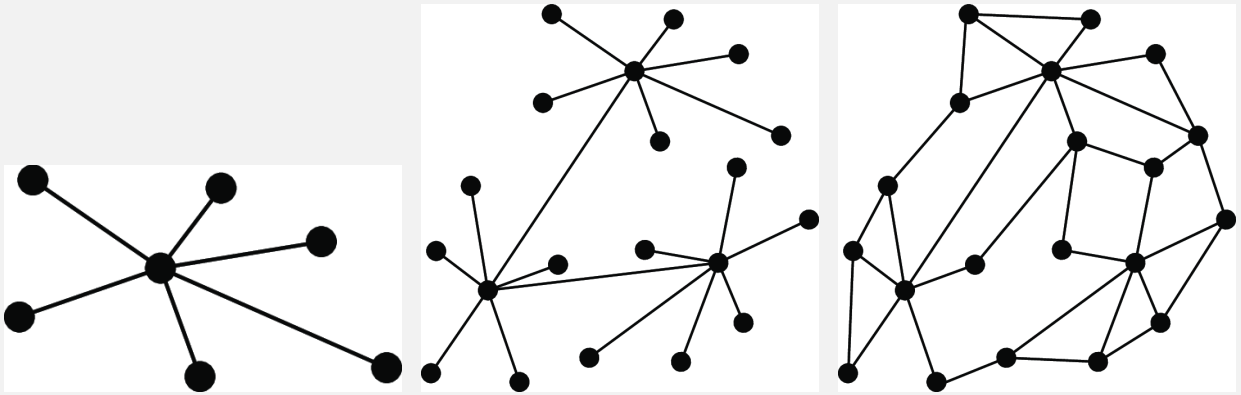
Chapter 1: Introduction

What is a Distributed System?

- *A distributed system is a collection of independent computers that appears to its users as a single coherent system.*

Centralized, Decentralized, Distributed Systems?

- Simplistic views of Centralized, Decentralized, Distributed systems.



Decentralized vs Distributed System

- **Integrative View**: There was a need to connect existing (networked) computer systems to each other.
- **Expansive View**: An existing computer system required an extension through additional computers.
- A decentralized system is a networked computer system in which processes and resources are **necessarily** spread across multiple computers.
- A distributed system is a networked computer system in which processes and resources are **sufficiently** spread across multiple computers.

Decentralized Examples

- **Federated Learning**: AI models need massive amounts of data to train models, which is typically done by bringing the data to high performance computer clusters. When the data needs to stay at an organization then the training needs to come to the data.
- **Blockchain (Distributed Ledger)**: In this case, the processes and resources are necessarily spread across multiple computers due to a lack of trust.
- **Geographically Dispersed**: Systems in which actual locations need to be monitored. A collection of satellites, traffic control, power plant, etc.

Distributed Examples

- **Gmail**: A coherent view provided to billions of users. The distributed system implementing the service expands (or shrinks) dependent upon the number of users. The system needs sufficient amount of distributed resources to handle the load.
- **Content Delivery Networks (CDNs)**: The contents of a website is copied and spread across various servers of the CDN. When visiting a website, a user is transparently redirected to a nearby CDN server that holds all or part of the content of that website. The content isn't copied to all servers but only to where it makes sense, that is sufficiently. For example, Akamai is a CDN that has around 365,000 servers across 135 countries.
- **Networked Attached Storage (NAS)**: A NAS is a computer server optimized for file storage and offers the ability to easily share those files. Users see the files as local files on their laptops or desktops. Where and how the files are stored is hidden. Thus NAS provides sufficient spreading of processes and resources.

In-class Exercise

- Simulate a centralized and a distributed algorithm for the same problem in class. The problem is to send a secret message to all students in the class. The entire class will participate together in this exercise!

- Does a decentralized approach make sense for this problem?
- Does a centralized approach make sense for some problems (even at scale)?

