Distributed Data Processing Systems

Dr. Alexandru Uta, LIACS
a.uta@liacs.leidenuniv.nl

What do I do?

- Low-level systems work, at the intersection of hardware and software
- Large-scale distributed systems
- Data processing systems (Spark, Hadoop, graph processing)
- Performance evaluation
- Designing new systems
- Topics on:
    - networking,
    - operating systems
    - serverless
    - resource management and scheduling
- Novel types of hardware: GPUs, TPUs, Intel KNL, FPGA

Teaching Assistants

- Yuxuan Zhao → y.zhao@liacs.leidenuniv.nl
- Weikang Weng → w.weng@liacs.leidenuniv.nl

- Use them wisely !!!

- Contact them for questions
  - Deadlines
  - Assignments
  - Practical

- Mid-term evaluation for both assignments, arrange meeting with them!

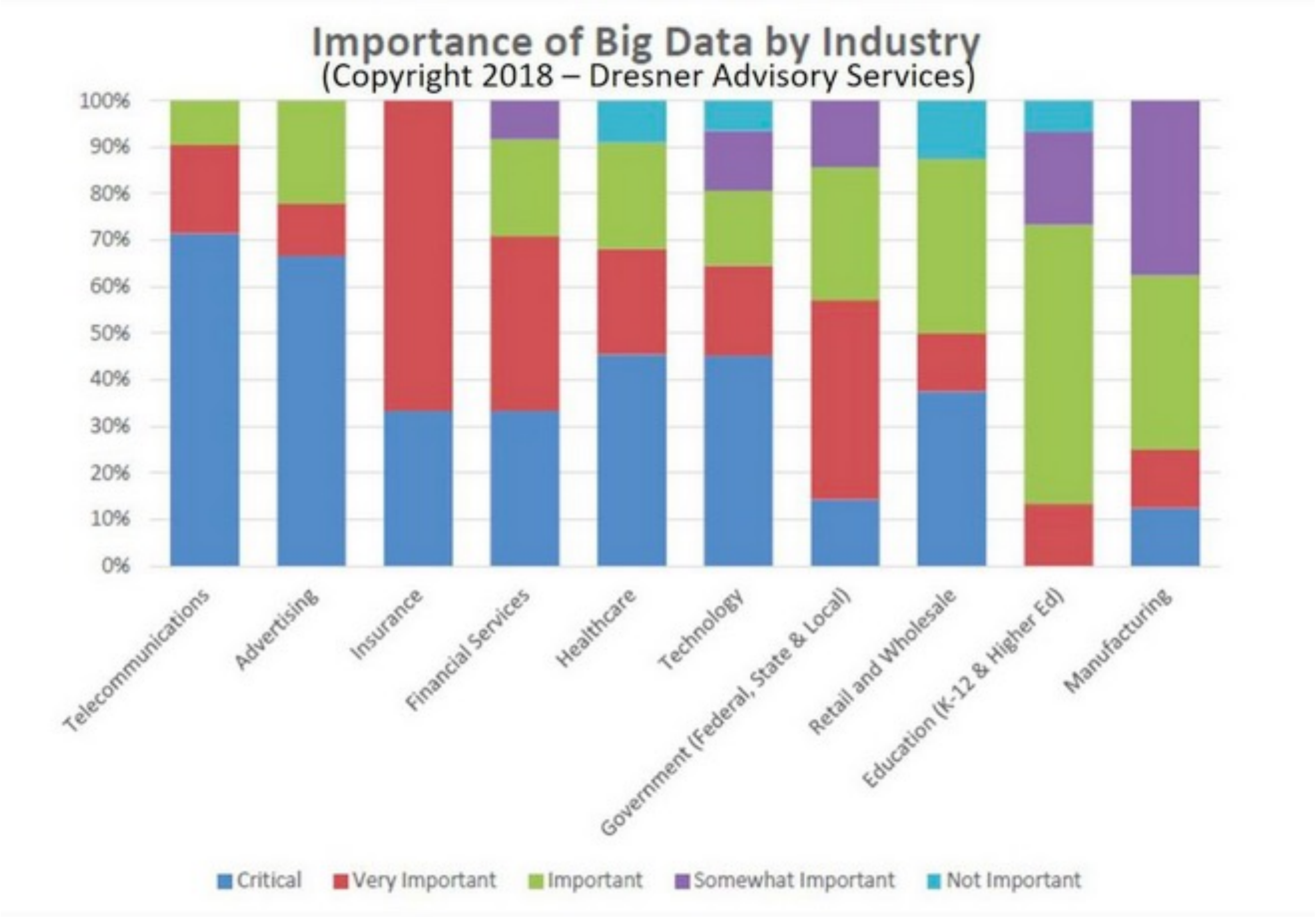
What do you do?

- 49 students registered by Sept 7
- Need to know who takes the course in max 1 week!

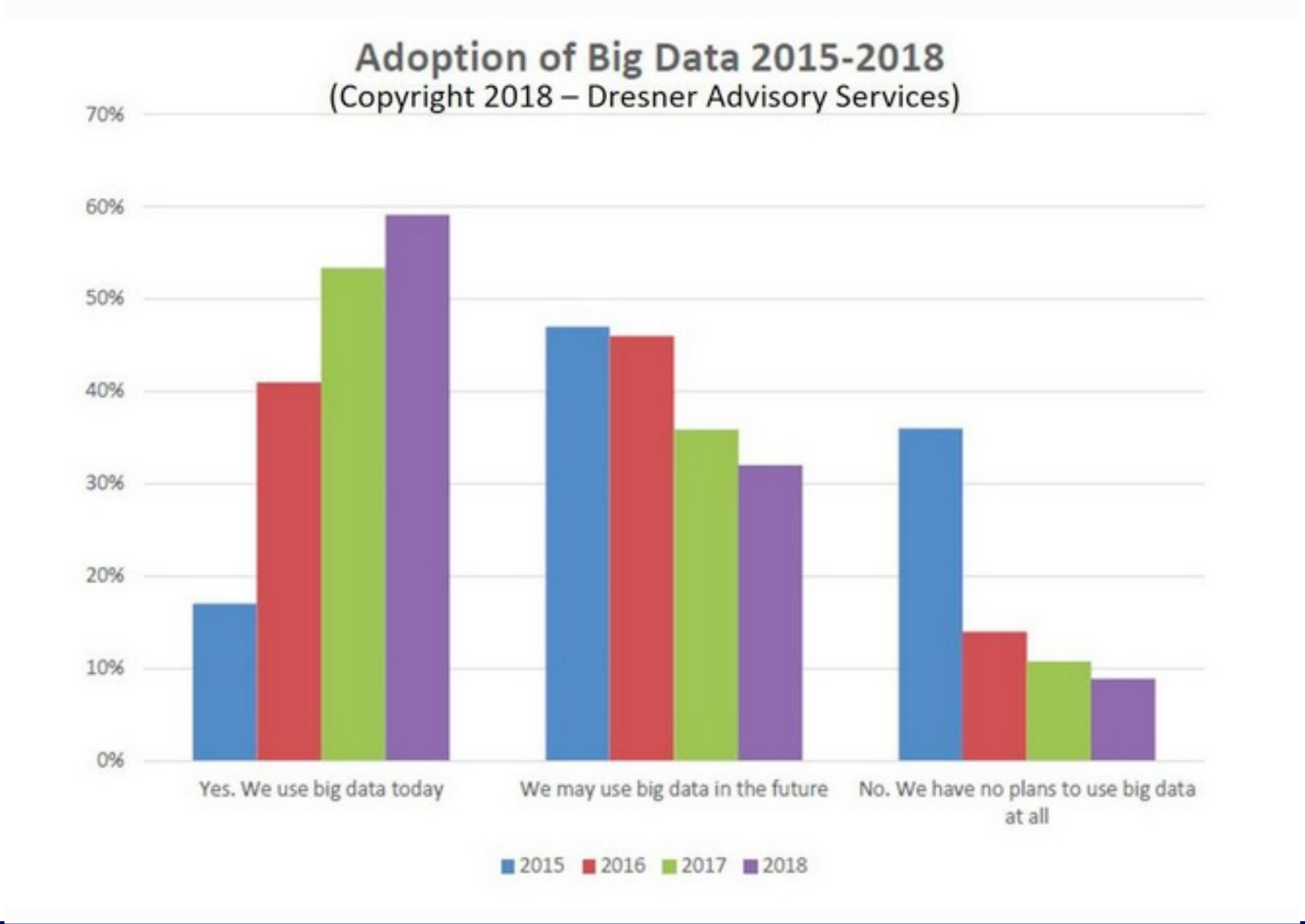
[A Practical, Systems View on...] Distributed Data Processing Systems

Dr. Alexandru Uta, LIACS
a.uta@liacs.leidenuniv.nl

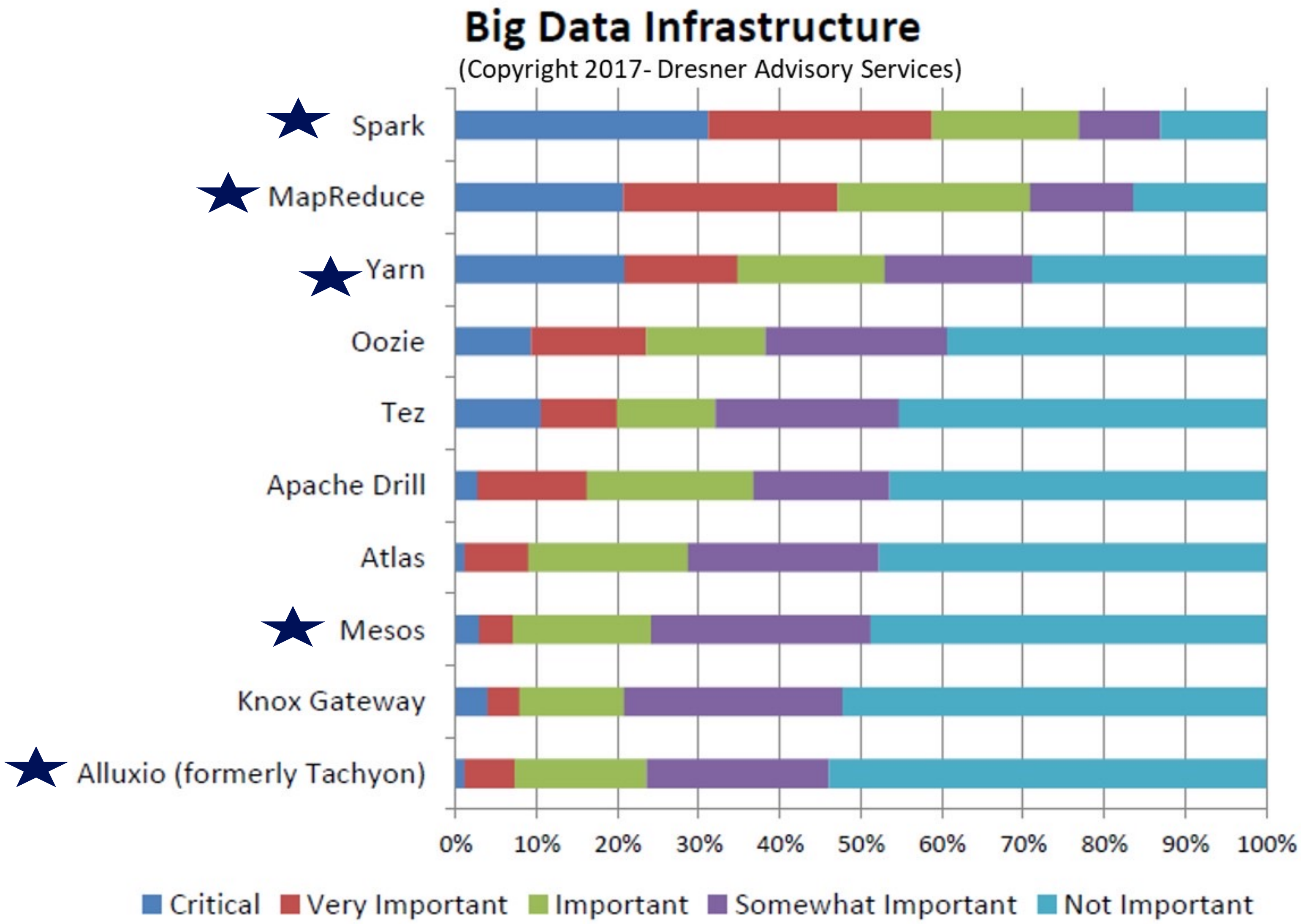
Why distributed data processing?



