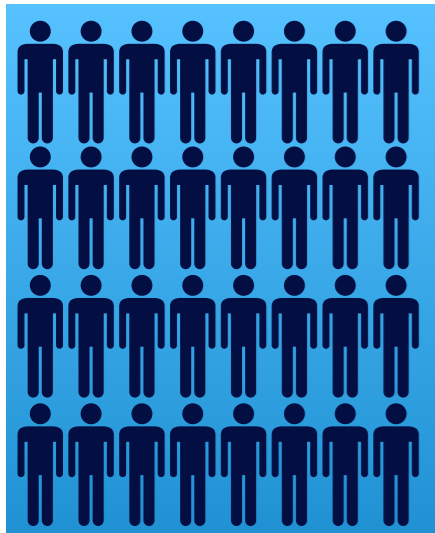
An open distributed system is essentially a system that offers components that can easily be used by, or integrated into other systems.

At the same time, an open distributed system itself will often consist of components that originate from elsewhere.

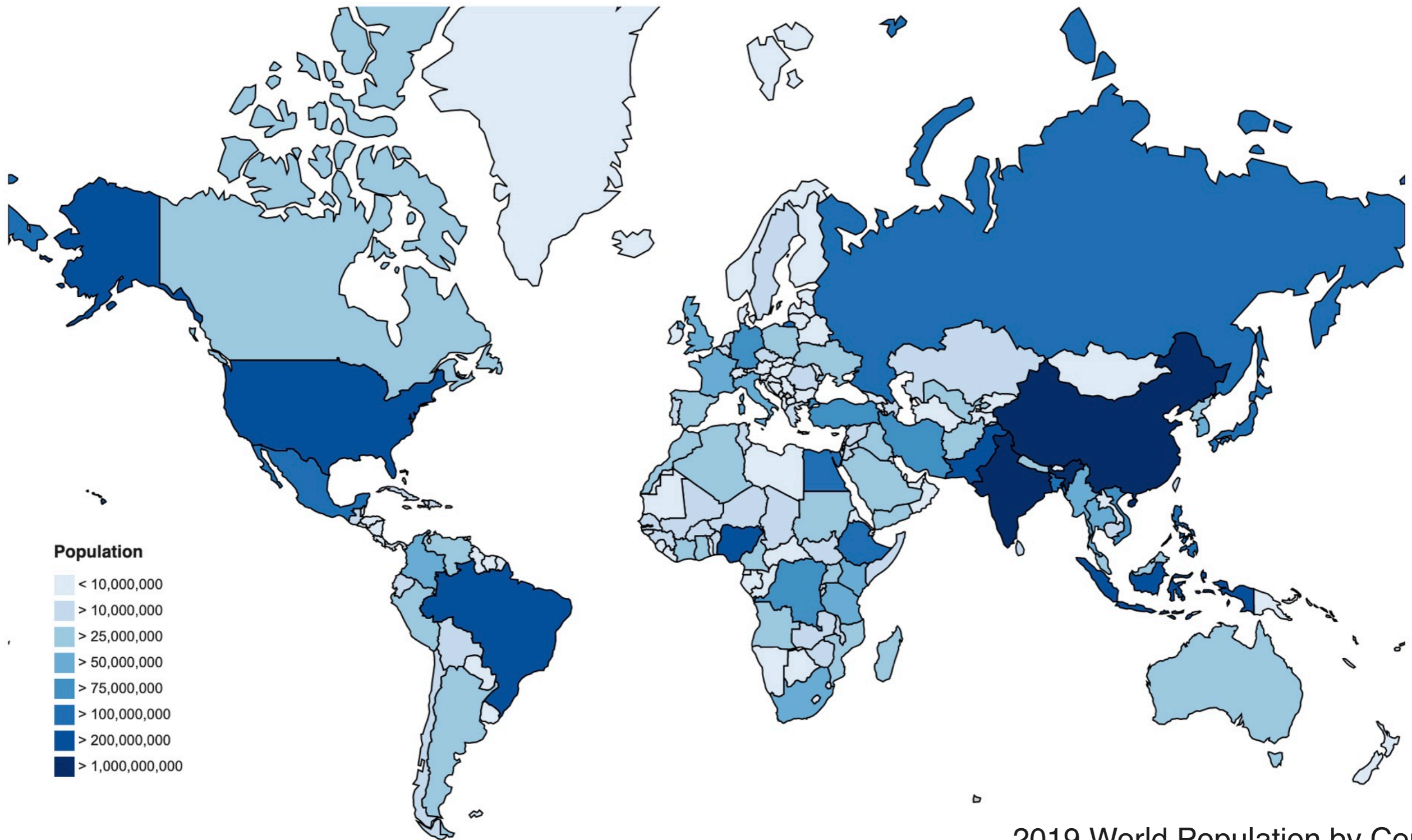
## Be scalable

It can be measured along at least 3 dimensions  
[Neuman, 1994]:

- Size scalability
- Geographical scalability
- Administrative scalability



# DS: Why? - Scalability

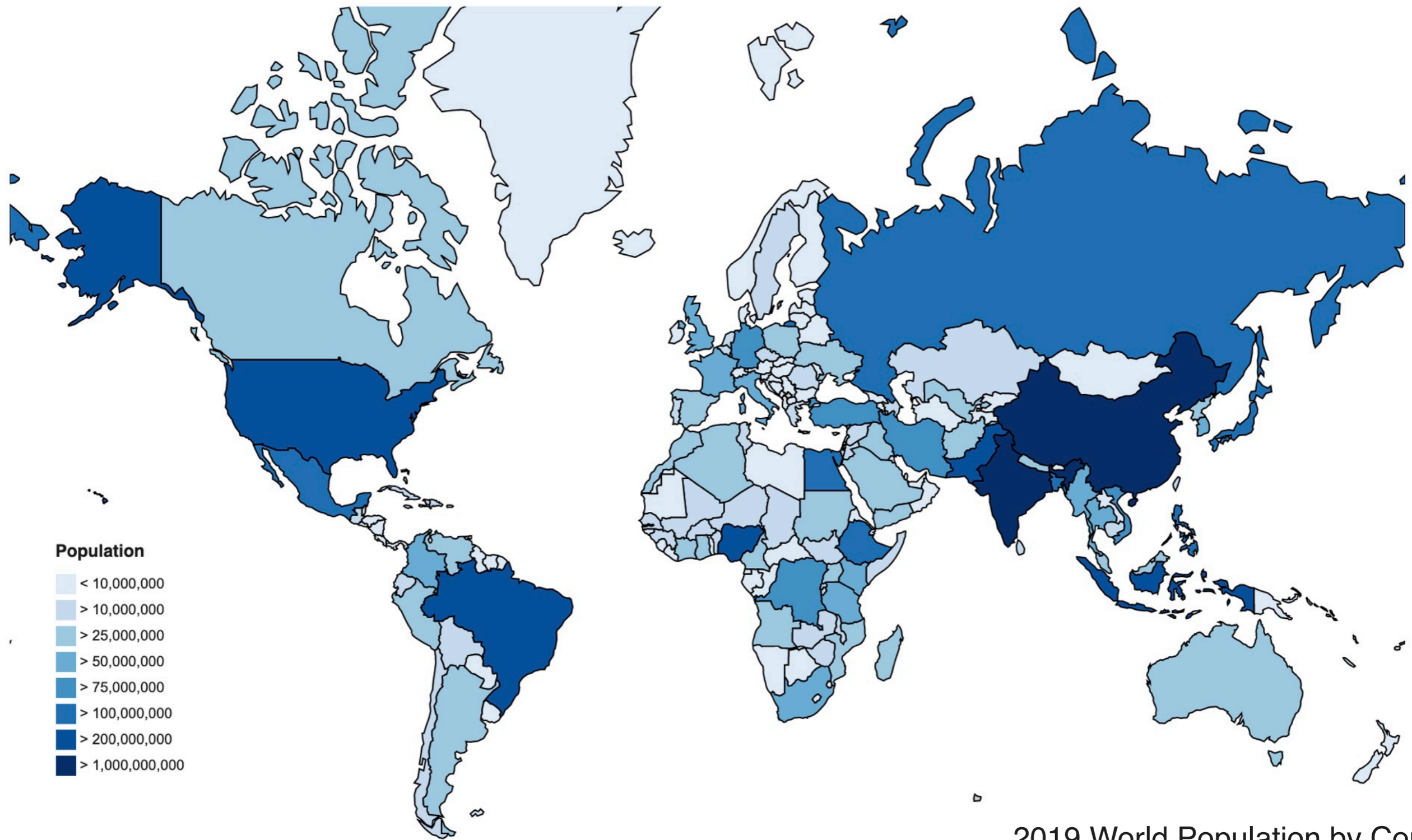


2019 World Population by Country

Live Population: 7,755,348,728

<http://worldpopulationreview.com/>

# DS: Why? - Scalability



## Population

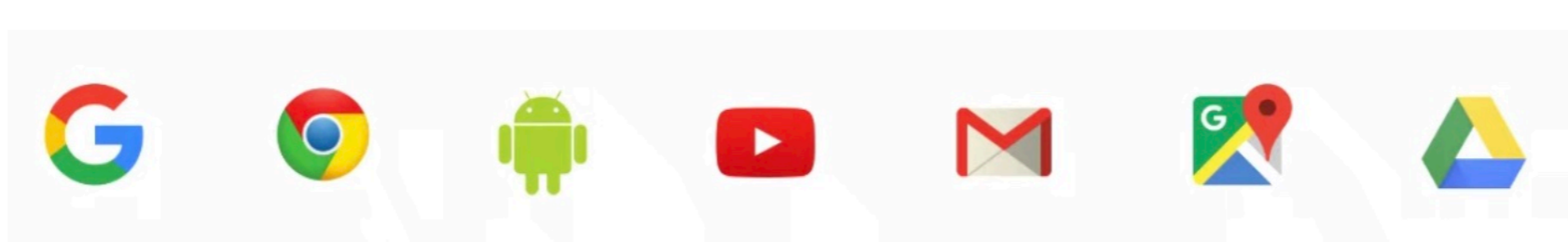
- < 10,000,000
- > 10,000,000
- > 25,000,000
- > 50,000,000
- > 75,000,000
- > 100,000,000
- > 200,000,000
- > 1,000,000,000