Major Perspectives on Distributed Systems

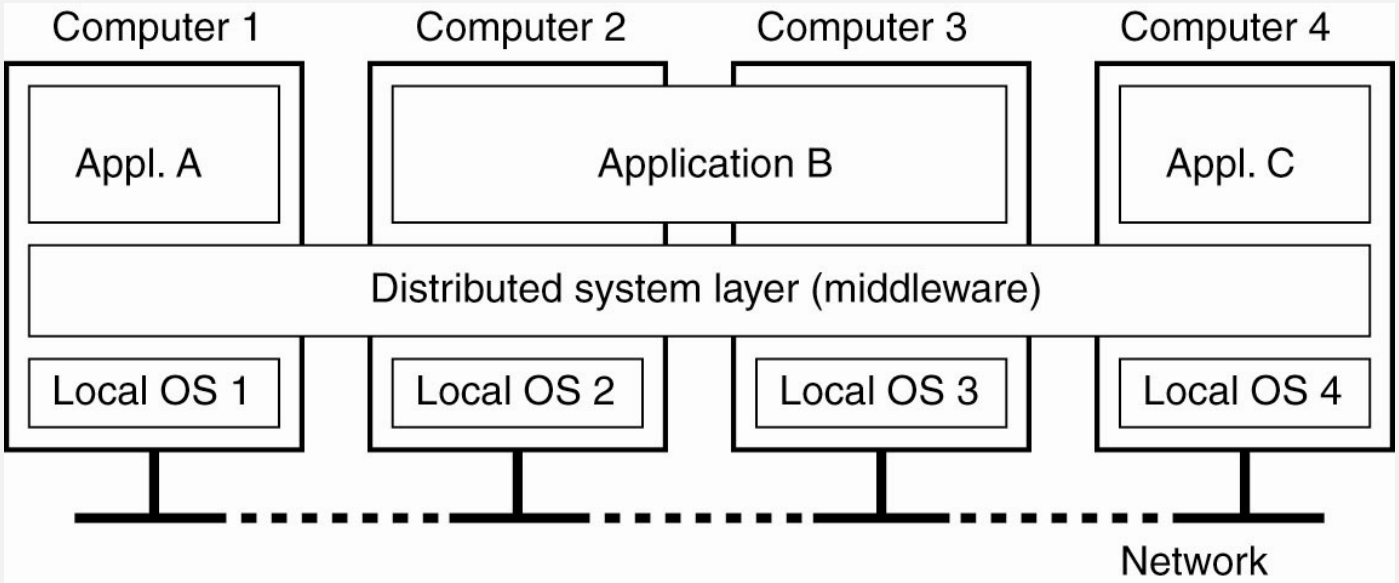
- Architectural
- Processes
- Communication
- Coordination
- Naming
- Consistency and replication
- Fault Tolerance
- Security

Characteristics of a Distributed System (1)

- Collection of autonomous computing elements
  - Computing elements (or nodes) can be hardware and/or software
  - No global clock
  - Group membership is difficult: Open groups versus Closed groups
  - Often organized as an overlay network: structured, unstructured, peer-to-peer

Characteristics of a Distributed System (2)

- Single coherent system
  - Requires transparency: process execution and data storage
  - Often implemented using middleware: a separate layer of software that is logically placed on top of the operating systems of the computers that are part of the distributed system.