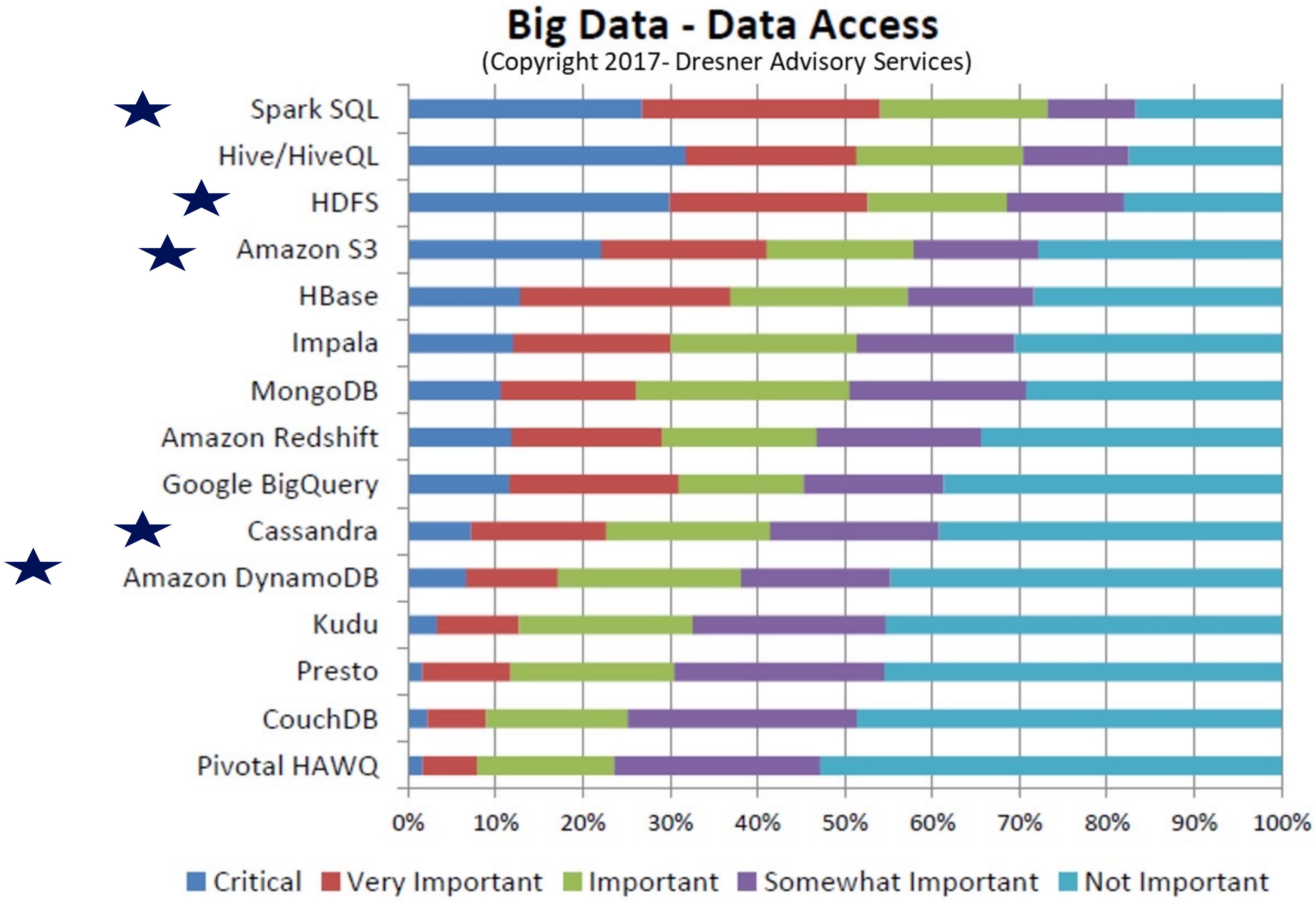
Why distributed data processing? (2)



Why distributed data processing? (3)



Why distributed data processing? (4)



Why distributed data processing? (5)

- All top computer systems conferences have topics on
  - Systems for big data
  - Distributed systems
  - Distributed file systems
  - Distributed storage systems

- all with a strong focus on **performance**

- E.g., Spark paper > 4,500 citations
- E.g., Original MapReduce paper > 13,000 citations

Extremely important topics.

How do we learn about all this?

- **Theoretical knowledge** from reading modern research articles

- **Practical knowledge** from
  - (1) deploying
  - (2) running
  - (3) analyzing performance
  - (4) designing own (toy) distributed system

Course format

- **Lectures** (every Friday)
  - 4 by lecturer
  - 2 lectures by TAs
  - 1-2 invited talks
  - Several lectures composed of **student presentations** (graded, but not mandatory)

- **Practical assignments** (graded)
  - (1) Reproducibility / experimental study
  - (2) Build your own distributed system

- **Position paper** (graded) (non-mandatory)

Prior knowledge needed!

We assume you know the following:

- Networks, Operating Systems, Software engineering, basic statistics
- Good to have but not mandatory: Parallel programming, Cloud computing

Programming skills:

- Scripting: bash, python
- Java, C, C++

Concepts:

- Threads vs. processes
- Sockets, TCP, UDP, serialization

Grading

- **Assignment 1** – Reproducibility Study – 35% (mandatory)
- **Assignment 2** – Build your own DS – 35% (mandatory)
- **Article presentation** – 15% (optional)
  - In-class activity – 5% (optional)
- **Position** paper – 10% (optional)
- NO EXAM!

Assignment 1 – Reproducibility study

- Work in teams of 2
- Teams of 1 get the same workload!!!
- Pre-selected list of articles we analyze (check brightspace assignment 1)
- Two teams may pick the same article, but report and work **must be significantly different**
- Pick **two experiments** from this article, and try to reproduce!

- In general, these are Spark, Hadoop, HDFS papers
- If the app in the paper is not available: **use Spark/Hadoop example apps (there are quite a few)**
- **Use only the scale that is available to you**
  - E.g., paper uses 100 machines → you can use 16 (will not lower grade)
  - E.g., paper uses 100TB of data → you can use 100GB (will not lower grade)
  - If unsure, please ask!

Assignment 1 – Requirements

- Reproduce <u>two</u> experiments (mind the mentions about limitations)

- Write a report about your findings
  - What experiments did you choose and why?
  - How did you replicate them?
  - Are the results similar or different?
  - What techniques did you apply to ensure result correctness and reproducibility?
  - What did you learn from this study?

- Your report should include: graphs, tables, diagrams (not all may apply).
- **Grade**: 40% report quality + 60% work quality.

Assignment 2 – Build your own DS

- Several pre-defined assignments
    - Build simple MapReduce engine
    - Build Dropbox-like service
    - Build a storage service compatible with the Amazon S3 API
- Own idea is welcome, please contact me

- Step 1: Design and build the system
- Step 2: Write a report

Assignment 2 – Requirements

- Your DS should have 2-3 key features:
  - Performance
  - Fault-tolerance
  - Consistency
  - Scalability
- Write a report about your DS
  - What is the design of your DS? Why?
  - How did you implement it?
  - How did you evaluate your DS?
  - Why did you choose those experiments?
  - What are the results?
  - What did you learn from building it?
- **Grade**: 40% report quality + 60% work quality.

Assignment Deadlines

- Assignment 1 – week 8 (recommended start week 2)
- Assignment 2 – week 15 (recommended start week 9)
- On Friday of that week, at 23:59
- Consultation time: contact the TAs individually

Mid-term Presentations

- Check Schedule on Brightspace
- Arrange mid-term presentation with Tas

- No mid-term presentation == -2p from final grade !!!

- No exceptions (unless serious motive)

Student Presentation

- Each team selects a paper from a given list (top-tier articles selected by instructor) (check brightspace recommended reading)
- **Announce before Sept 30!!!**

- One topic per week, 3-4x presentations per topic
- FIFO selections of articles/slots

- 15-20 mins presentation + 5-10 questions and discussion

- **Graded** – based on presentation quality & depth + Q&A
- **Attendance is highly recommended! (**otherwise you cannot get in-class activity grade)

Student Presentation Format

- Follow the **Why? What? How?** questions
- **Why** is this article important? **What** do the authors do? **How** do they do it?

- Be sure to follow these questions:

  0. Why is this topic important and timely?
  1. What are the main challenges being addressed?
  2. What are the main technical contributions?
  3. What are the main conceptual contributions?
  4. How are these contributions evaluated?
  5. What are interesting directions to continue? (Future work)
  6. What is the main take-home message?

Position Paper

- **1p (10%)** of the grade, **not mandatory**
- You cannot get 10 without it

- **Select 5 articles from recommended literature**
- Find an idea (or a list of ideas) that tie these articles together
- Argue why this idea ties these articles together and explain it in your paper
- Identify what is your *position* on the topic, e.g.,
  - Clouds will always exhibit performance variability
  - Disaggregated hardware is the future
  - Serverless is the most efficient form of computing

- Give arguments, explain your position
- Write up to 4-page paper

Week	Date	Lecture	Deadline	Deadline Time
1	09/09	11:15 - Lecture 1 - Intro, DS, Architectures, RPC		
2	16/09	14:15 - Lecture 2 - Distributed Data Processing Systems, DAS demo	**Start Assignment 1**	
3	23/09	14:15 - Lecture 3 - Performance Evaluation		
4	30/09	14:15 - Lecture 4 - Serverless	**Announce presentation article**	
5	07/10	14:15 - Lecture 5 - Consistency, Fault-tolerance, Consensus	**Midterm for Assignment 1**	
6	14/10	14:15 - Lecture 6 - Middleware and OS topics for DS		
7	21/10	14:15 - Invited Lecture		
8	28/10	14:15 - Student Presentations		
9	04/11	14:15 - Student Presentations	**Deadline Assignment 1, Start Assignment 2**	4/11, 23:59
10	11/11	14:15 - Student Presentations		
11	18/11	14:15 - Student Presentations		
12	25/11	no session	**Midterm for Assignment 2**	
13	02/12	no session		
14	09/12	no session		
15	16/12	no session	**Deadline Assignment 2**	16/12, 23:59
16	23/12	no session	**Deadline position paper**	23/12, 23:59

6 EC = 168 hrs

- Assignment 1 = 50 hrs
- Assignment 2 = 50 hrs
- Presentation = 15 hrs
- Position paper = 10 hrs
- Online sessions (lectures etc.) = 28 hrs
- Self-study = 15 hrs

If you put in this much time, you can get a 10 (ten)!