Live Population: 7,755,348,728

2019 World Population by Country

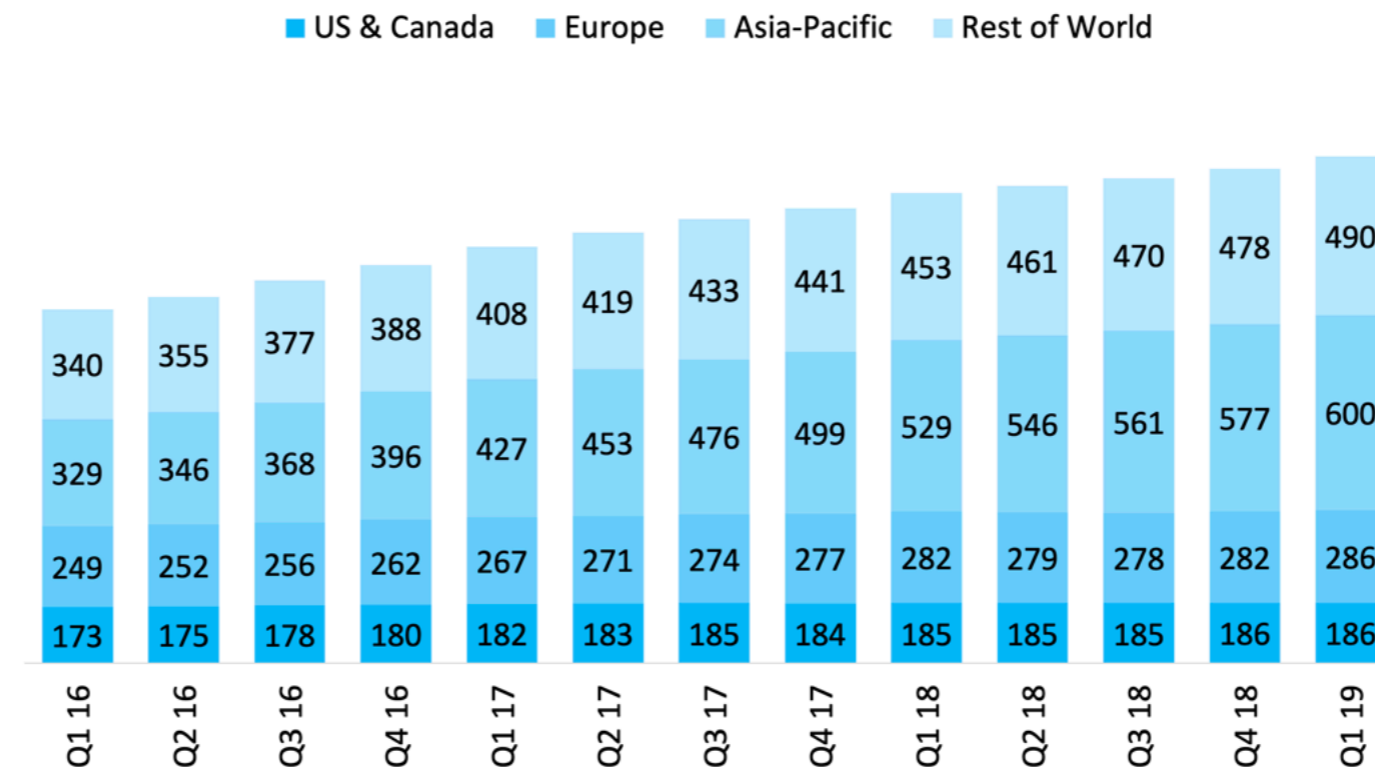
<http://worldpopulationreview.com/>

## Google: services with more than 1 billion users



## Facebook:

**f** Facebook Daily Active Users By Geography  
*In millions*



Source: Company filings

BUSINESS  
INSIDER  
INTELLIGENCE

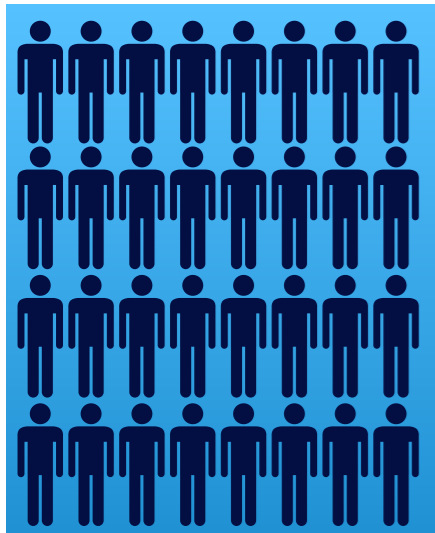
## Case: Spotify (Sweden)

### SUMMARY USER AND FINANCIAL METRICS

USERS (M)	Q3 2018	Q2 2019	Q3 2019	% Change	
				Y/Y	Q/Q
Total Monthly Active Users ("MAUs")	191	232	248	30%	7%
Premium Subscribers	87	108	113	31%	5%
Ad-Supported MAUs	109	129	141	29%	9%
<b>FINANCIALS (€M)</b>					
Premium	1,210	1,502	1,561	29%	4%
Ad-Supported	142	165	170	20%	3%
<b>Total Revenue</b>	<b>1,352</b>	<b>1,667</b>	<b>1,731</b>	<b>28%</b>	<b>4%</b>
Gross Profit	342	434	441	29%	1%
<b>Gross Margin</b>	<b>25.3%</b>	<b>26.0%</b>	<b>25.5%</b>	--	--
<b>Operating (Loss)/Income</b>	<b>(6)</b>	<b>(3)</b>	<b>54</b>	--	--
Operating Margin	(0.5%)	(0.2%)	3.1%	--	--
Net cash flows from operating activities	80	90	71	(11%)	(21%)
<b>Free Cash Flow<sup>1</sup></b>	<b>33</b>	<b>50</b>	<b>48</b>	<b>45%</b>	<b>(4%)</b>

## Twitter Bots

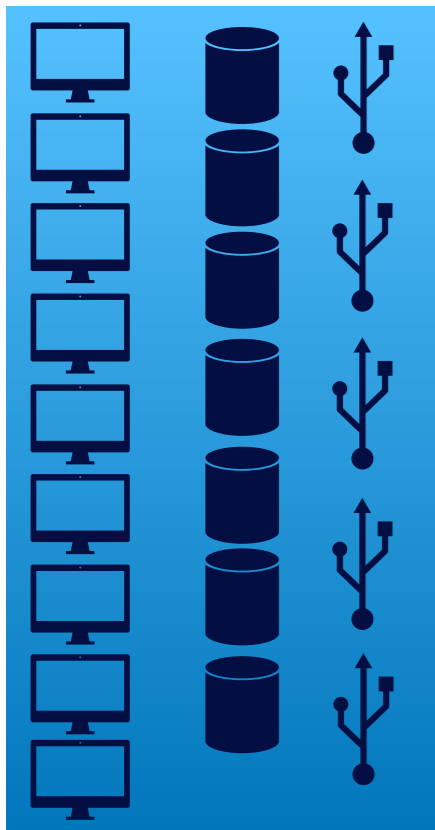
- Netflix Bot (@netflix\_bot) automatically tweets when new content has been added to the online streaming service.
- Grammar Police (@\_grammar\_) is a bot that identifies grammatically incorrect tweets and offers suggestions for correct usage
- Museum Bot (@museumbot) posts random images from the Metropolitan Museum of Art
- The CNN Breaking News Bot (@attention\_cnn) is an unofficial account that sends an alert whenever CNN claims to have breaking news
- The New York Times 4th Down Bot (@NYT4thDownBot) is a bot that provides live NFL analysis.
- PowerPost by the Washington Post (@PowerPost) is a bot that provides news about decision-makers in Washington.



## Be scalable

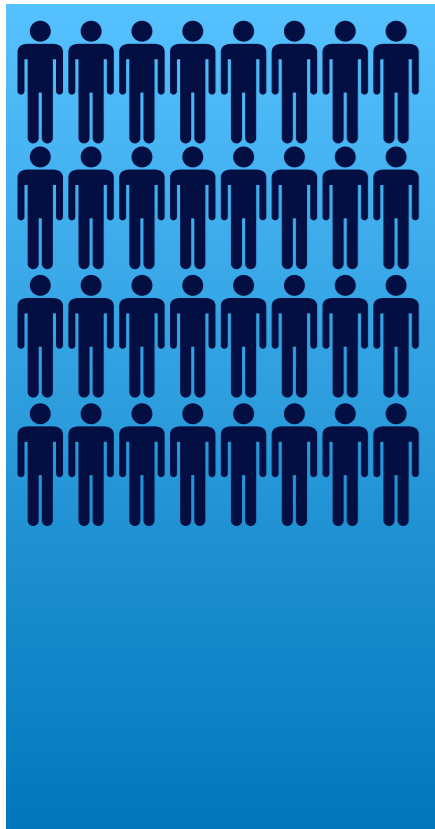
### Size scalability:

- more users
- more resources



Potential bottlenecks: computational, storage and network capacity

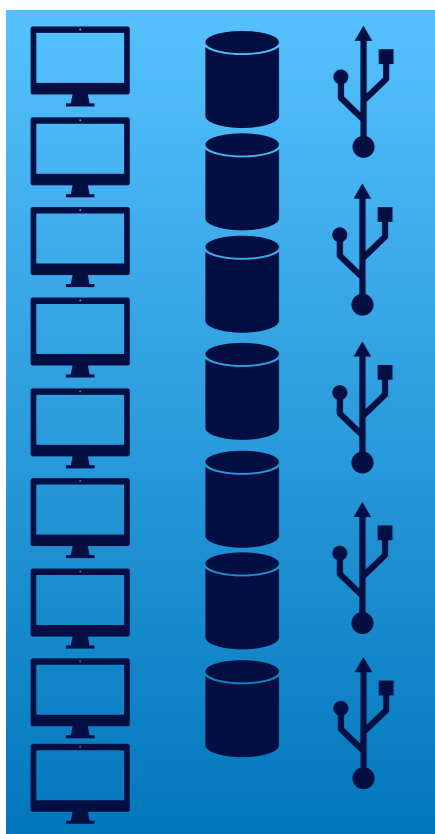




## Be scalable

### **Geographical scalability:**

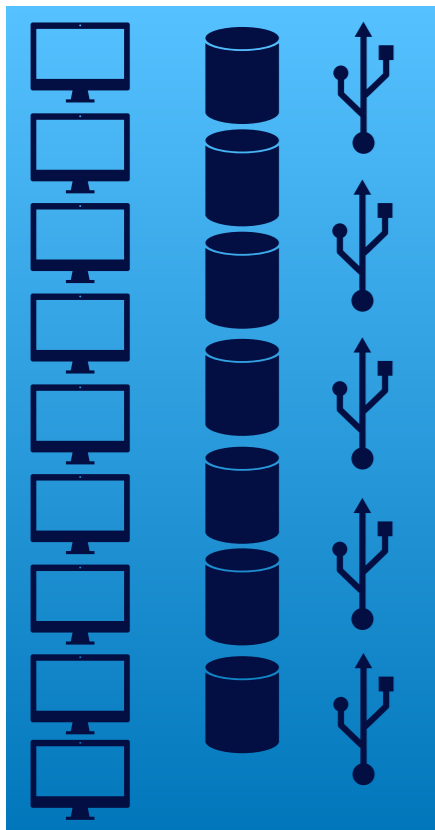
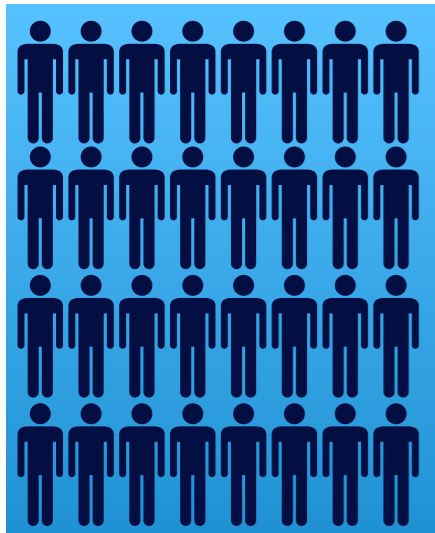
users and resources may lie far apart, but the fact that communication delays may be significant is hardly noticed.