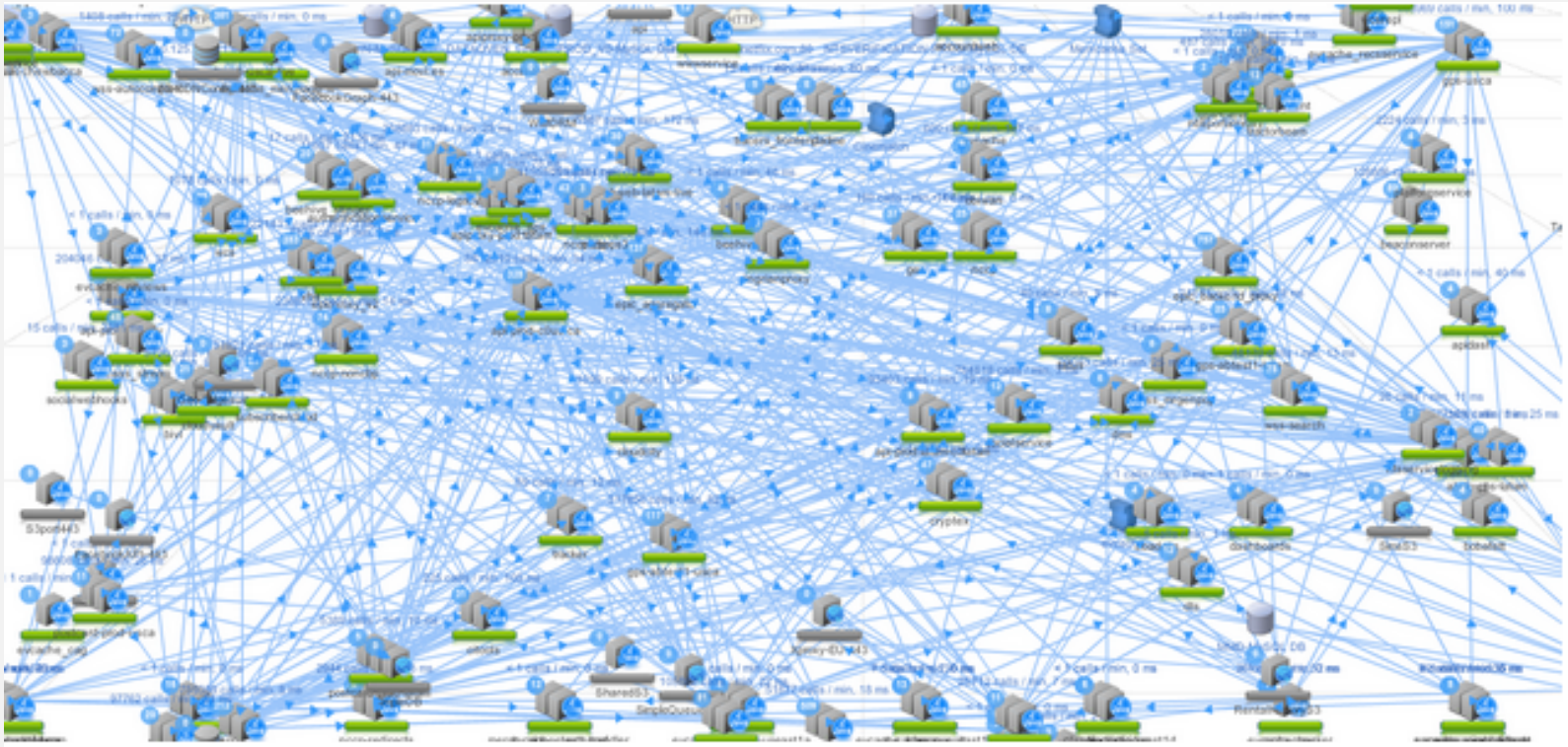


  - It handles resource management, interapplication communication, masking of and recovery from failures, security and accounting services.

Examples (1)

The Internet is just a world passing around notes in a classroom. –Jon Stewart

Examples (1)