Office hours

- Starting Week 4, we will offer “office hours”

- 1 x per week, contact the TAs

- Yuxuan Zhao --> y.zhao@liacs.leidenuniv.nl
- Weikang Weng --> w.weng@liacs.leidenuniv.nl

What is a distributed system?

"You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done."

Leslie Lamport

"A collection of autonomous computing elements that appears to its users as a single coherent system."

Tanenbaum & van Steen

"An application that executes a collection of protocols to coordinate the actions of processes on a network, such that all components cooperate together to perform a single or small set of related tasks."

Google University

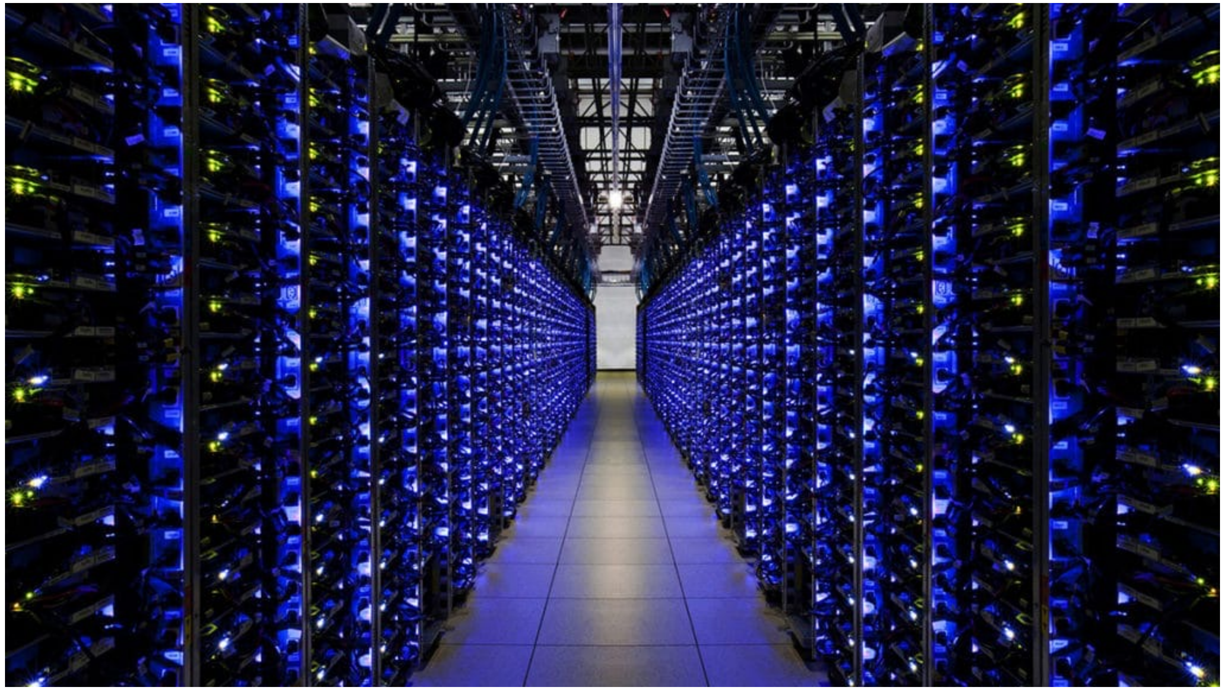
What is a distributed (data processing) system?

- "All of the above", but related to pre-processing, processing, storing, and getting knowledge out of large amounts of data.

- Group of 2+ constituents, could be heterogeneous
- Physically distributed, communicate through network
- Constituents orchestrate their actions
- There is a structure and/or organization
- Desirable non-functional properties
- Possible service-level agreements
- Touches large amounts of data

Large-scale Infrastructure

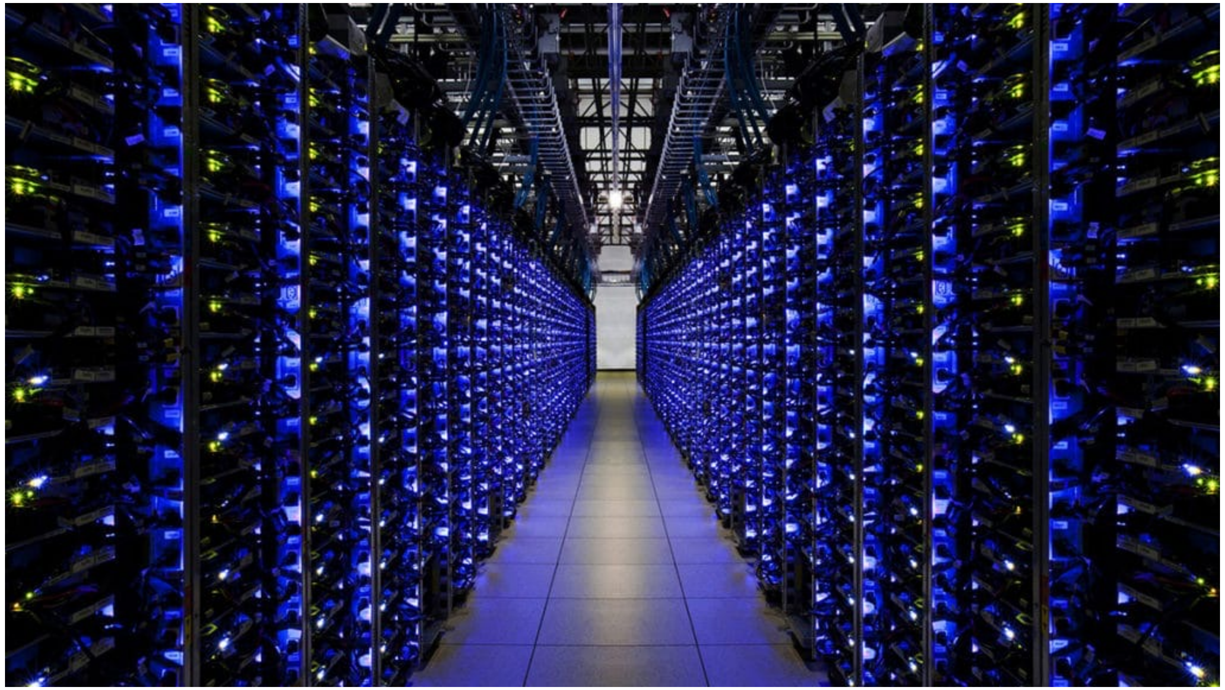
- Datacenters
- Clusters
- Grids
- Clouds
- Supercomputers