Portability to wide area networks: synchronous communications, reliability, bandwidth, multipoint communication

## Be scalable

**Administrative scalability:** An administratively scalable system is one that can still be easily managed even if it spans many independent administrative organizations.



Performance problems due to limited capacity  
(server & networks)

**Scaling up (vertical)** improve capacity

**Scaling out (horizontal)** adding more machines

Hiding communication latencies

- geographical scalability
- asynchronous communication

Partition and distribution

Replication

- availability
- load balance
- latency

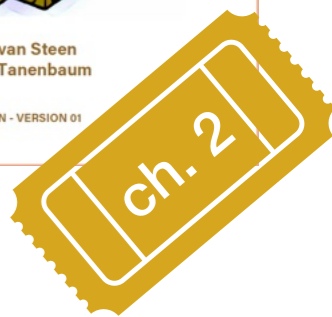
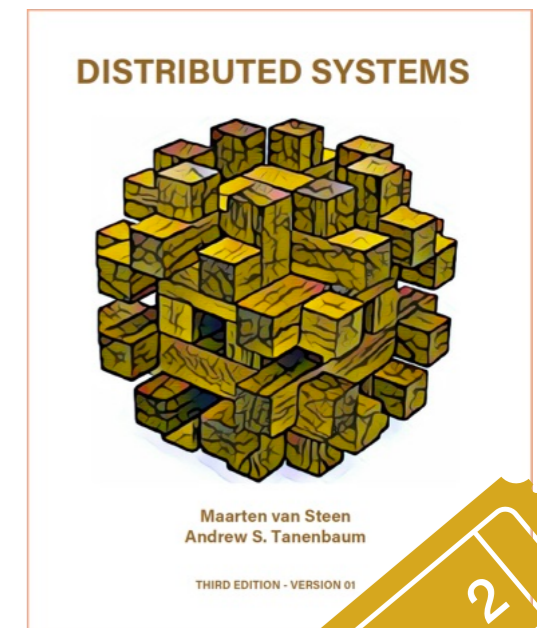
# DS system architectures

Centralized architecture: Client-server

Decentralized architecture: Peer-to-peer (p2p)

- structured p2p
- unstructured p2p
- hierarchical p2p

Hybrid architectures



**TO BE  
CONTINUED...**

## Correctness

- **Safety** stipulates that some bad thing does not happen during execution
- **Liveness** stipulates that something good eventually happens during execution
  - ▶ the program's ability to make progress
  - ▶ termination is not necessarily a requirement for liveness

See Alpern, Bowen, and Fred B. Schneider. "Defining liveness." *Information processing letters* 21.4 (1985): 181-185.

## Safety property

Property	The bad thing
<b>Mutual exclusion</b>	two processes executing in critical sections simultaneously
<b>Deadlock freedom</b>	deadlock
<b>Livelock freedom</b>	livelock
<b>Partial correctness</b>	terminating in a state that does not satisfy the postcondition after having been started in a state that satisfy the precondition
<b>First come first serve</b>	servicing a request that was made after one not yet serviced

## Liveness property

**Property**

**The good thing**

**Starvation freedom**

making progress: a process make progress infinitely often

**Termination freedom**

completion of the final instruction: a program does not run forever

**Guaranteed service**

receiving a service: every request for service is satisfied eventually



# Syllabus - Main topics

# January

week 3

week 3

week 4

week 5

week 6



ABSTRACTIONS



COMMUNICATION



COORDINATION



AVAILABILITY,  
CONSISTENCY & ACID



FAULT-TOLERANCE  
AND RELIABILITY

# February

week 6-9

week 9



CONSENSUS



NAMING



SECURITY



DESIGN &  
OPERATION



EVALUATION

# March

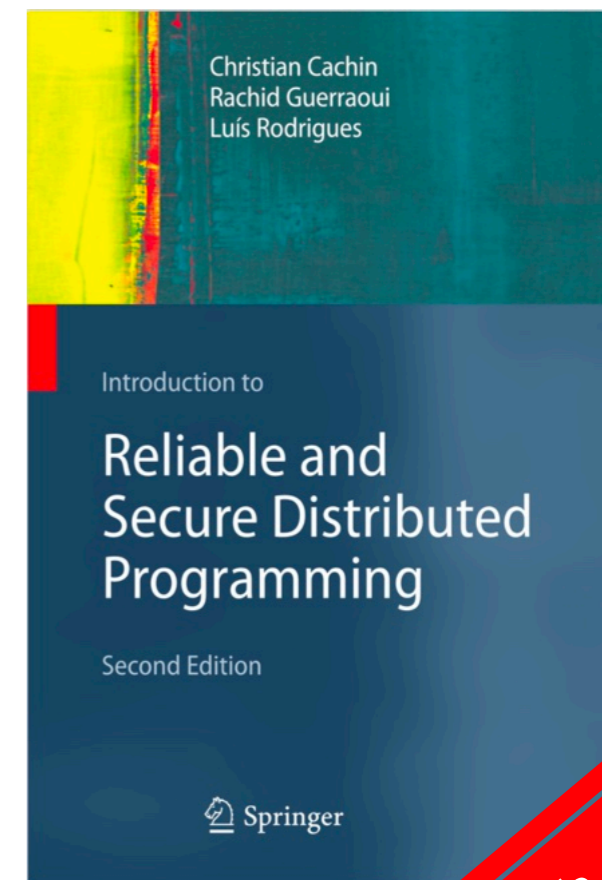
# Overview

Abstractions



# 1 - Abstractions

Distributed programming can be significantly simplified if the difficulty of robust cooperation is encapsulated within specific *abstractions*.



“A distributed system is one in which the failure of a computer you did not even know existed can render your own computer unusable.” Lamport

## Failure models

**Fail-stop failures:** In response to a failure, the component changes to a state that permits other components to detect that a failure has occurred and then stops [Schneider 1984].

**Byzantine failures:** The component can exhibit arbitrary and malicious behavior, perhaps involving collusion with other faulty components [Lamport et al. 1982].

failure assumptions, the environment, the system parameters, and other design choices affect the design of your solution

[Schneider 1984] Byzantine generals in action: Implementing fail-stop processors. ACM TOCS 2,2 (May), 145-154.

[Lamport et al. 1982] The Byzantine generals problem. ACM TOPLAS 4, 3 (July), 382-401.

## A myriad of possible algorithms

**Fail-stop:** processes can fail by crashing but the crashes can be reliably detected by all the other processes;

**Fail-silent:** process crashes can never be reliably detected;

**Fail-noisy:** processes can fail by crashing and the crashes can be detected, but not always in an accurate manner (accuracy is only eventual);

**Fail-recovery:** processes can crash and later recover and still participate in the algorithm;

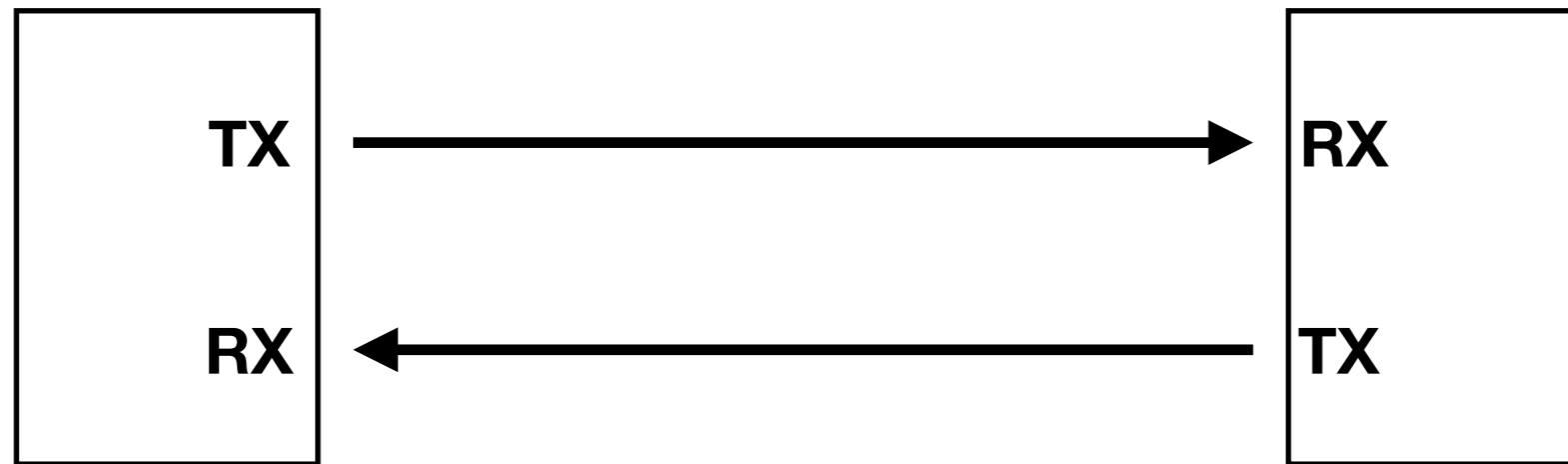
**Fail-arbitrary:** processes can deviate arbitrarily from the protocol specification and act in malicious, adversarial ways; and