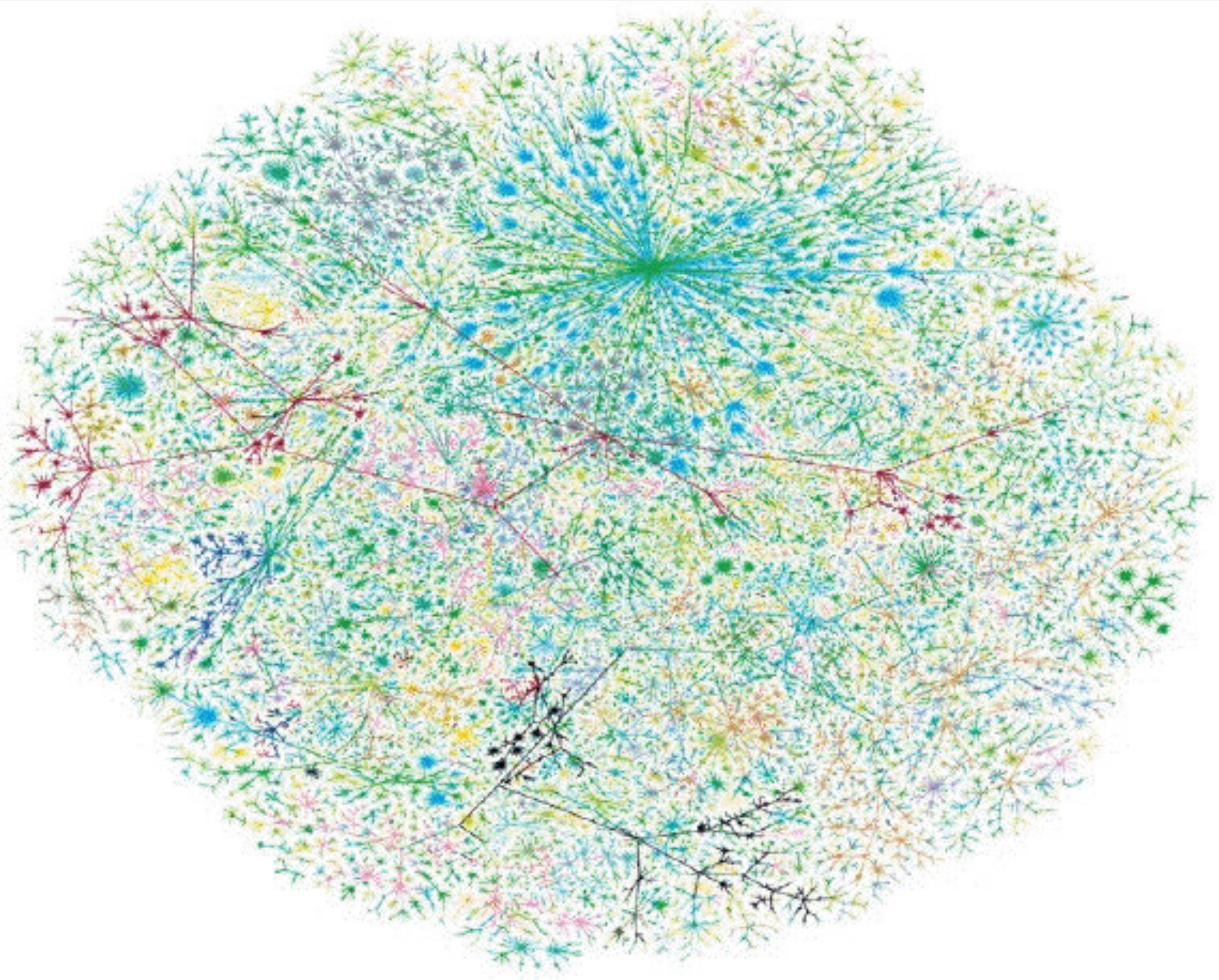


The Internet is just a world passing around notes in a classroom. –Jon Stewart

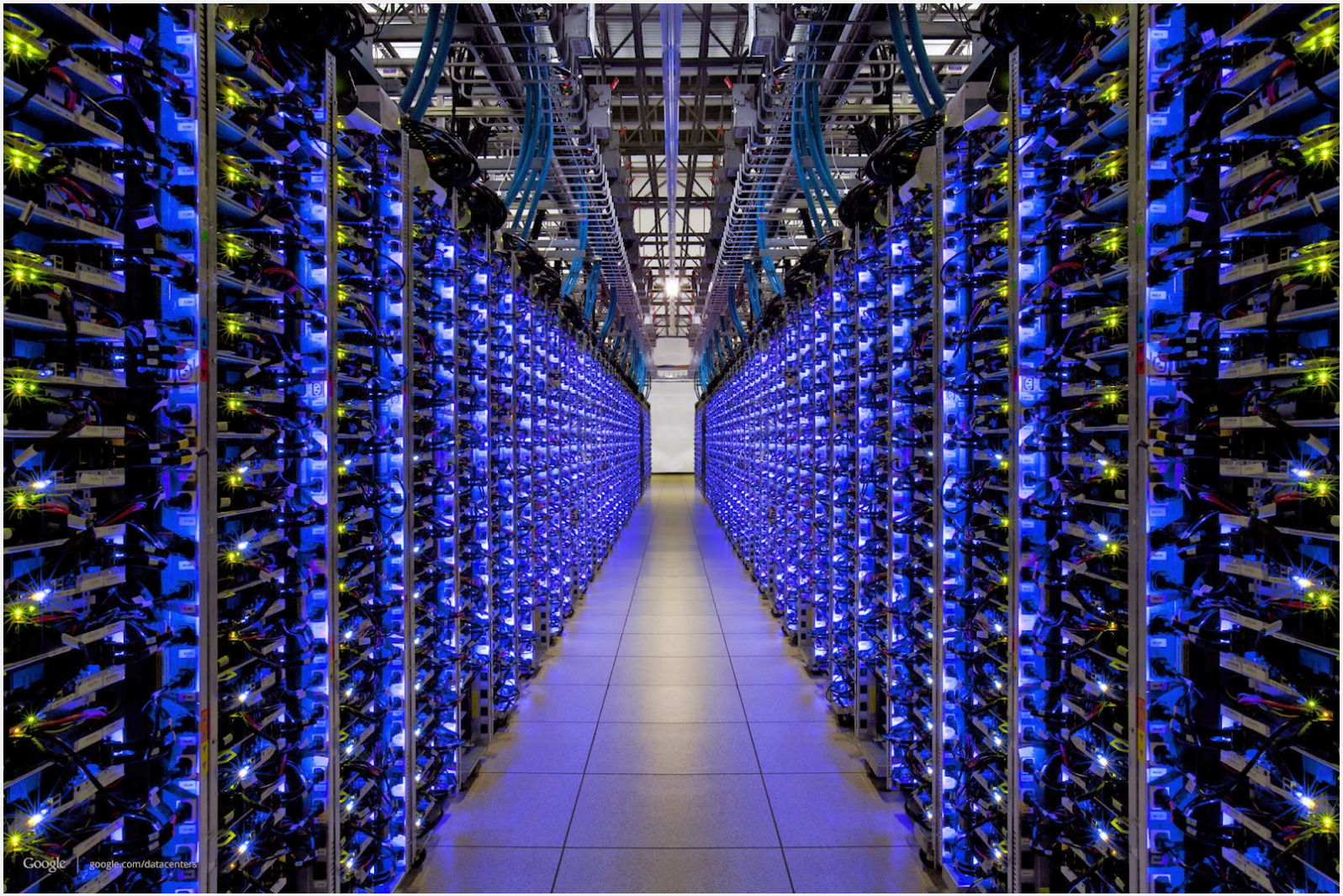
Examples (2)