Large-scale Infrastructure

- **Datacenters**

- Clusters
- Grids
- Clouds
- Supercomputers

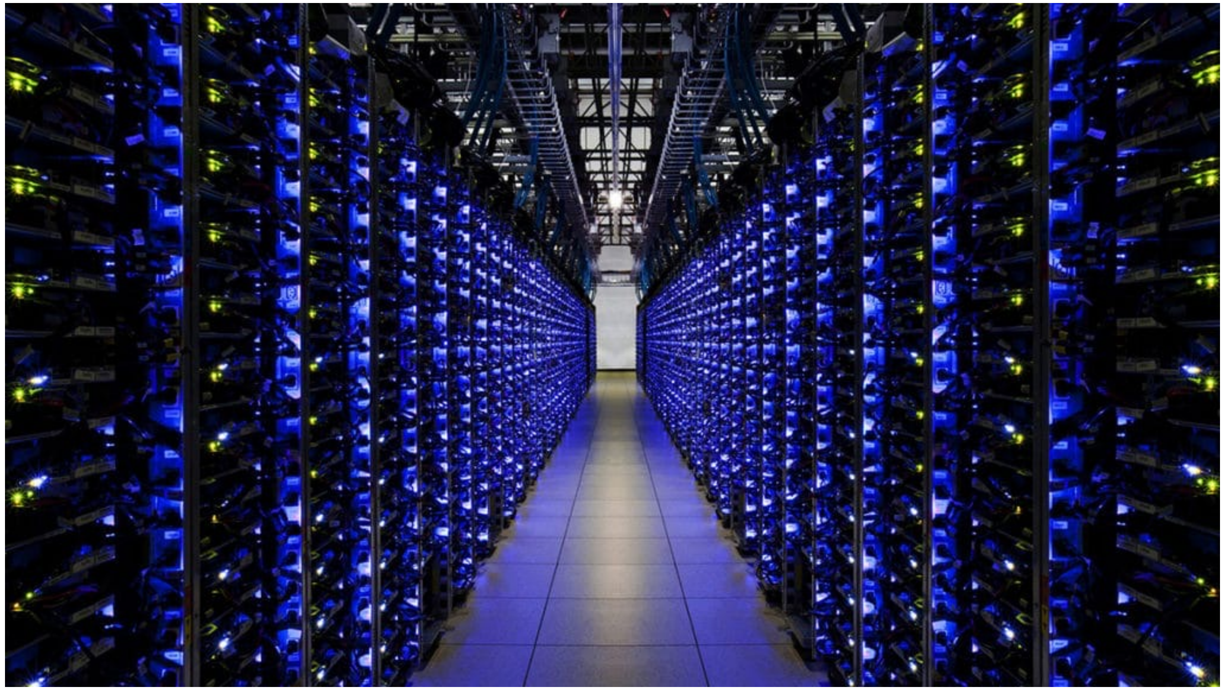


- Definition = dedicated space for computer systems and associated equipment

Large-scale Infrastructure

- Datacenters

- **Clusters**
- Grids
- Clouds
- Supercomputers

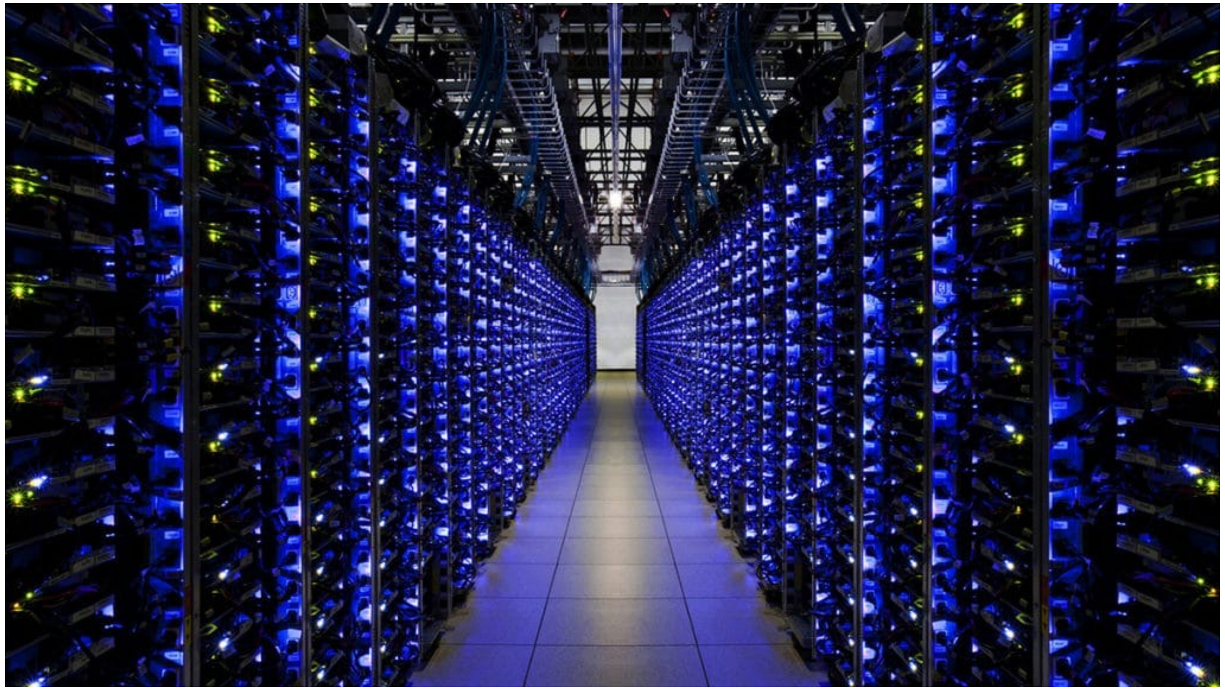


- Definition = tightly/loosely-coupled set of computers, in the same datacenter, each running the same OS, usually commodity hardware
- Examples: DAS-5

Large-scale Infrastructure

- Datacenters

- Clusters
- **Grids**
- Clouds
- Supercomputers

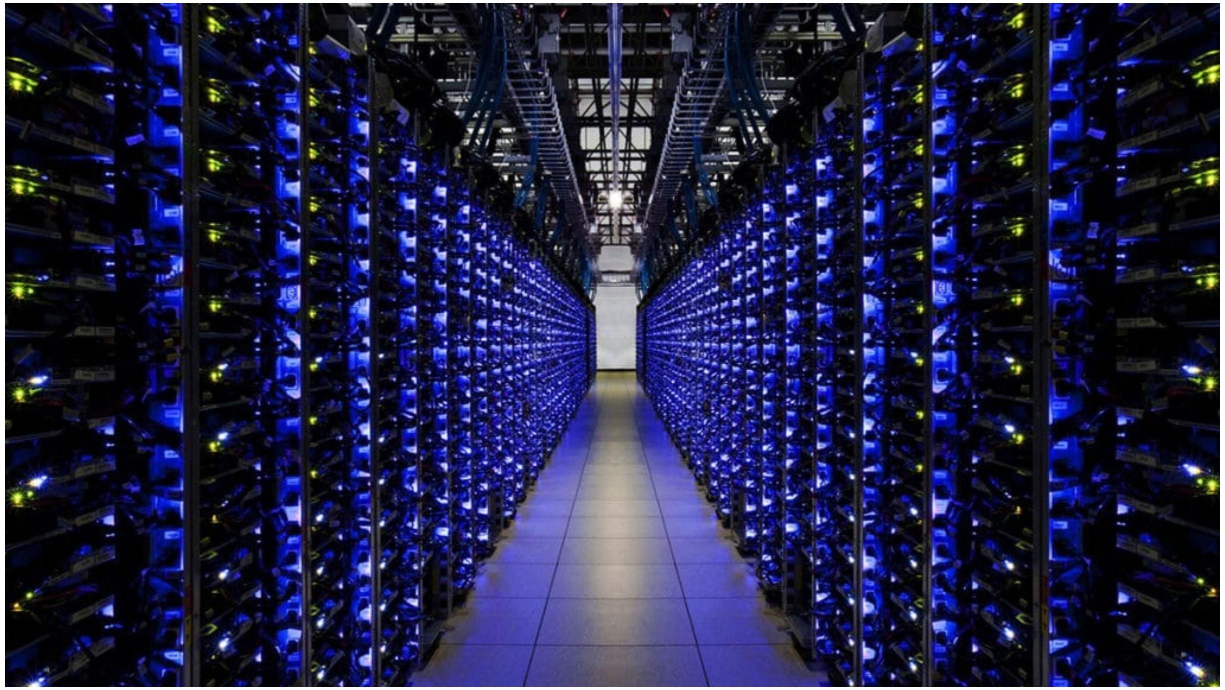


- Definition = loosely-coupled set of computers, geographically distributed, heterogeneous hardware & software, different administrative domains
- Examples: Grid 5000

Large-scale Infrastructure

- Datacenters

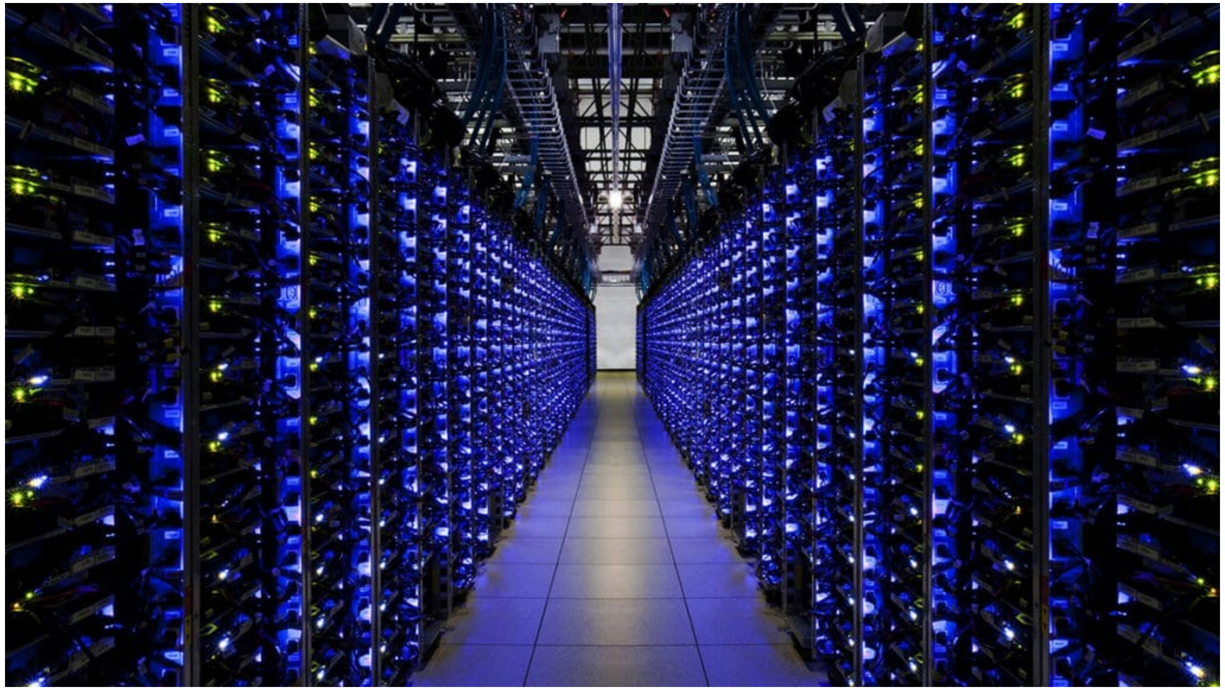
- Clusters
- Grids
- **Clouds**
- Supercomputers



- Definition = accessible pool of virtualized resources, which could be geographically distributed, heterogeneous, all the physical infrastructure is handled by a 3rd party, clients access virtual resoures and are given SLA
- Examples: Amazon, Azure, GCE

Large-scale Infrastructure

- Datacenters
- Clusters
- Grids
- Clouds
- **Supercomputers**



- Definition = highly tightly-coupled set of high-end machines, connected through high-speed networks, usually with the purpose of running HPC tasks
- Examples: Fugaku – Japan, Summit – US, TaihuLight – China (see top500.org)

Policy vs. Mechanism

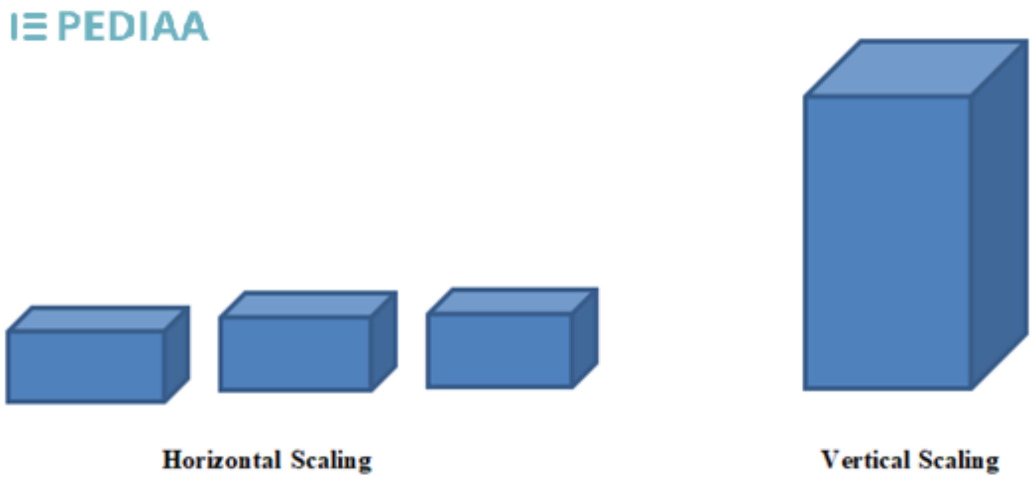
- Extremely important in computer systems!