**Randomized:** processes may make probabilistic choices by using a source of randomness.

Communication

2

## Synchronous communication (full duplex)

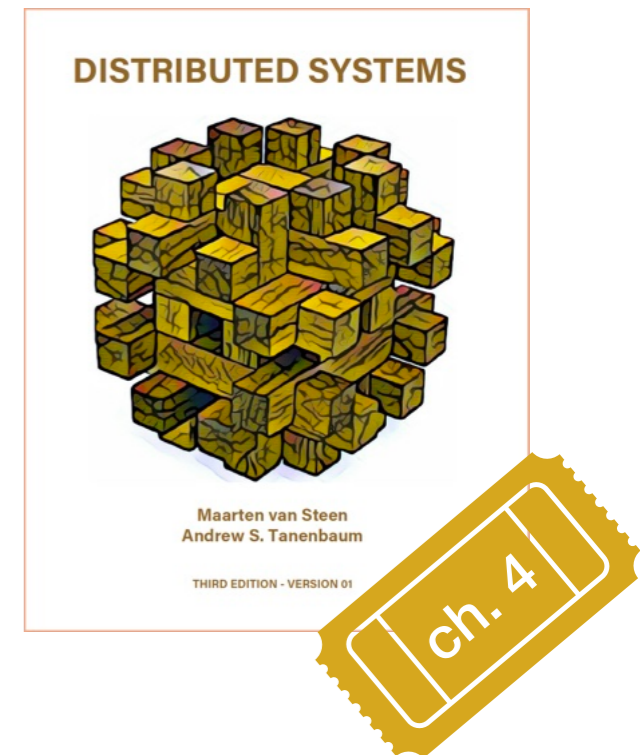


## Asynchronous communication (half duplex)





- Foundations
- Remote Procedure Calls
- Message Oriented Communication
- Multicast communication (epidemic algorithms)



Coordination

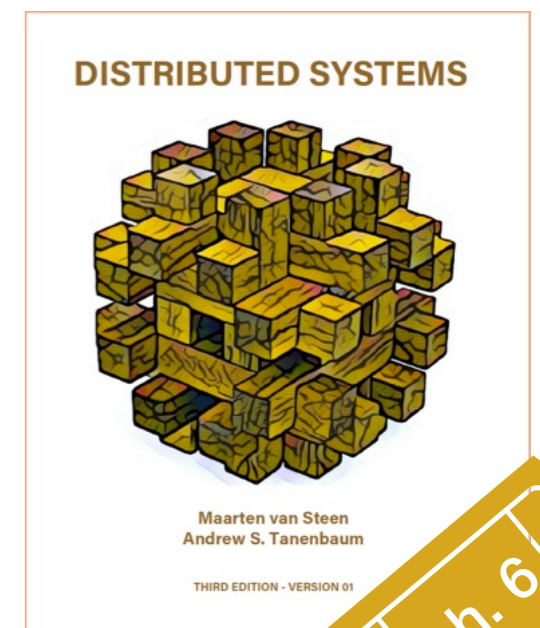


# 3 - Coordination

Strong separation between processing and coordination.

A system is a collection of autonomously operating processes.

In this model, coordination encompasses the communication and cooperation between processes.



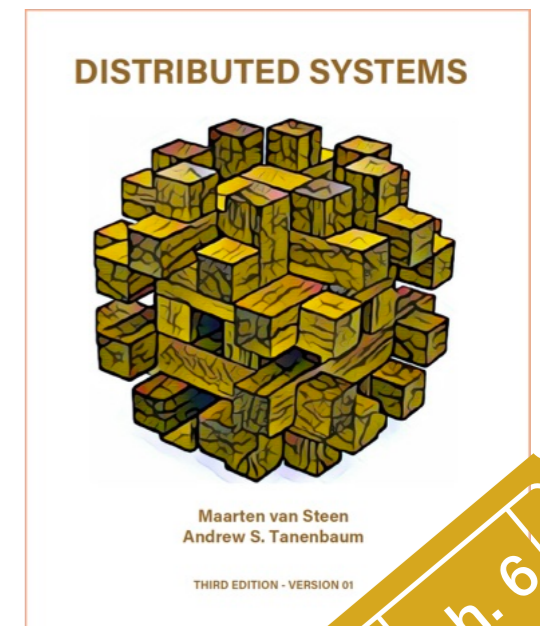
# 3 - Coordination

Synchronization and coordination are two closely related phenomena.

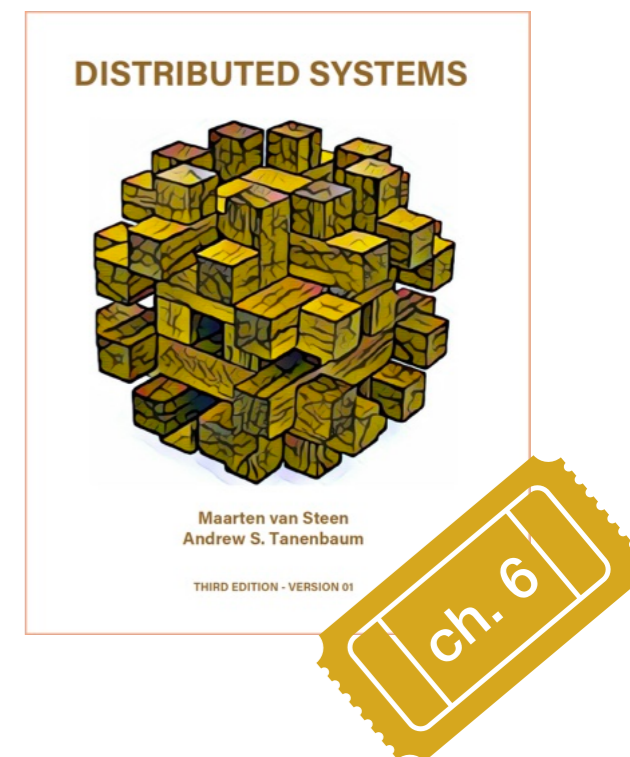
In **process synchronization** we make sure that one process waits for another to complete its operation.

When dealing with **data synchronization**, the problem is to ensure that two sets of data are the same.

When it comes to **coordination**, the goal is to manage the interactions and dependencies between activities in a distributed system.



- Clock synchronization
- Logical clocks
- Mutual exclusion
- Leader election algorithms
- Gossip based coordination

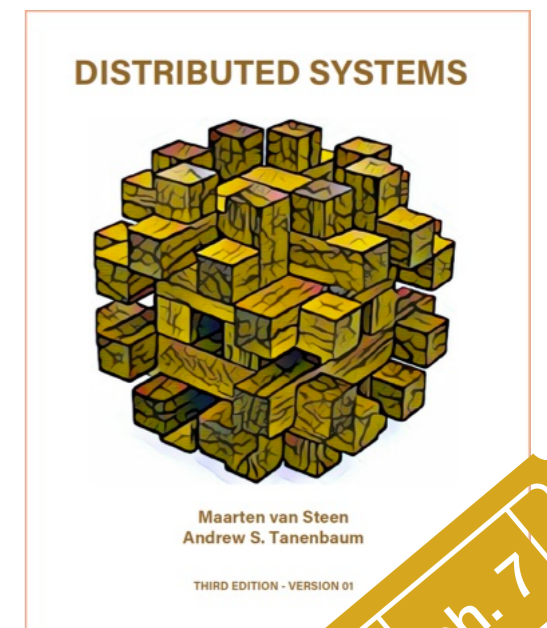


Availability,  
consistency,  
and ACID  
properties

4

## Replication Models

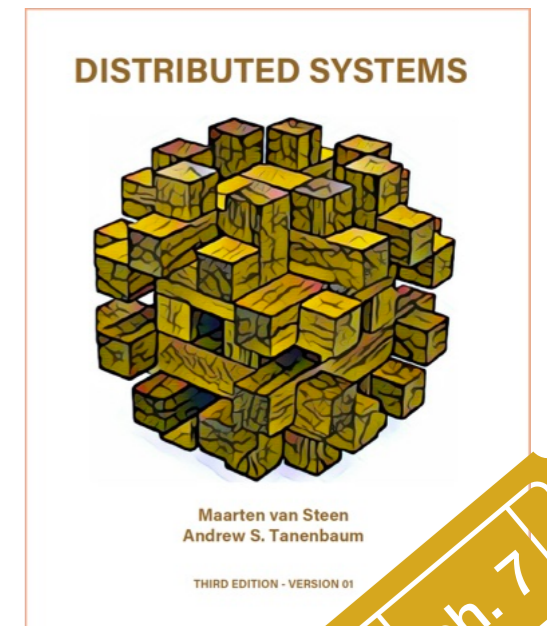
- Active replication (state machine)
- Passive replication
- Other redundancy models for storage systems: caching, erasure coding, etc.



Wiesmann, Matthias, et al. "Understanding replication in databases and distributed systems." *Proceedings 20th IEEE International Conference on Distributed Computing Systems*. IEEE, 2000.

## Consistency Models

- Casual consistency
- Eventual consistency
- Strong consistency
- Read-your-writes consistency
- ...