Google Search I'm Feeling Lucky

Examples (2)



Examples (2)



Examples (3)

- Google search, GMail, Google Docs, et al
- Amazon
- Cloud storage, such as Dropbox et al
- Facebook, Twitter, EBay
- LinkedIn, WhatsApp, Instagram, Snapchat, TikTok
- Uber, Lyft
- Hadoop, Hive, Zookeeper, et al
- No-SQL Databases like HBase, Cassandra, MongoDB, et al
- Distributed file systems: NFS (Network File System), Google File System (GFS), Hadoop Distributed File System (HDFS), Amazon S3, Lustre, GlusterFS, Parallel Virtual File System (PVFS)
- P2P: Peer to Peer: Bit Torrent, Gnutella
- Domain Name System (DNS)
- Git: distributed version control system.
- SETI: a distributed computing project in which volunteers donate idle computer power to analyze radio signals for signs of extraterrestrial intelligence.
- Bitcoin: decentralized digital currency (and other blockchain applications)
- Virtually every substantial website!

Goals of a Distributed System?

- Makes resources accessible
- Reasonably hides the fact that resources are distributed across a network (*Transparency*)
- Open
- Scalability
- Dependability
- Security

Making Resources Accessible

- Benefits
  - Better economics by sharing expensive resources
  - Easier to collaborate and exchange information
  - Create virtual organizations where geographically dispersed people can work together using groupware
  - Enables electronic commerce
- Problems
  - Eavesdropping or intrusion on communication
  - Tracking of communication to build a profile

Transparency

Type	**Description**
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Relocation	Hide that a resource may move to another location while in use
Migration	Hide that a resource may move to another location
Replication	Hide that a resource has multiple copies
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Degree of Transparency

Completely hiding the distribution aspects from users is not always a good idea in a distributed system.

- Attempting to mask a server failure before trying another one may slow down the system
- Expecting several replicas to be always consistent could degrade performance unacceptably
- For mobile and embedded devices, it may be better to expose distribution rather than trying to hide it
- Signal transmission is limited by the speed of light as well as the speed of intermediate switches

Openness

An open distributed system offers services according to standard rules that describe the syntax and semantics of those services.

- Use of standard protocols.
- Services are described via interfaces, which are often describe via an Interface Definition Language (IDL). Interfaces only specify syntax so semantics is left to the ambiguities of natural language.
- Interoperability, Portability, Extensibility.
- Separating policy from mechanism. For example: *caching* in a web browser.

Dependability (1)

- Availability: A system is **available** if it is operational and accessible when required for use.
- Reliability: A system is **reliable** if it continues to operate correctly over time.
- *Availability vs Reliability*: If a system is available, it can be used. If a system is reliable, it can be used with confidence. **Example**: If a system goes down on average for one random millisecond every hour, then it is available for 99.999% of the time. However, if the system goes down for one random millisecond every hour, then it is not reliable.
- Safety: A system is **safe** if no catastrophic consequences occur to the system itself, to its users, or to itself when it temporarily fails to operate correctly.
- **Common Metrics**: Mean Time To Failure (MTTF), Mean Time To Repair (MTTR), Mean Time Between Failures (MTBF) = MTTF + MTTR, Availability = MTTF / (MTTF + MTTR)

Dependability (2)

- Maintainability: A system is **maintainable** if it can be repaired and enhanced effectively and efficiently.
- Failure $\leftarrow$ Error $\leftarrow$ Fault
- Faults can be **Transient**, **Intermittent**, or **Permanent**.
  - A transient fault is a temporary fault that disappears after some time.
  - An intermittent fault is a fault that occurs repeatedly and unpredictably.
  - A permanent fault is a fault that does not disappear and requires repair.
- Confidentiality is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes.
- Integrity is the property that data has not been altered in an unauthorized manner.

Security

- Confidentiality
- Integrity
- Encryption is a key technique. Symmetric versus public-key techniques.

Scalability

Scalability can be measured against three dimensions.

- Size: be able to easily add more users and resources to a system
- Geographical: be able to handle users and resources that are far apart
- Administrative: be able to manage even if it spans independent administrative organizations

Centralized versus distributed implementations.