- **Mechanism** = what to do, implementation-specific
- **Policy** = how and when to apply mechanism

- Example:
  - LRU cache replacement policy
  - CPU scheduling

Scalability

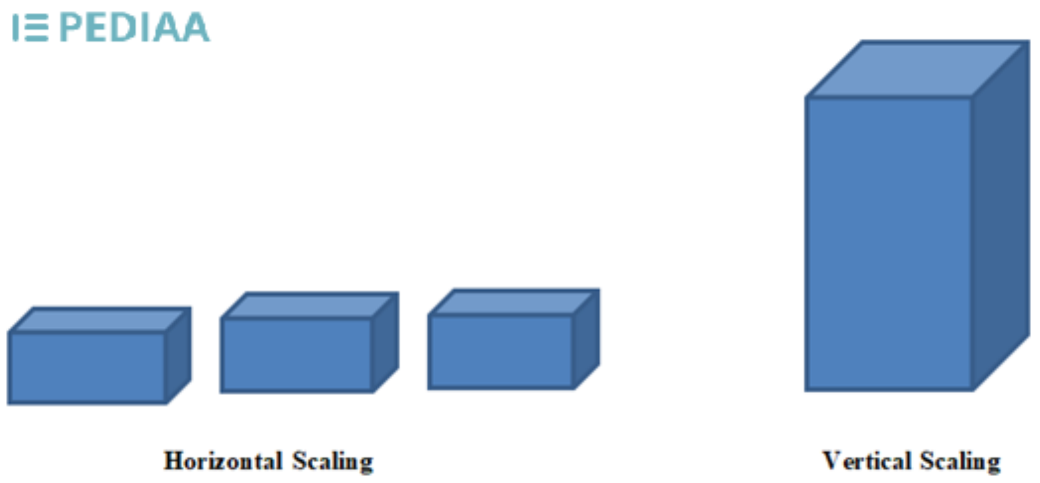
- **Vertical scalability** = adding more resources to the same set of machines
- **Horizontal scalability** = adding more machines to the pool



Scalability

- **Strong scalability** = adding more resources but keeping problem size constant
- **Weak scalability** = scaling problem size proportionally with the number of resources

- Combinations of weak/strong and horizontal/vertical scaling possible.



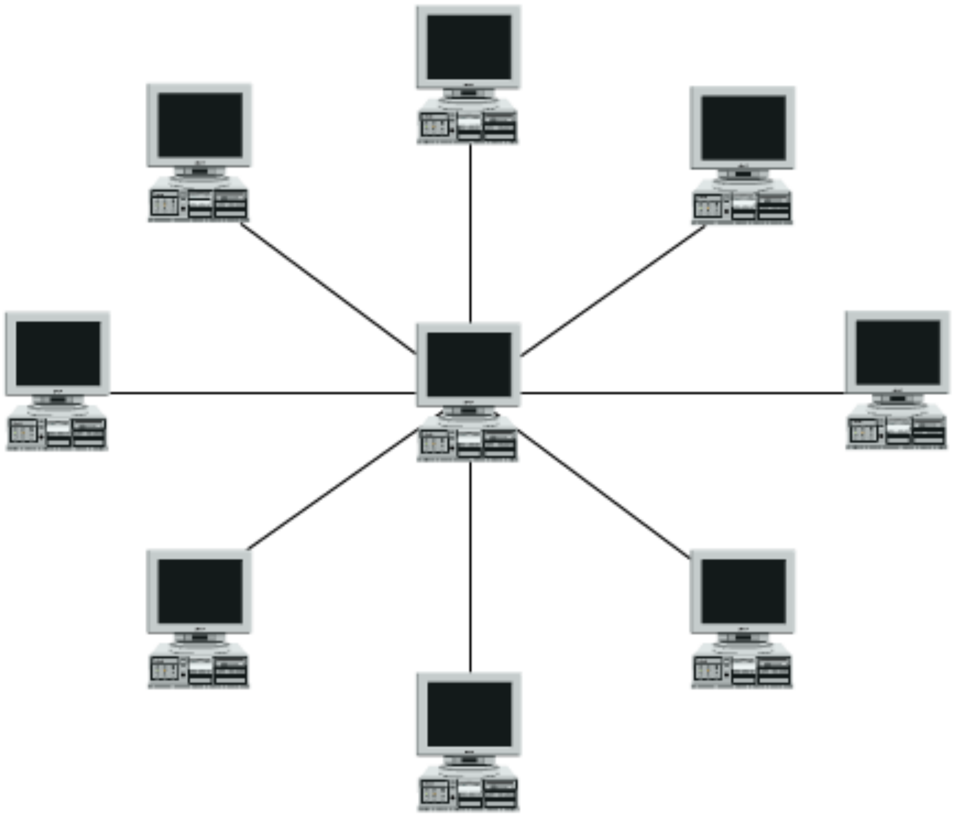
System Architectures

- Star (centralized)
- All-peers (decentralized)

How do the system components interconnect?

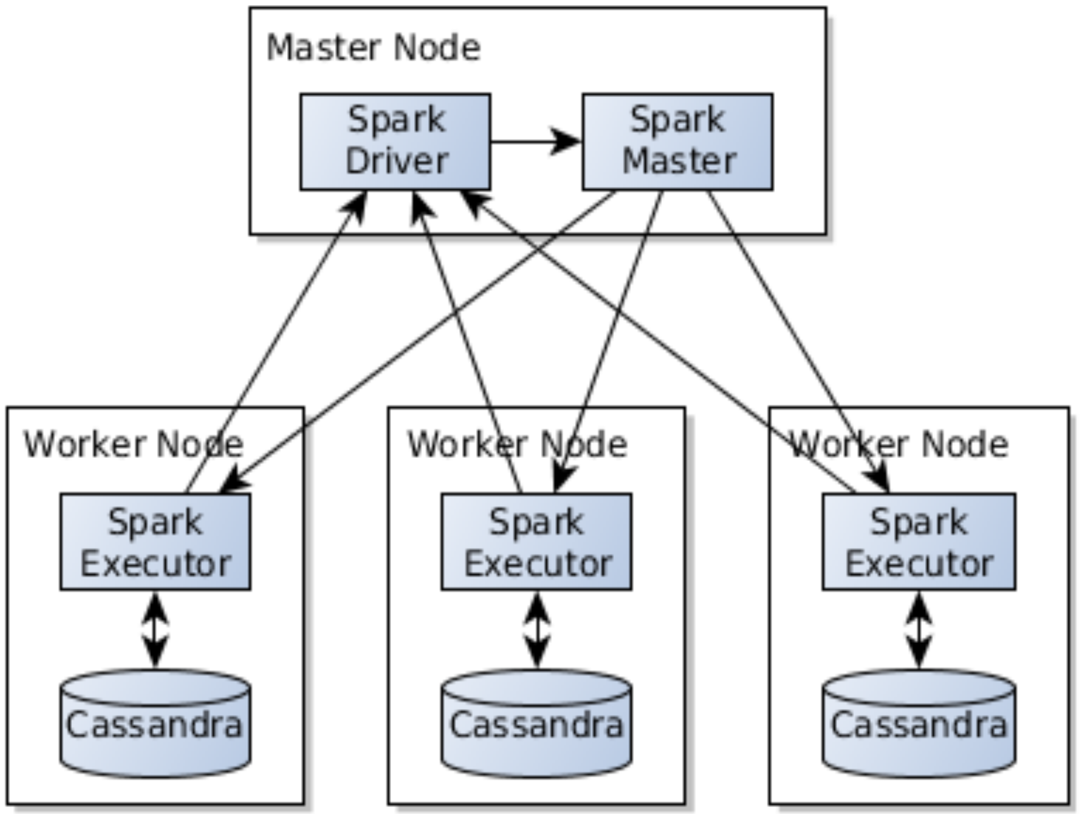
System Architectures - Centralized

- **Star (centralized)**
- Hierarchical
- Super-peers
- All-peers (decentralized)



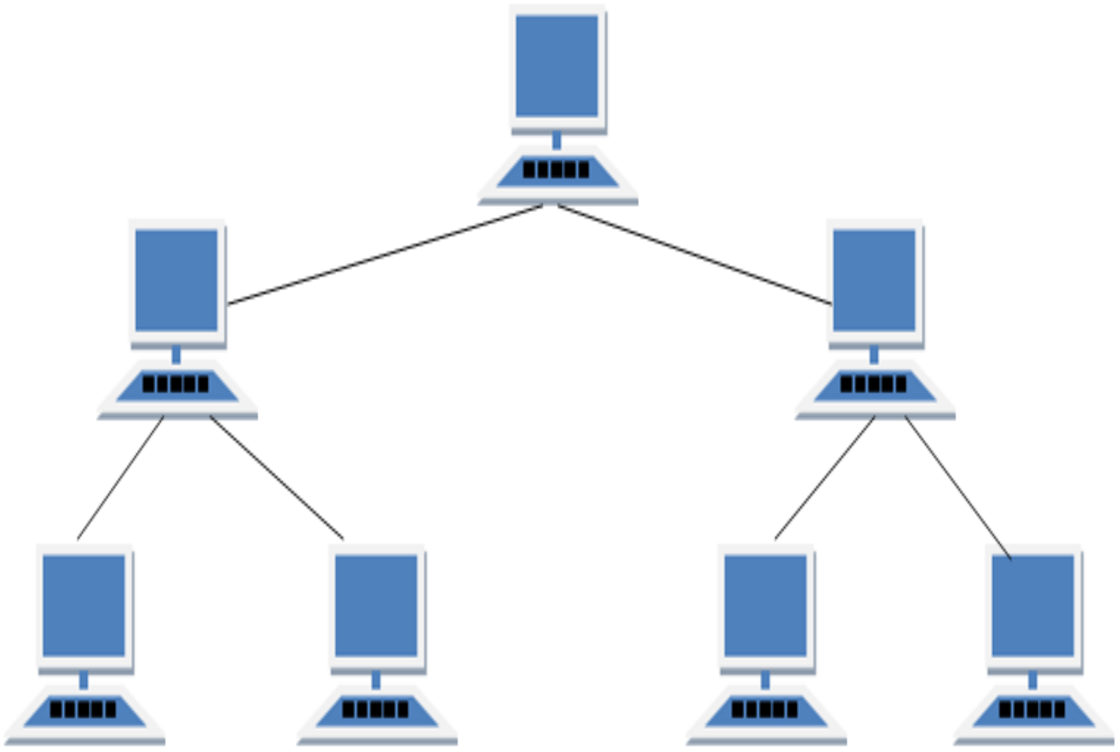
Centralized Example: **Spark**

- **Star (centralized)**
- Hierarchical
- Super-peers
- All-peers (decentralized)