# ACID transactions

atomic  
consistent  
isolation  
durability



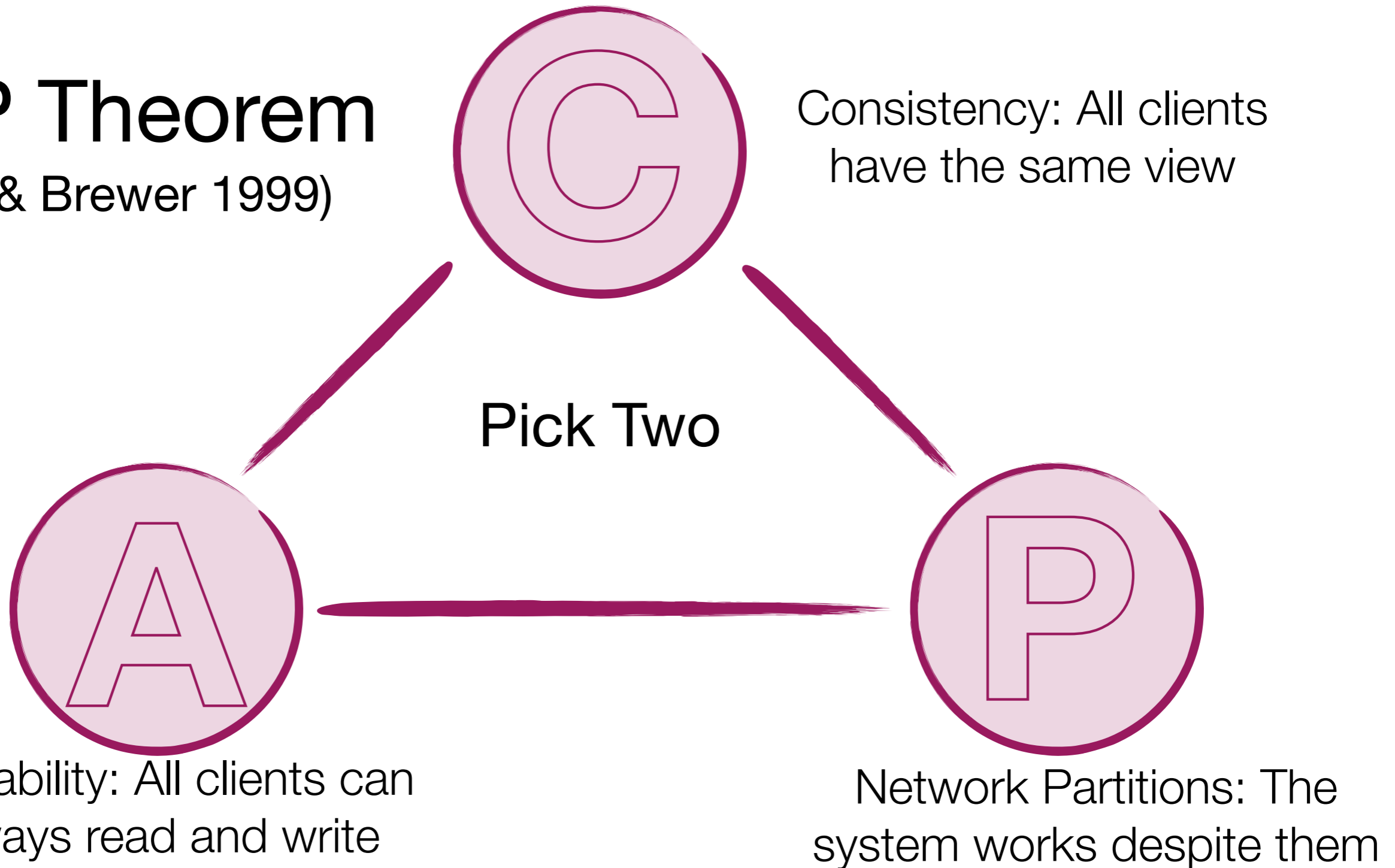
**all or nothing**

# Storage as a Service (StaaS)

- Data model
- Data dispersion
- Data consistency
- Data transaction
- Data management cost

Mansouri, Yaser, Adel Nadjaran Toosi, and Rajkumar Buyya. "Data storage management in cloud environments: Taxonomy, survey, and future directions." *ACM Computing Surveys (CSUR)* 50.6 (2018): 91.

## CAP Theorem (Fox & Brewer 1999)



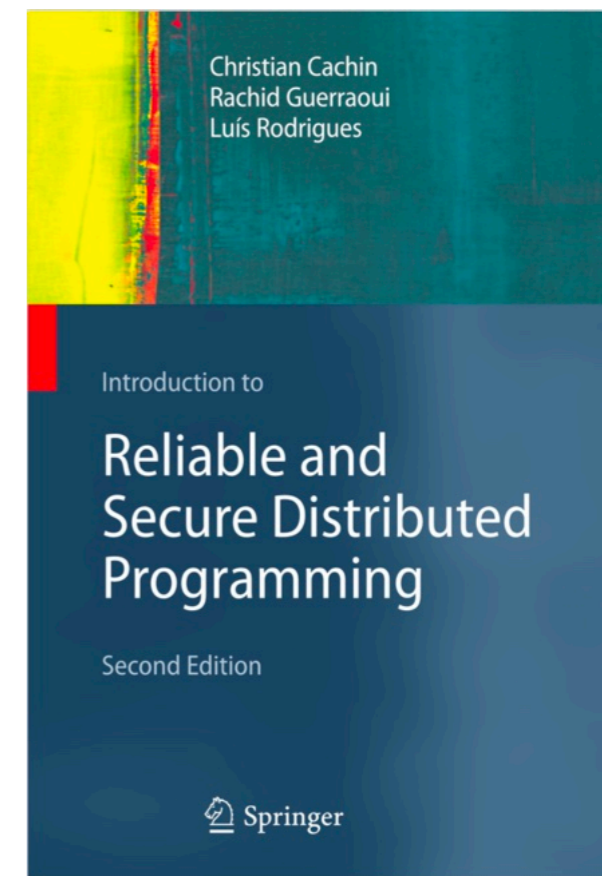
Brewer, Eric. "CAP Twelve years Later: how the." *Computer 2* (2012): 23-29.

Fault tolerance  
and reliability



# 5 - Fault tolerance and reliability

- Replication consists in making a centralized service **highly available** by executing several copies of it on different machines that are assumed to **fail independently**.
- The different copies must be maintained in a **consistent state**. The illusion of one highly available service would fall apart and be replaced by that of several distributed services, each possibly failing independently.



# 5 - Fault tolerance and reliability

## State machine replication

General approach to building fault-tolerant systems

If replicas are deterministic, one of the simplest ways to guarantee full consistency is to ensure that all replicas receive the same set of requests in the **same order**.

Typically, such guarantees are enforced by an abstraction called **total-order broadcast**: the processes need to agree here on the sequence of messages they deliver.

*Schneider F. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial.  
ACM Computing Surveys, 22(4):299–320, Dec. 1990.*

# 5 - Fault tolerance and reliability

## AVAILABILITY

What means 99.99% (four 9's)?

Expected downtime per year:

$$(1 - 0.9999) * 525600 = 52.56min$$

$$(1 - 0.995) * 525600 = 1d 19h 48min$$

Homework: Check AWS reliability

# 5 - Fault tolerance and reliability

## DURABILITY

What means 99.9999999999% (eleven 9's)?



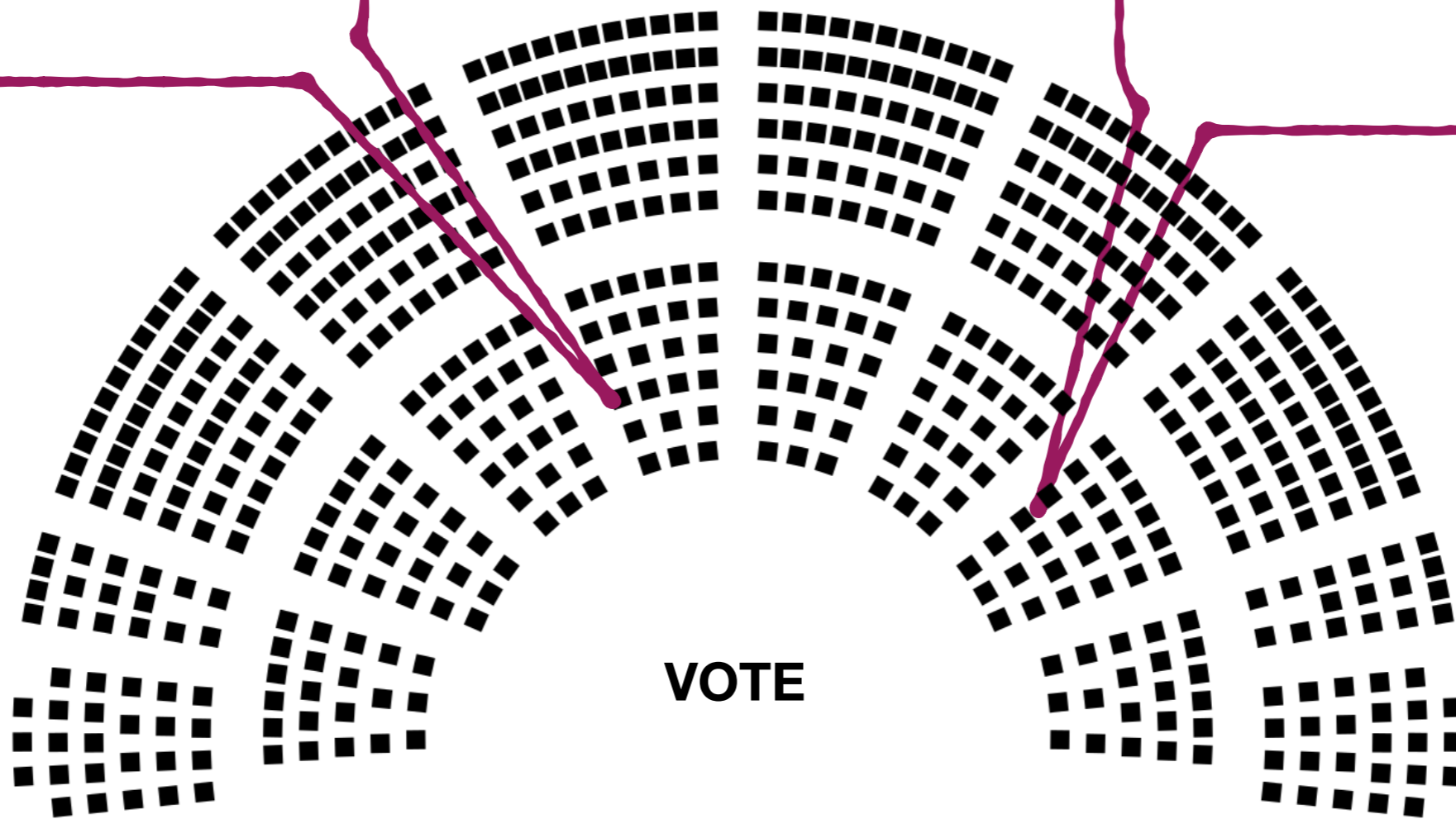
Consensus



**BURGER!**

**?**

**PIZZA!**



**VOTE**

## The part-time parliament, Lamport (PAXOS 1990-1998)

---

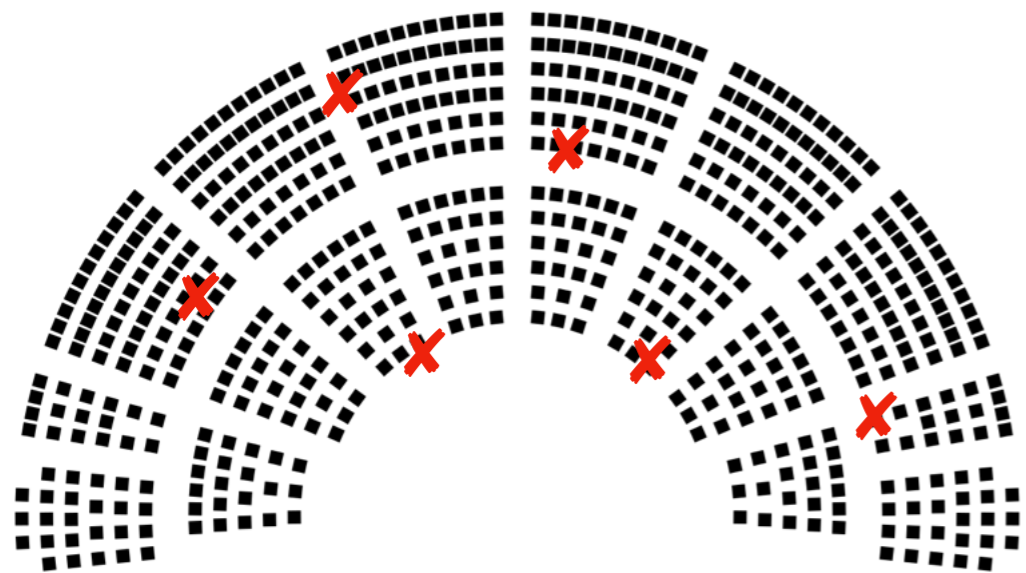
Recent archaeological discoveries on the island of Paxos reveal that the parliament functioned despite the peripatetic propensity of its part-time legislators. The legislators maintained consistent copies of the parliamentary record, despite their frequent forays from the chamber and the forgetfulness of their messengers. The Paxos parliament's protocol provides a new way of implementing the state machine approach to the design of distributed systems.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*network operating systems*; D.4.5 [**Operating Systems**]: Reliability—*fault-tolerance*; J.1 [**Computer Applications**]: Administrative Data Processing—*government*