Centralized Solutions with Scalability Problems

- Centralized services.
- Centralized data.
- Centralized algorithms.

Distributed Approach

Characteristics of decentralized algorithms:

- No machine has complete information about the system state.
- Machines make decisions based only on local information.
- Failure of one machine does not ruin the algorithm.
- There is no implicit assumption that a global clock exists.

Scaling Techniques

- Hiding communication latency: Examples would be asynchronous communication as well as pushing code down to clients (E.g. Javascript)
- Distribution: Taking a component, splitting into smaller parts, and subsequently spreading them across the system. (E.g. Domain Name System)
- Replication: Replicating components increases availability, helps balance the load leading to better performance, helps hide latency for geographically distributed systems. Caching is a special form of replication.

Distributed System Development Pitfalls

False assumptions made by first time developer *(formulated by Peter Deustch)*.

- 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

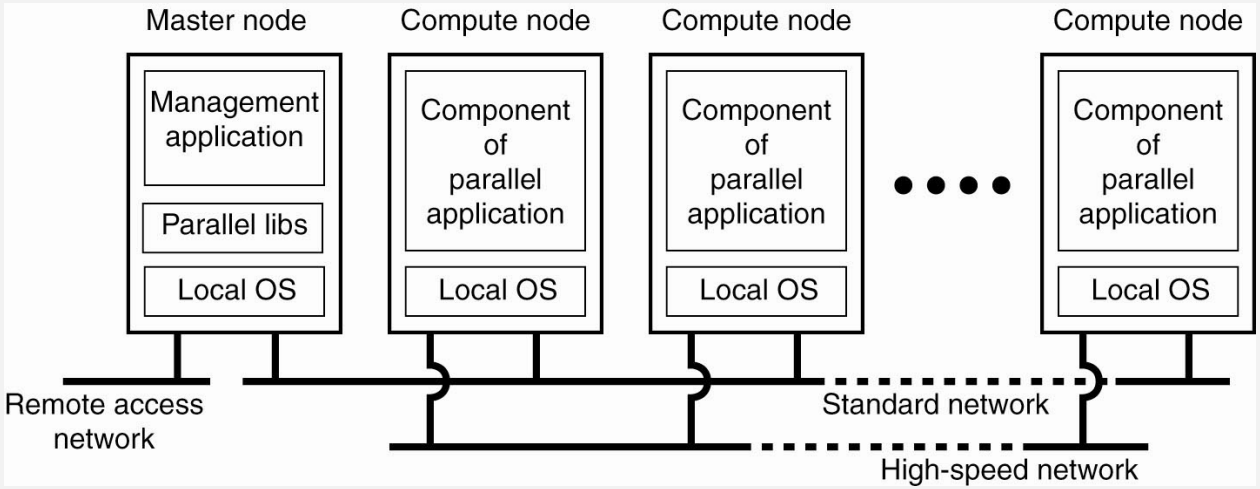
"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."
–Leslie Lamport

Types of Distributed Systems

- *Distributed Computing Systems*
  - Cluster Computing Systems
  - Grid Computing Systems
  - Cloud Computing
- *Distributed Information Systems*
  - Transaction Processing Systems
  - Enterprise Application Integration
- *Distributed Pervasive Systems*
  - Ubiquitous Computing Systems
  - Mobile Computing Systems
  - Sensor Networks