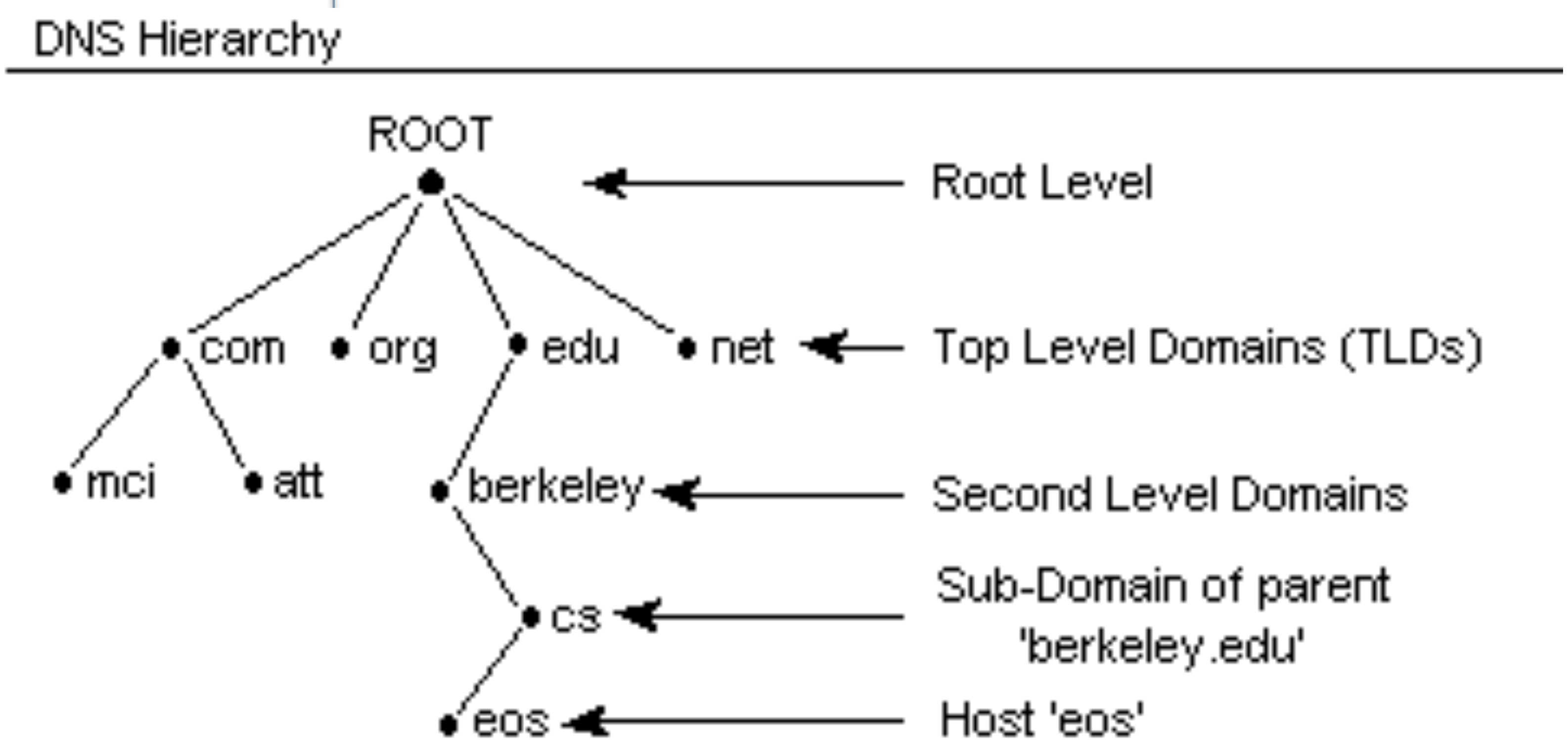
System Architectures - Hierarchical

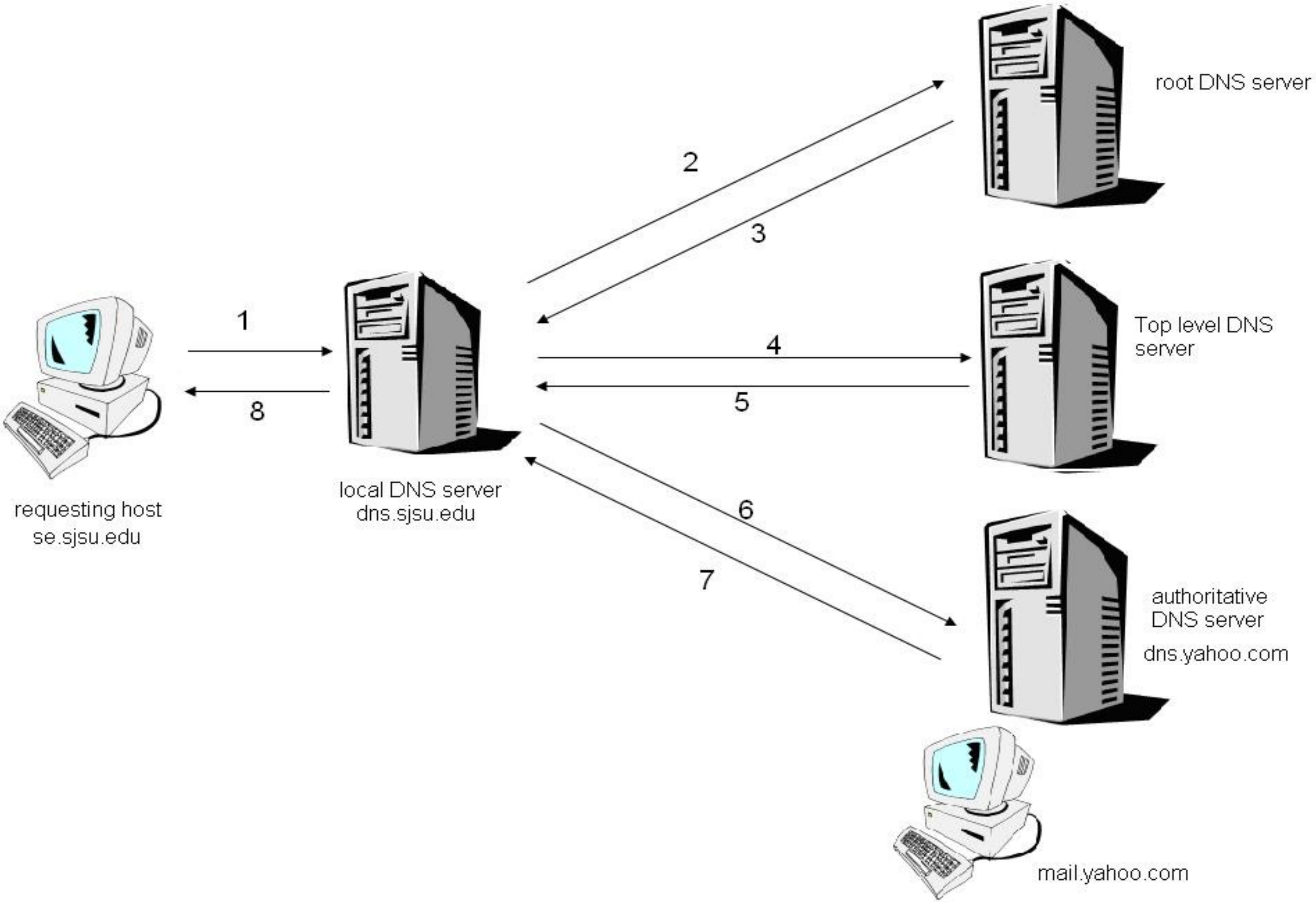
- Star (centralized)
- **Hierarchical**
- Super-peers
- All-peers (decentralized)



Hierarchical Example - DNS

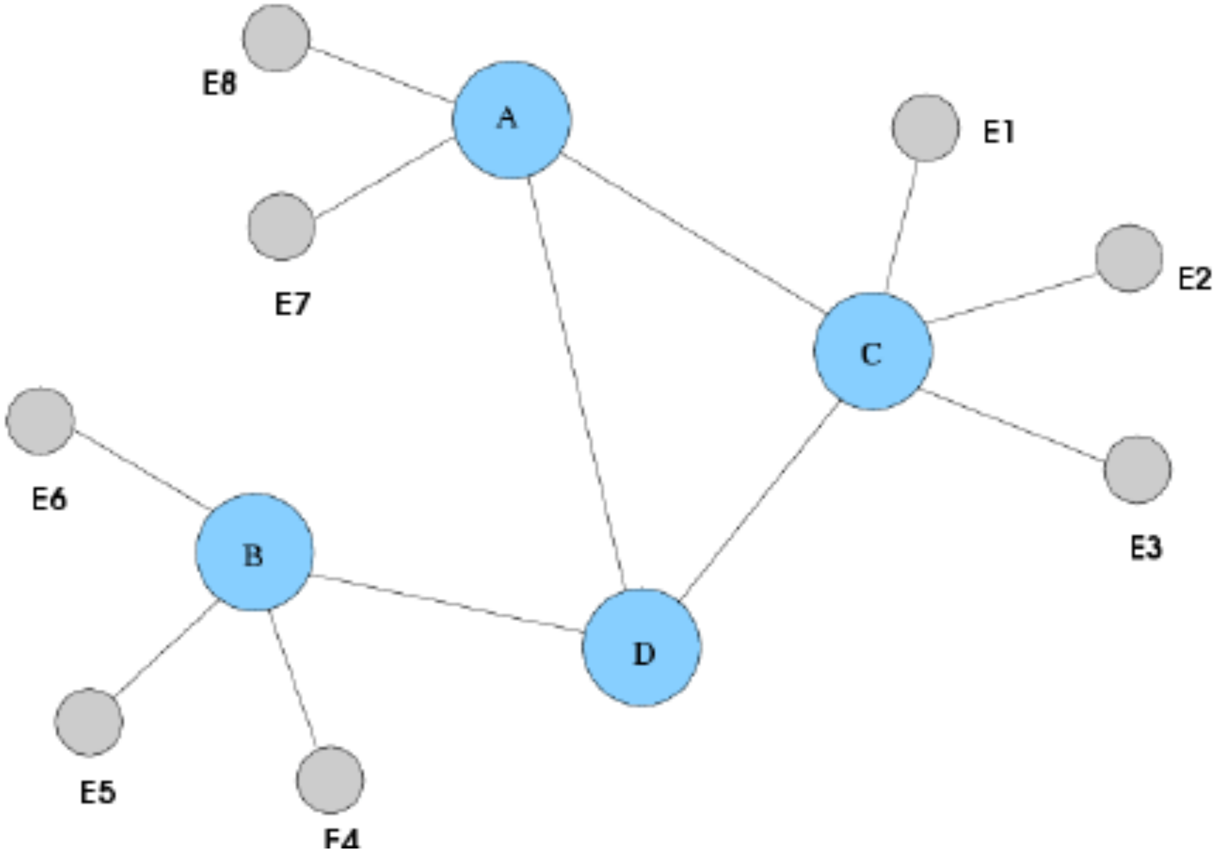
- Star (centralized)
- **Hierarchical**
- Super-peers
- All-peers (decentralized)




root DNS server
2
3
1
8
4
5
Top level DNS server
requesting host
se.sjsu.edu
local DNS server
dns.sjsu.edu
6
7
authoritative DNS server
dns.yahoo.com
mail.yahoo.com