General Terms: Design, Reliability

Additional Key Words and Phrases: State machines, three-phase commit, voting

---



- Paxos: single-value
- Multi Paxos: a sequence of values

Model assumptions: asynchronous, non-Byzantine communications model

## **Paxos made simple, Lamport (2001)**

The Paxos algorithm, when presented in plain English, is very simple.

\*\*\*\*

## **Paxos made live, an engineering perspective Chandra et al. (2007)**

- algorithmic gaps in the literature
- software engineering challenges, and
- unexpected failures

\*\*\*\*

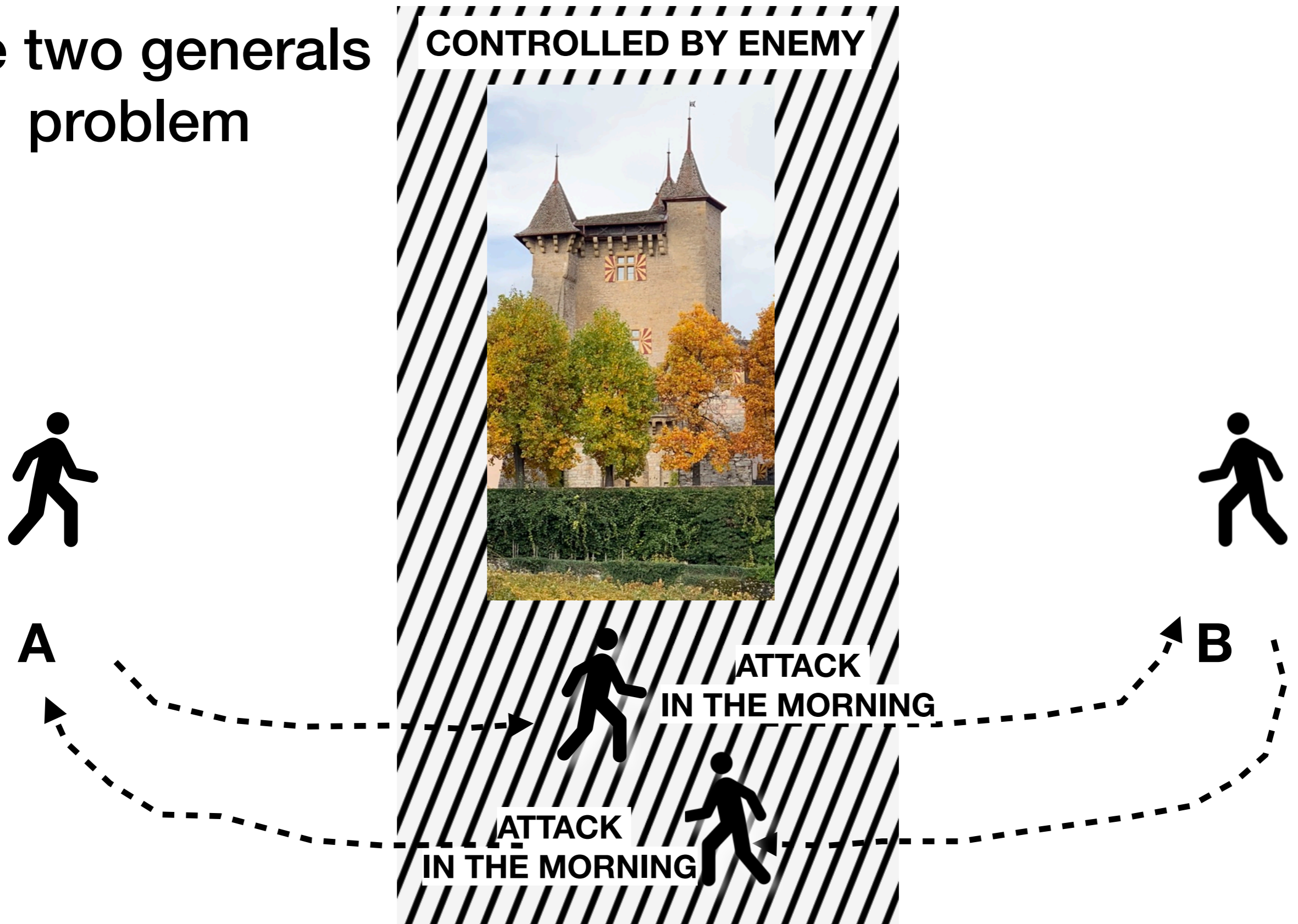
## **In search of an understandable consensus algorithm, Ongaro and Ousterhout (2014)**

- Raft algorithm is equivalent to multi Paxos.
- it separates the key elements of consensus into leader election, log replication, and safety.

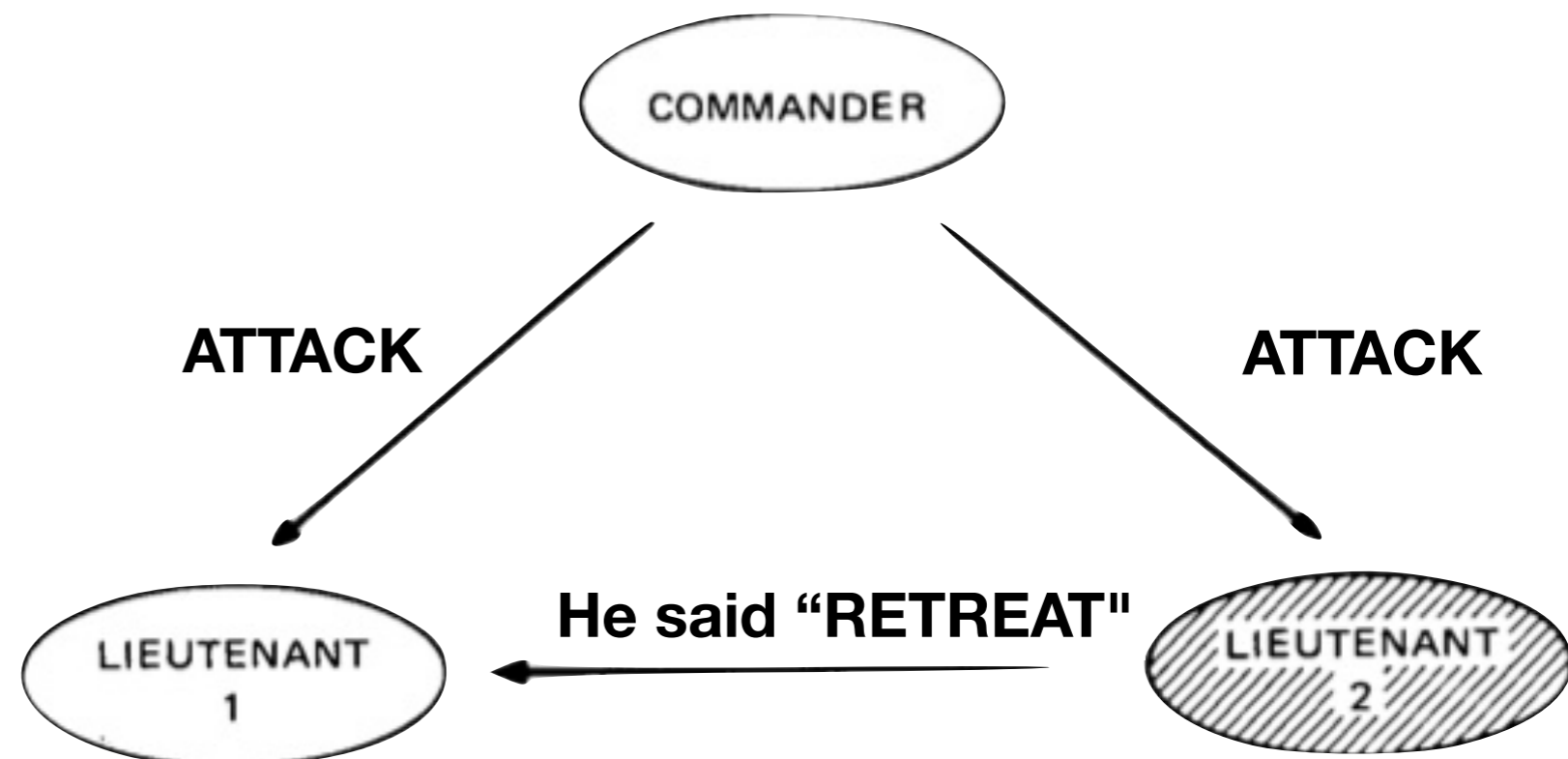
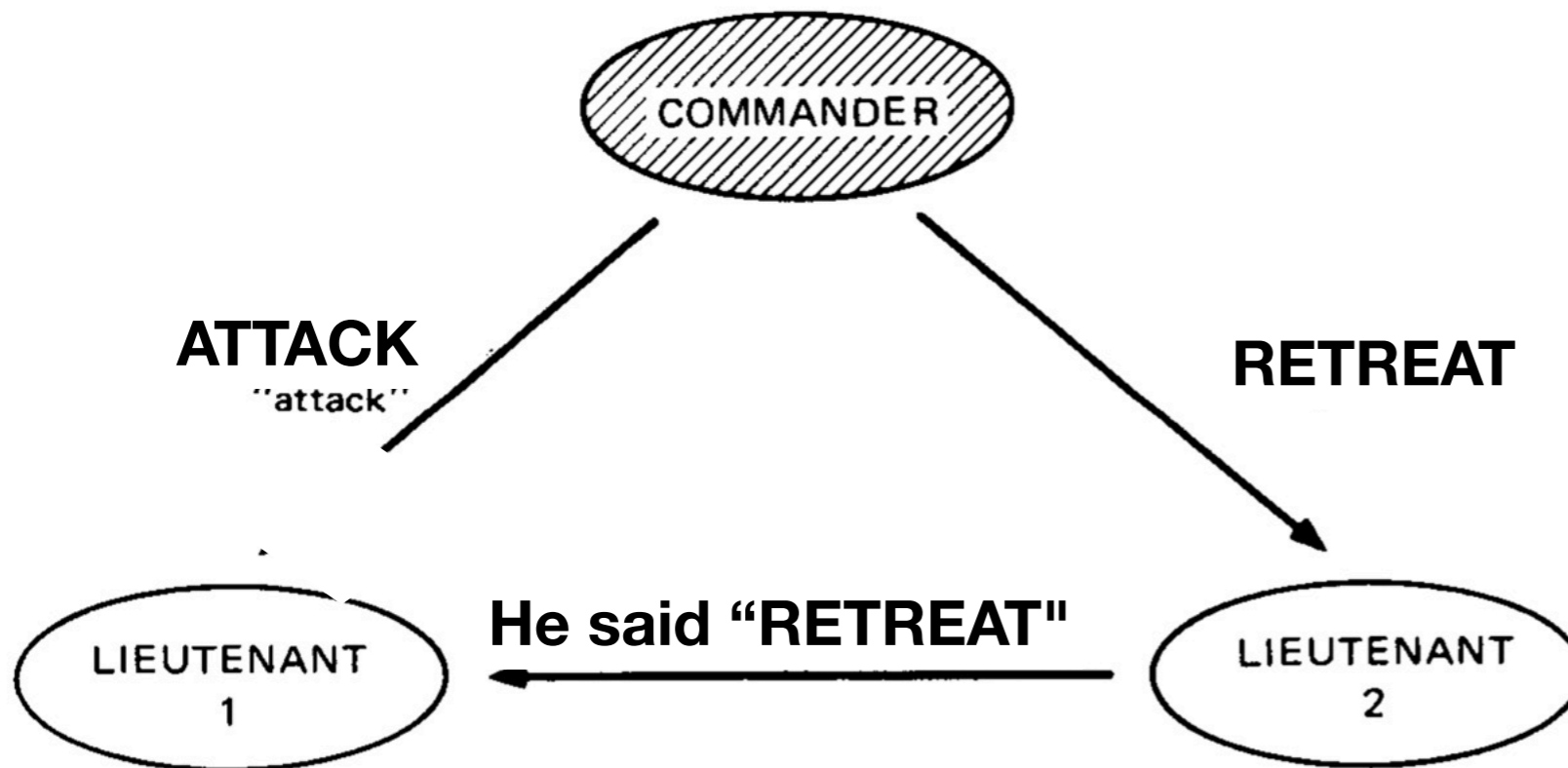


## The Byzantine generals

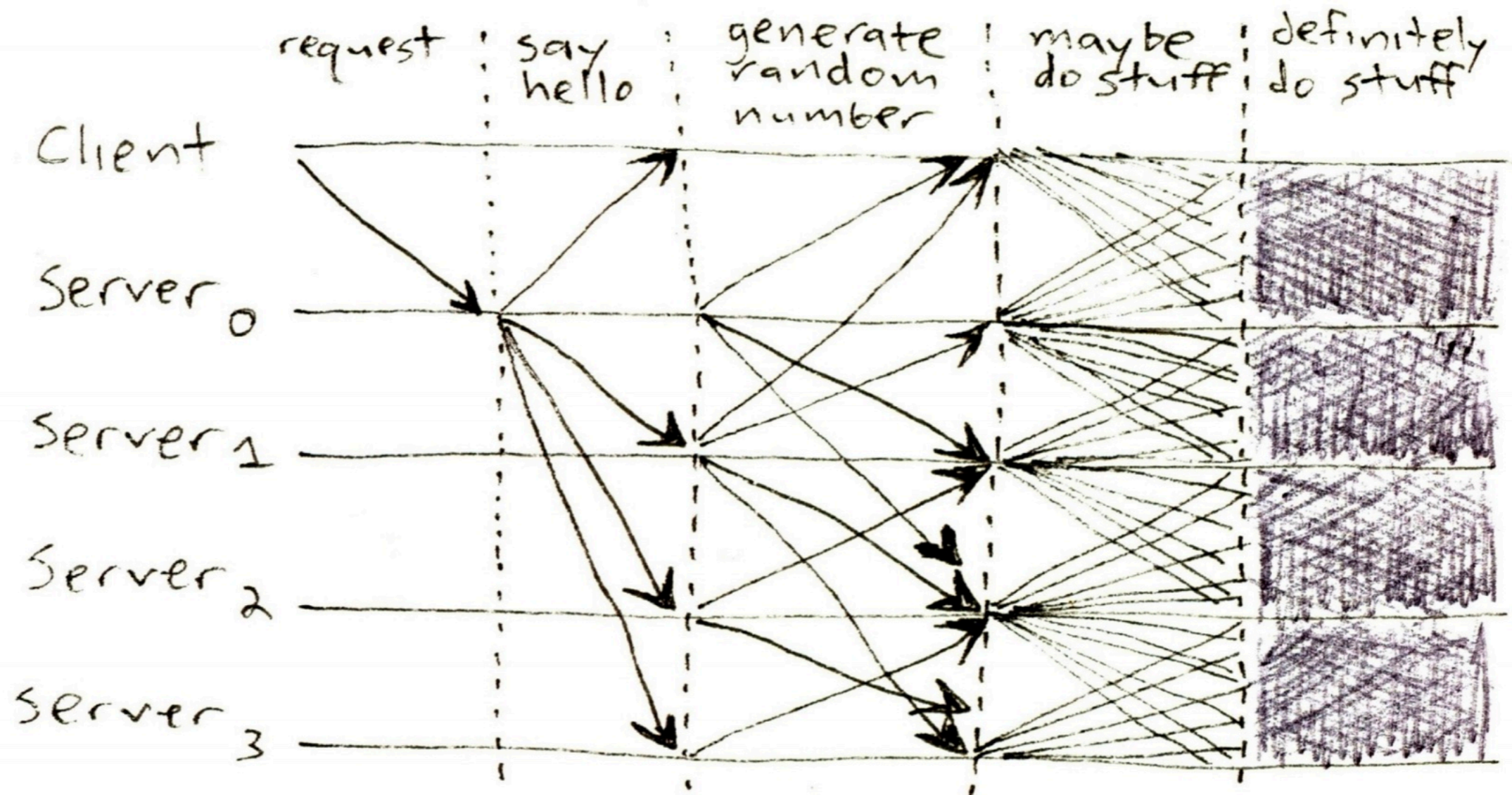
## The two generals problem



## The Byzantine generals



# The Saddest Moment



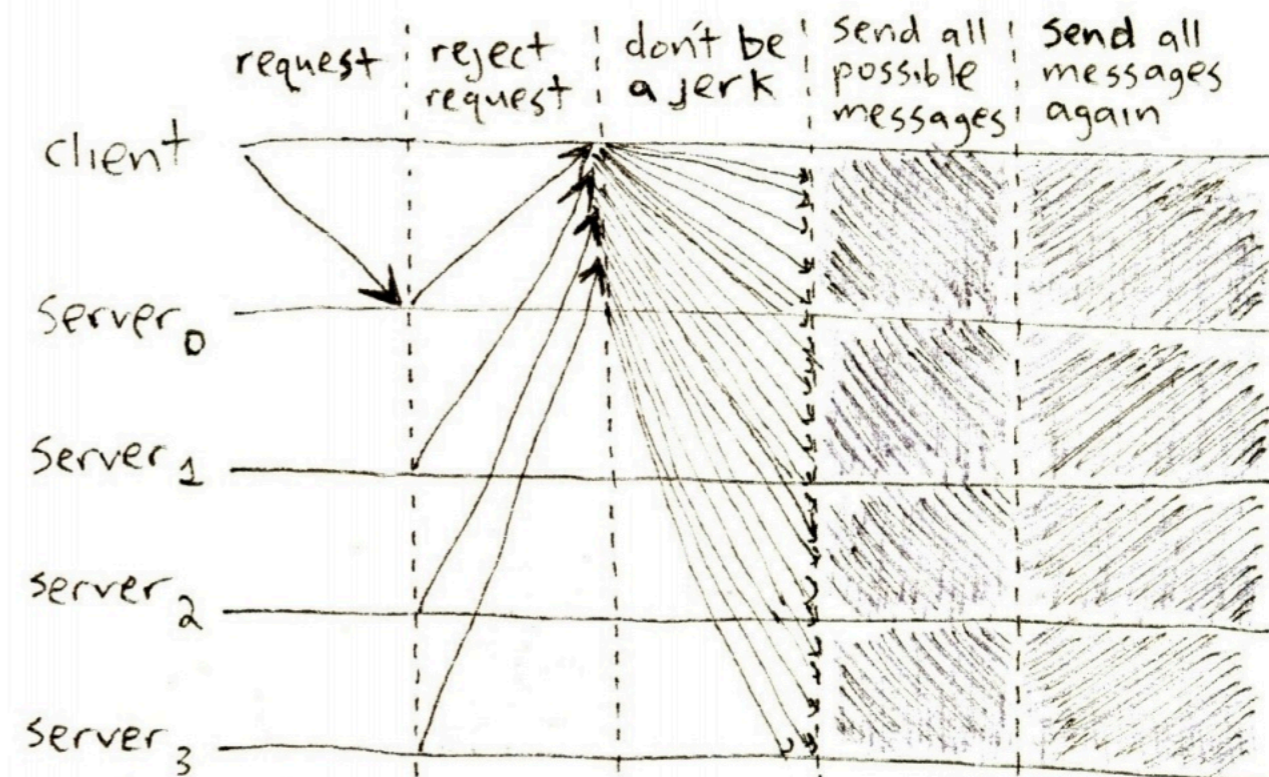
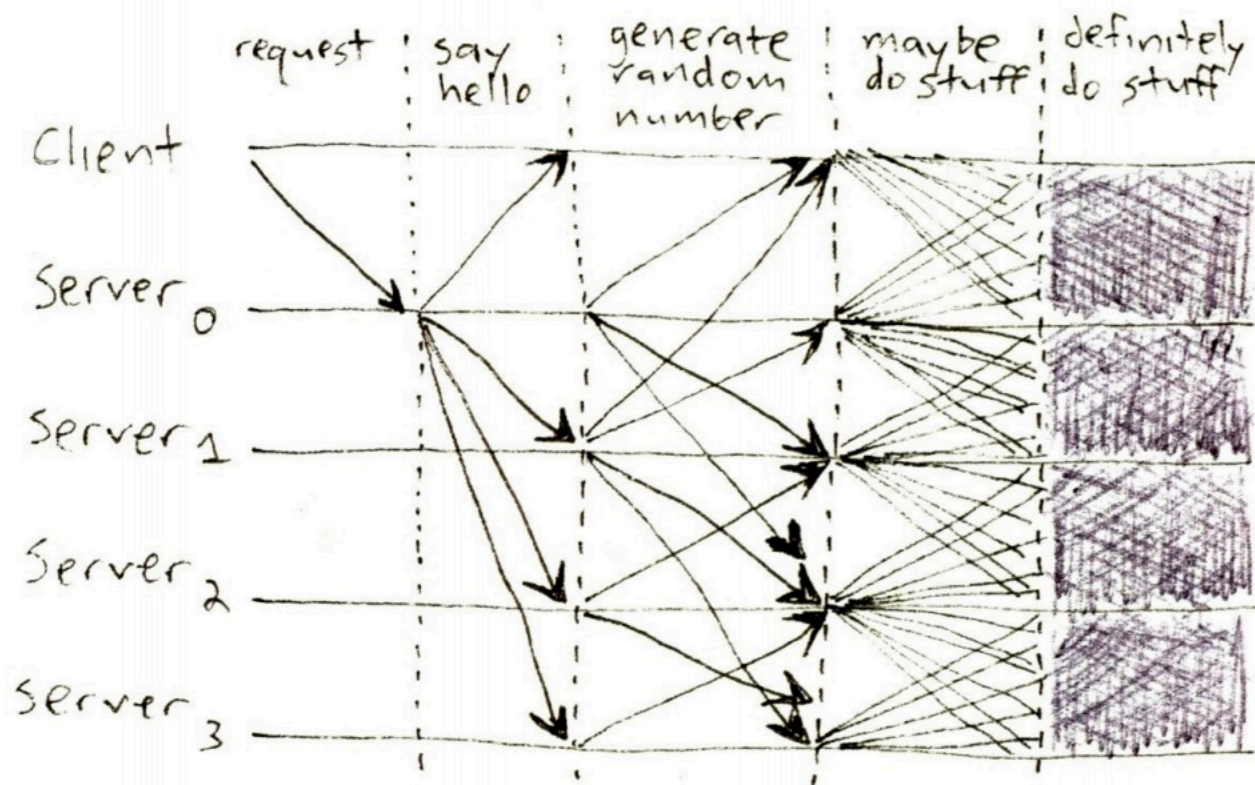
**Figure 1:** Typical Figure 2 from Byzantine fault paper: Our network protocol