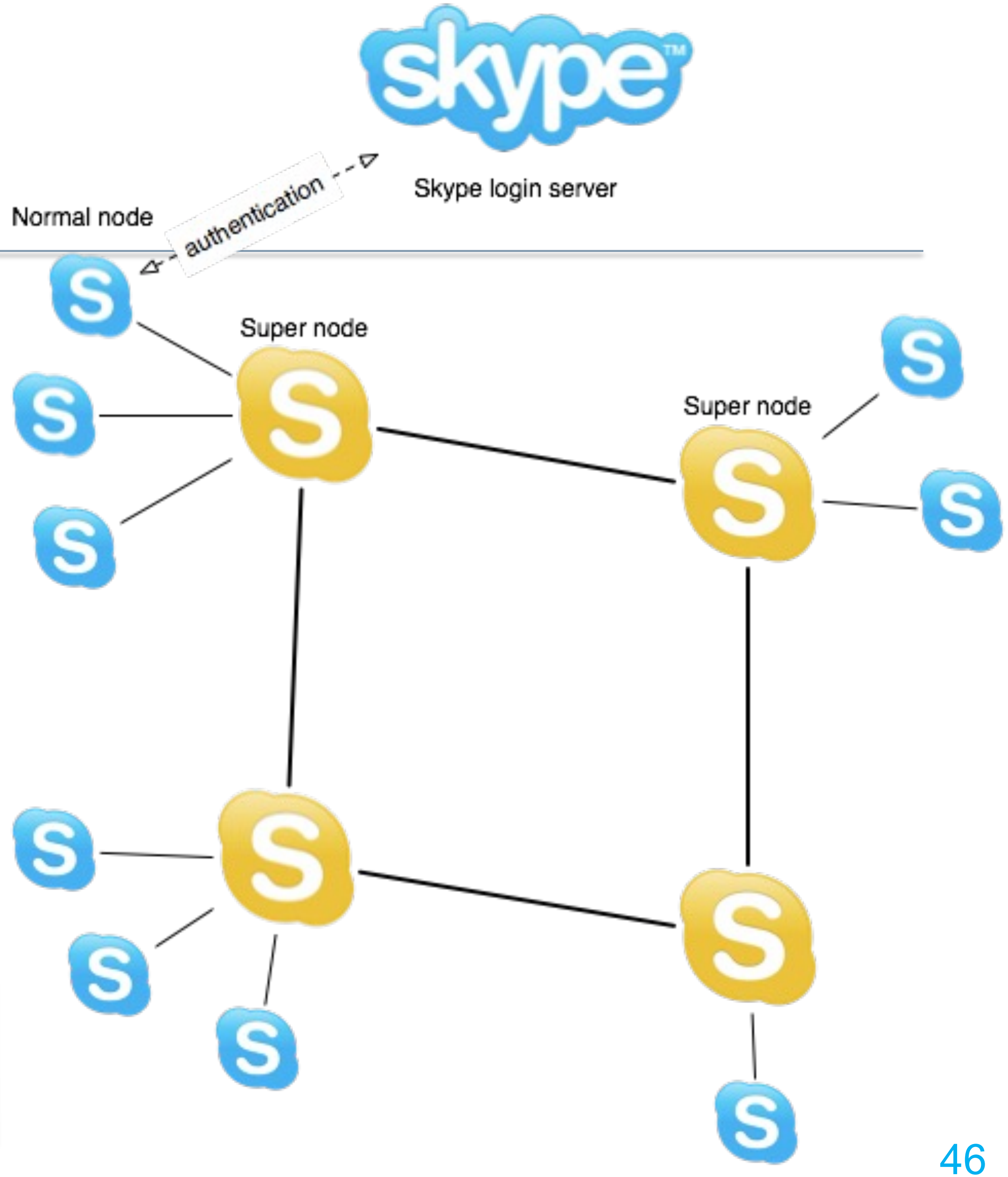
System Architectures

- Star (centralized)
- Hierarchical
- **Super-peers**
- All-peers (decentralized)



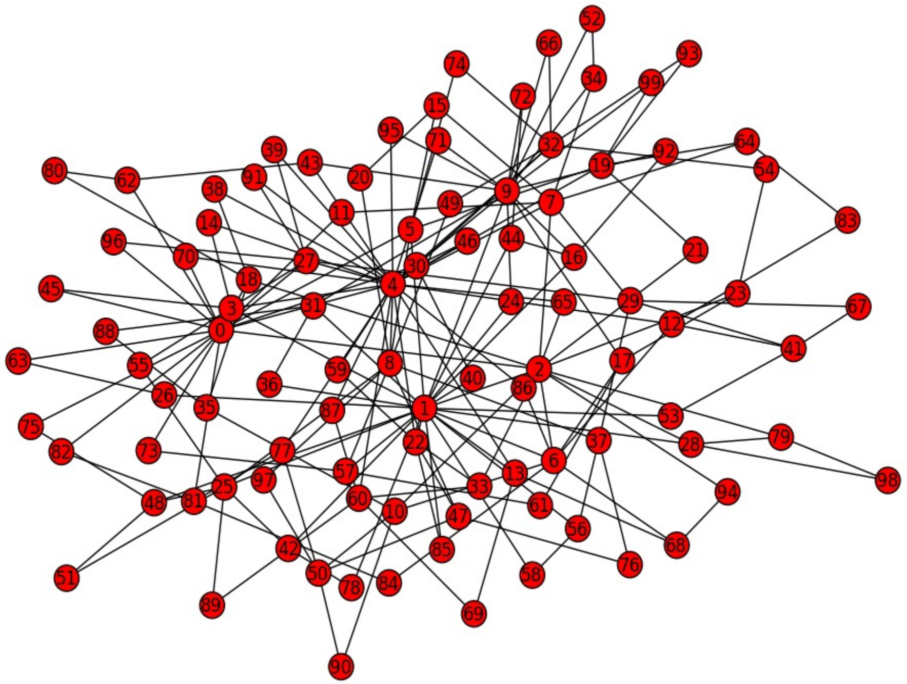
Super-peers Example

- Star (centralized)
- Hierarchical
- **Super-peers**
- All-peers (decentralized)



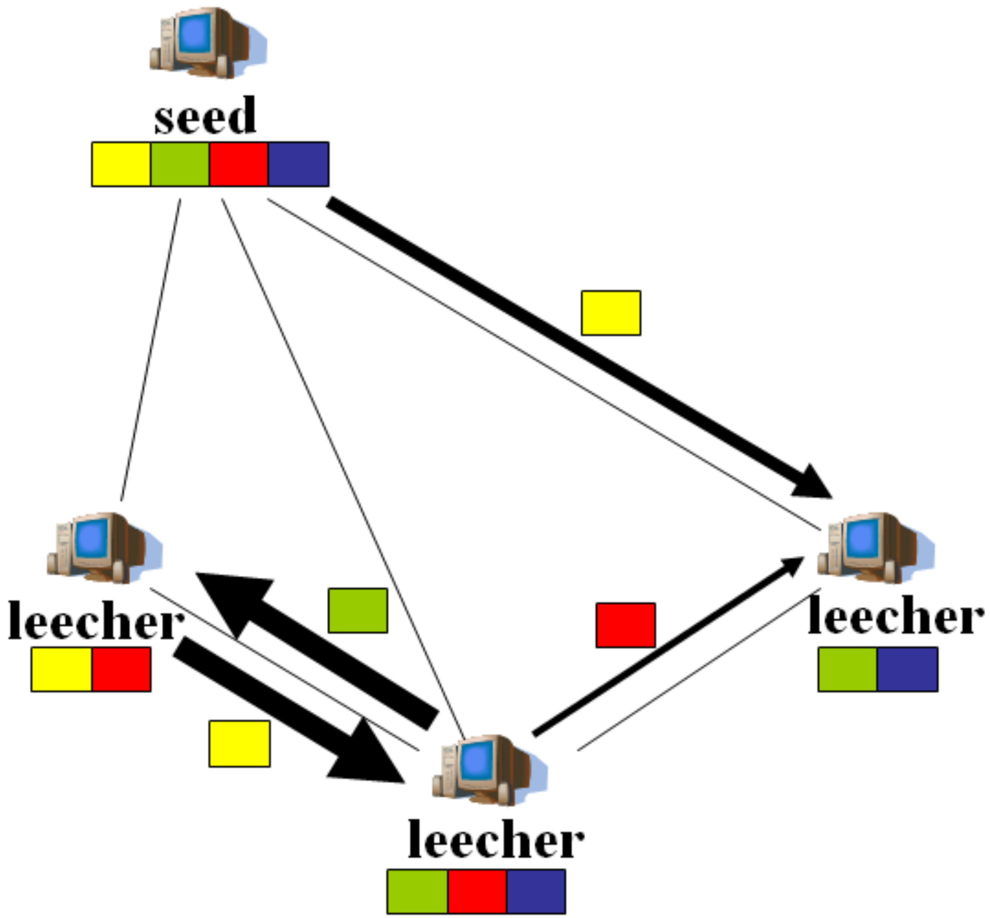
System Architectures

- Star (centralized)
- Hierarchical
- Super-peers
- **All-peers (decentralized)**



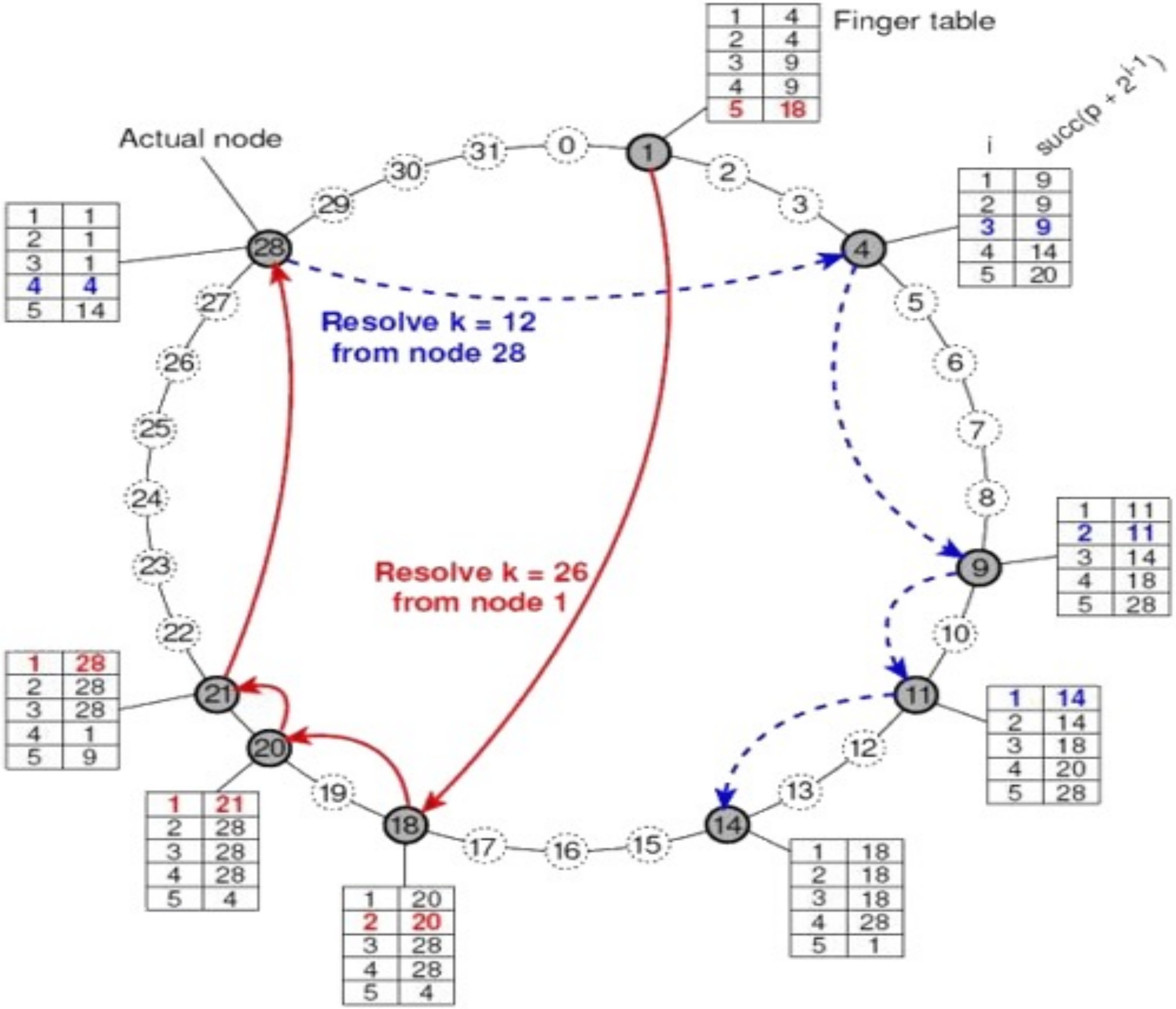
Decentralized Example - BitTorrent

- Star (centralized)
- Hierarchical
- Super-peers
- **All-peers (decentralized)**



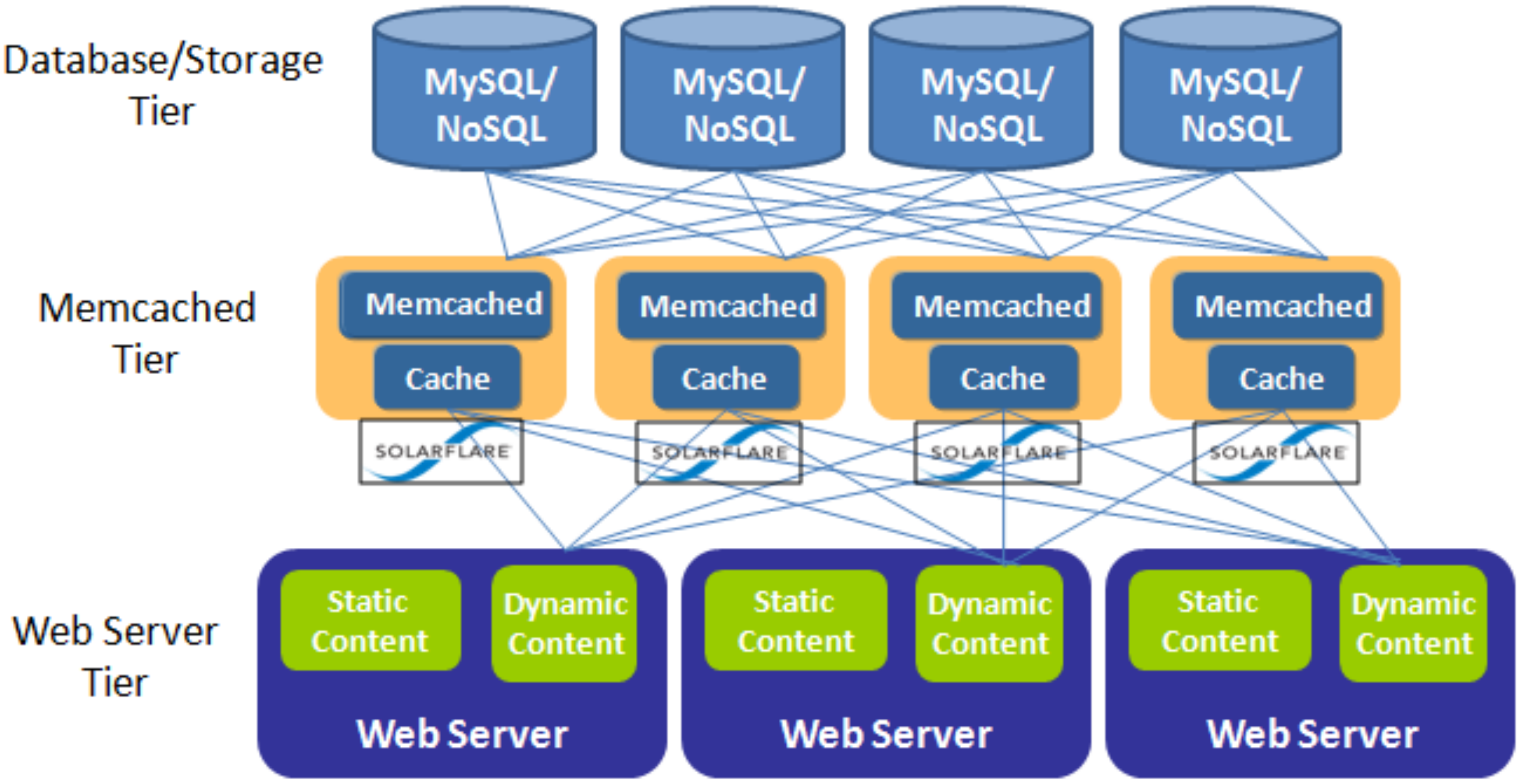
Decentralized Example – Chord DHT

- Star (centralized)
- Hierarchical
- Super-peers
- **All-peers (decentralized)**



Architecture Complexity - Layering

- Most production-ready systems are layered!

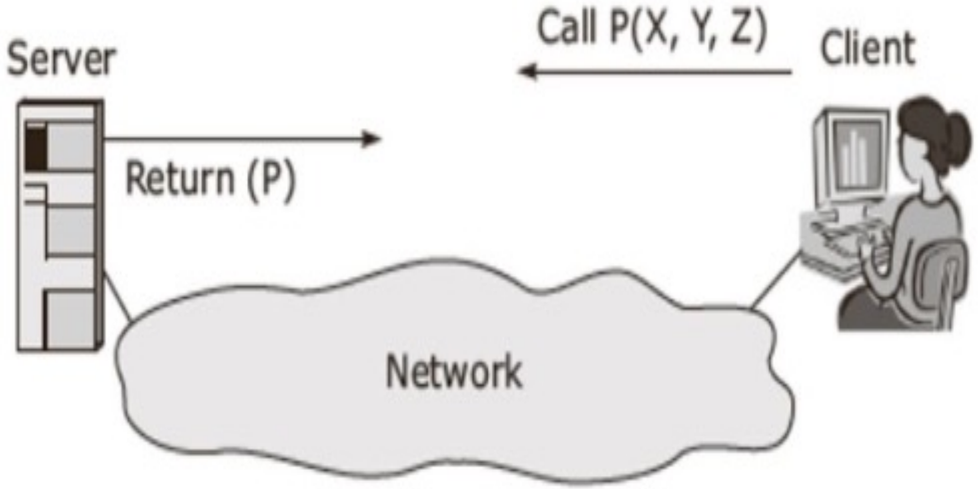


RPC – Remote Procedure call

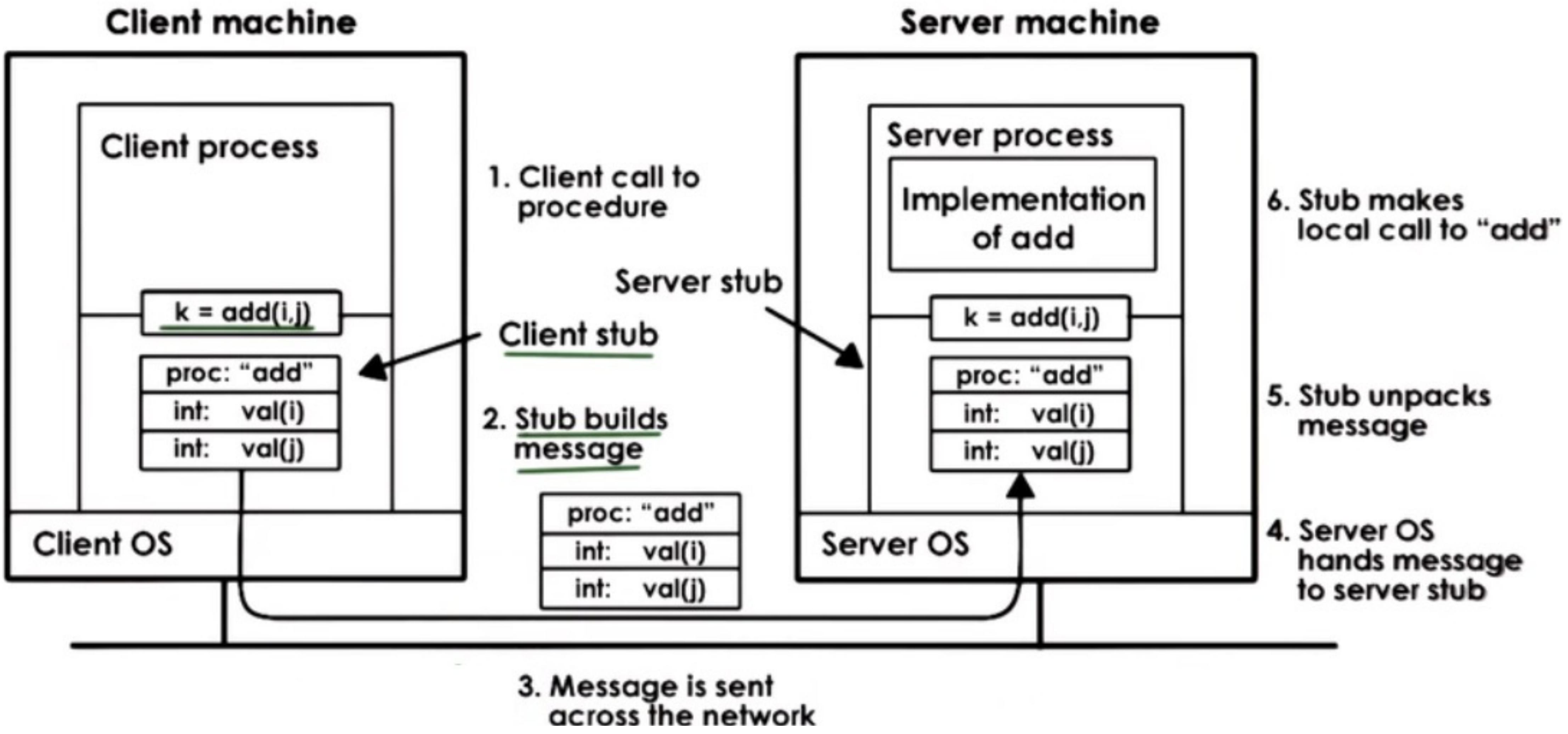
- Goal: emulate traditional mechanism of procedure calls

- Traditional issues:
- Efficient parameter passing
- References/pointers

- Distributed Systems issues:
- Client and server agree on function
- Marshalling parameters
- Failures



RPC – Remote Procedure Call (2)



IDL - Interface Description Language

- IDLs describe software component API
- Client calls using the interface
- Server implements the interface

- Used for:
- Function signatures
- Parameter types
- Data structures

Communication Techniques

Message-passing interface (MPI)

- Tightly-coupled infrastructure (clusters, supercomputers)

- **Queuing systems, message brokers**
  - Protocols: MQTT, AMQP
  - Implementations: ZeroMQ, RabbitMQ
  - Loosely-coupled infrastructure, middleware
- **Actor-model**
  - Progr. model for concurrent computing
  - Avoids locking, actors interact by messages
  - Tightly- and loosely-coupled infrastructure

Take-home message

- Course structure and homework assignments
- System architectures
- RPC and communication models
- For assignments, DAS-5 accounts!
- Brightspace or email for questions