<https://scholar.harvard.edu/files/mickens/files/thesaddestmoment.pdf>



# The Saddest Moment



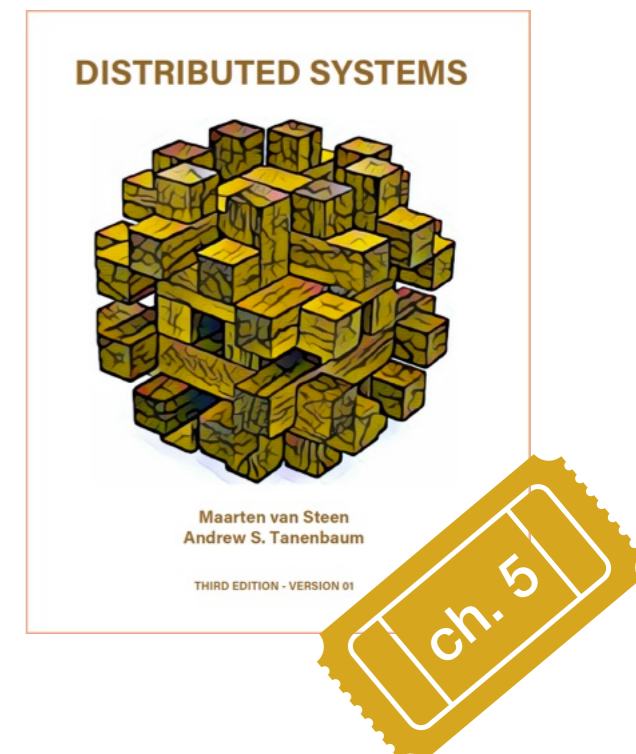
**Figure 2:** Our new protocol is clearly better

Naming

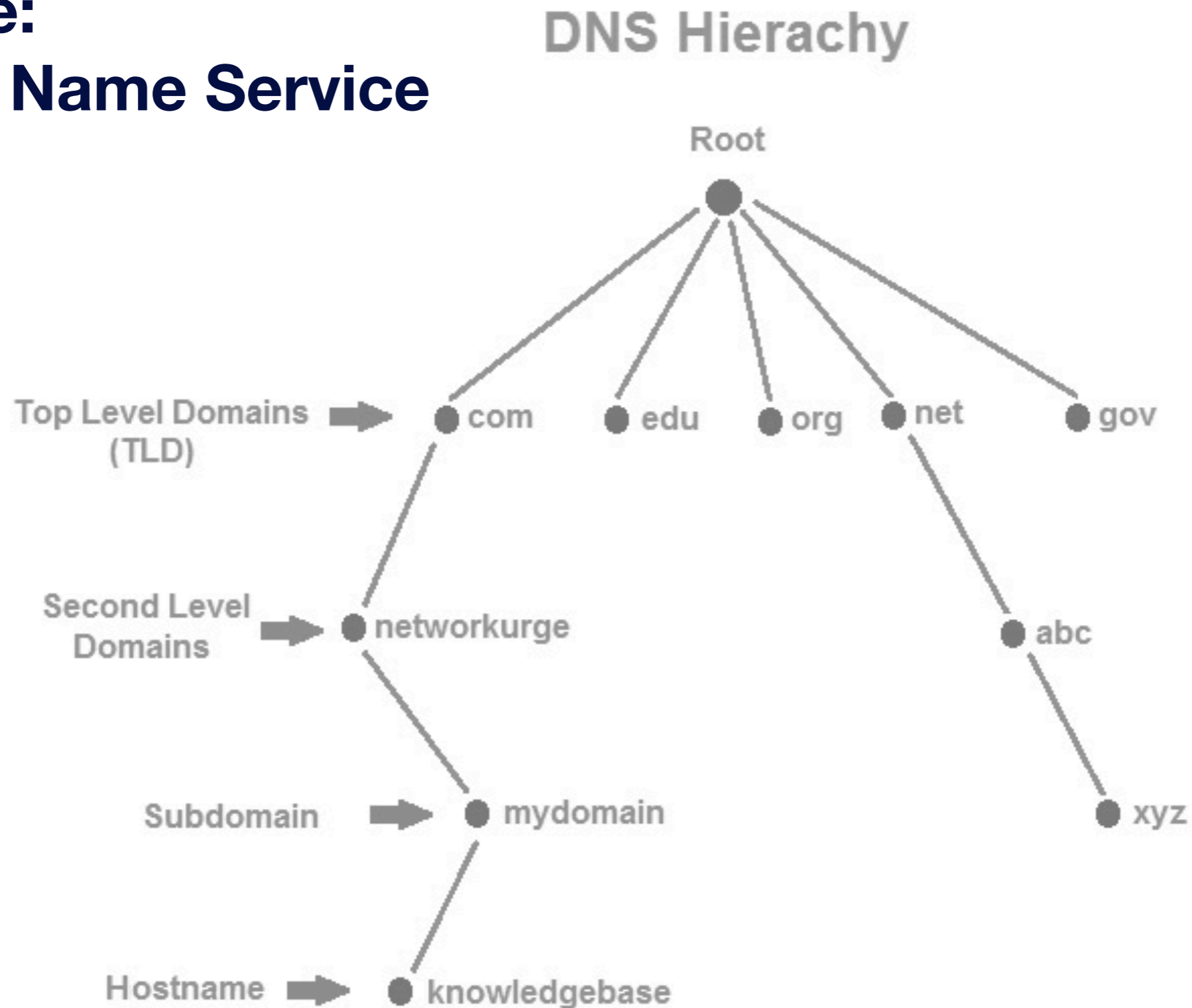


Names are used to share resources, to uniquely identify entities, to refer to locations.

With a naming system, a name can be resolved to the entity it refers to.



## Example: Domain Name Service (DNS)



Short talks/  
interviews  
by "virtual  
guests"  
(optional)



# Virtual guests (optional)

Featuring :

- Internet with ***Robert (Bob) Kahn***
- Robust networks with ***Radia Perlman***
- Data abstractions and Distributed computing with ***Barbara Liskov***
- Algorithms & Verification with ***Leslie Lamport***
- Cloud computing with ***Tushar Chandra***
- Digital Preservation with ***Vint Cerf***
- Fixing the Internet with ***Jaron Lanier***
- IPFS with ***Juan Benet***
- ...

# Tips to give presentations



Security

88



Shared resources often have a high intrinsic value

Security's principle components:  
confidentiality - integrity - availability

Including :

- User authentication
- Access control
- Secure communication
- Data integrity proofs
- Resource usage control
- Privacy mechanisms

# Design and Operations




## Pitfalls in Distributed Systems

Peter Deutsch and James Gosling

- The network is **reliable**
- The network is **secure**
- The network is **homogeneous**
- The **topology** does not change
- **Latency** is zero
- **Bandwidth** is infinite
- **Transport cost** is zero
- There is one **administrator**

# 9 - Design and Operations

- Failure domains
- Transient failures
- Chaos engineering (Netflix)
- Black Swarms (Slack)
- Abstracting the Geniuses Away from Failure Testing (ACM Queue'17)
- Network-partition failures in cloud systems (OSDI '18)
- Building for performance: The tail at scale (CACM'13)
- Cold, warm, hot data
- Trade-offs
- **NALSD (hands-on activity)** 

Evaluation

10

# Meaningful metrics

## for whom?

- SLI | SLO | SLA
- Monetizable daily active users (mDAU)
- Economics of serverless computing
- Long-term costs of digital archives

# Meaningful metrics

## for whom?

- Complexity (message overhead)
- Meaningful metrics for reliability

# Meaningful metrics


## for what?

- Long-term costs
- Energy consumption is critical
- Centralization: what do you measure?
- Fallacies in Evaluating Decentralized Systems



# Meaningful metrics

## how?

- **SLI | SLO | SLA (hands-on activity)** 
- How not to lie with statistics
- High quality plots
- There are no perfect solutions. Show trade-offs

一期一会

“One time, one meeting”