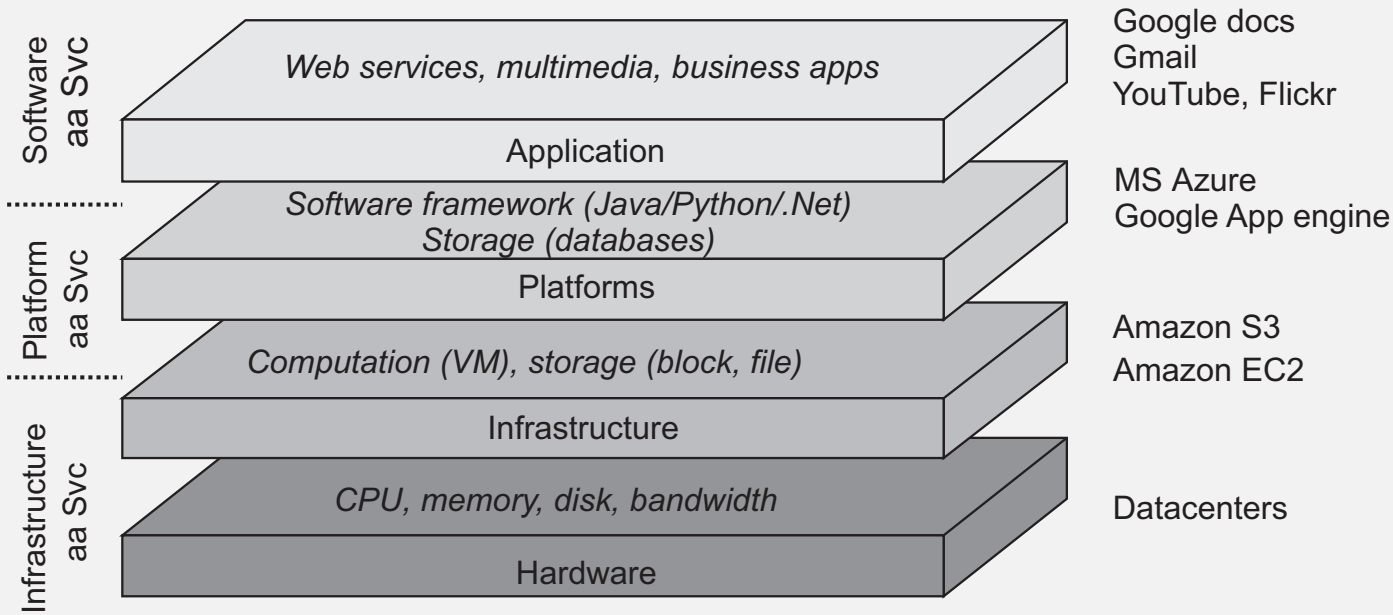
Cluster Computing

- A computer cluster is primarily used to run a single program in parallel on multiple machines. The program relies on parallel libraries and frameworks such as MPI (Message Passing Interface), MapReduce and others.
- A *computer cluster* consists of a collection of compute and I/O nodes that are controlled and accessed from a single master node. The master allocates nodes to a parallel program, maintains a queue of submitted jobs and provides an interface to manage the cluster.
- A cluster usually has a dedicated high-speed interconnection network and an optional administrative network.



Cloud Computing

- Cloud Computing provides the facilities to dynamically construct an infrastructure and compose what is needed from available services.



Layers of Cloud Computing

- Hardware: Processors, routers, power and cooling systems. Customers normally never get to see these.
- Infrastructure: Deploys virtualization techniques. Evolves around allocating and managing virtual storage devices and virtual servers.
- Platform: Provides higher-level abstractions for storage and such. Example: Amazon S3 storage system offers an API for (locally created) files to be organized and stored in so-called buckets.
- Application: Actual applications, whether distributed or not. Includes apps such as office suites (text processors, spreadsheet applications, presentation applications). Comparable to the suite of apps shipped with traditional OSes.

Integrating Applications

- **Situation**: Organizations confronted with many networked applications, but achieving interoperability was painful.
- **Basic approach**: A networked application is one that runs on a server making its services available to remote clients. Simple integration: clients combine requests for (different) applications; send that off; collect responses, and present a coherent result to the user.
- **Next step**: Allow direct application-to-application communication, leading to Enterprise Application Integration (EAI).

Example EAI: Transaction Processing Systems (1)

Transaction primitives:

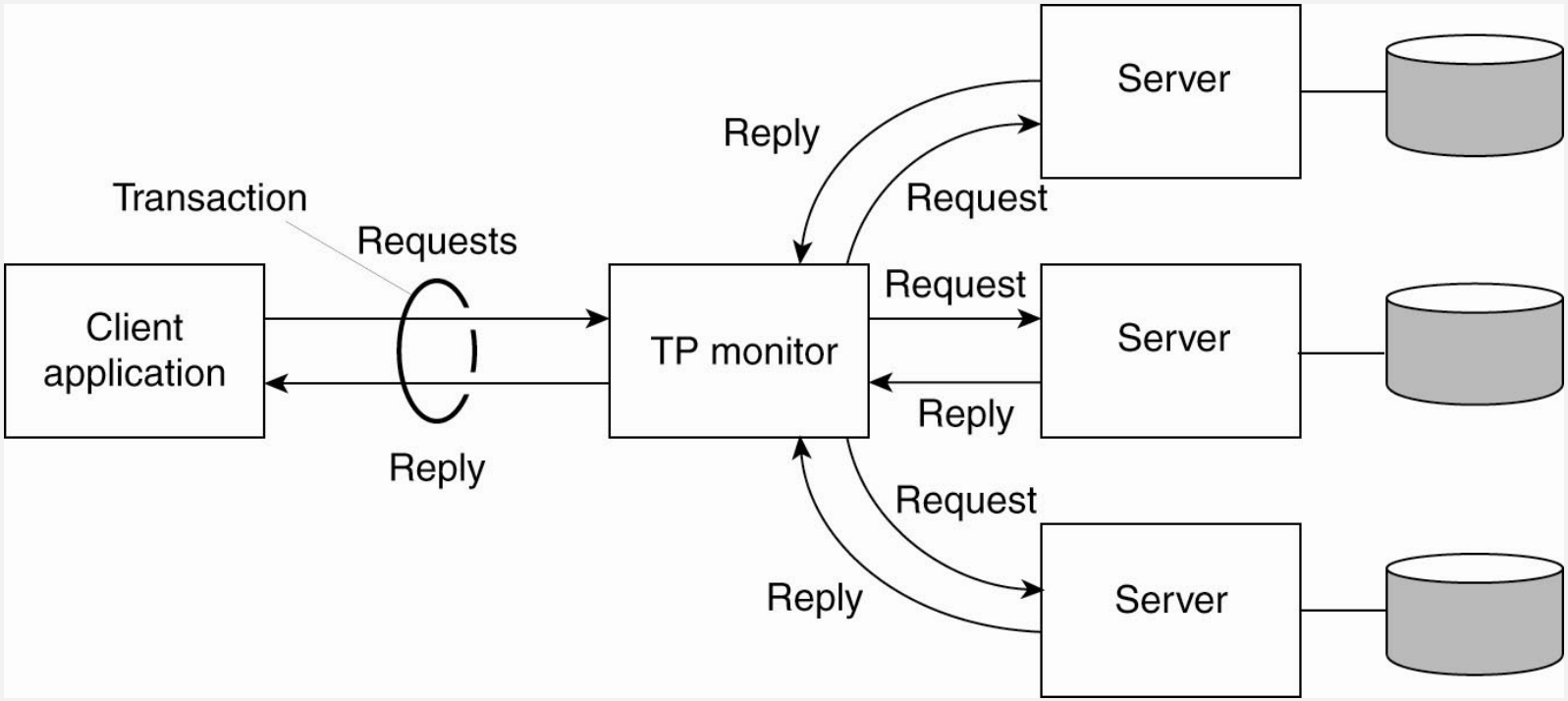
- BEGIN_TRANSACTION
- END_TRANSACTION
- ABORT_TRANSACTION
- READ, WRITE

ACID characteristic properties of transactions:

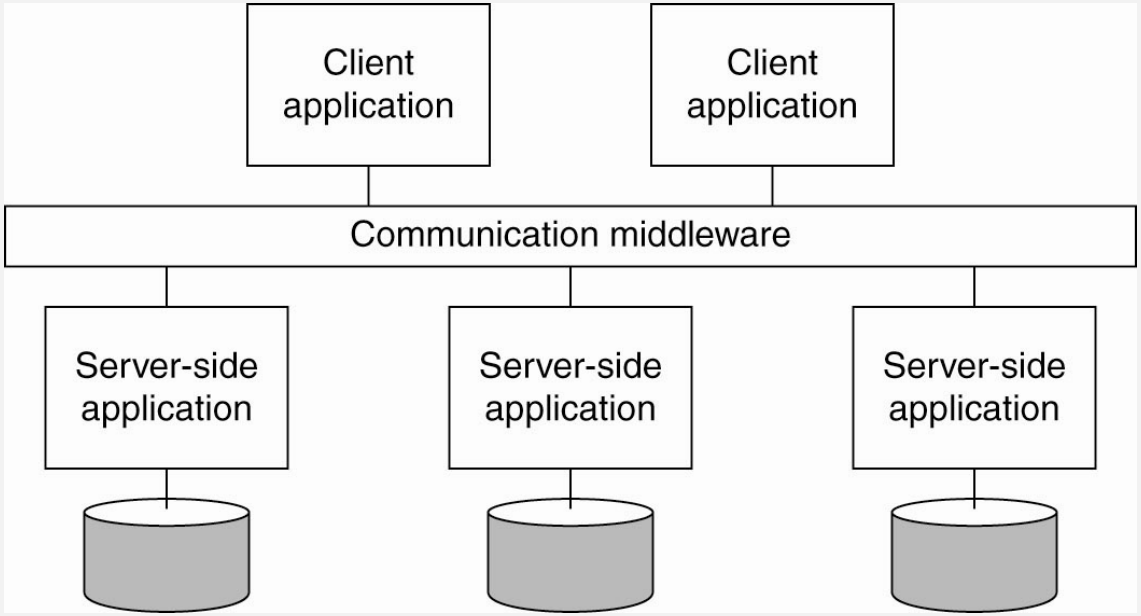
- Atomic: To the outside world, the transaction happens indivisibly
- Consistent: The transaction does not violate system invariants
- Isolated: Concurrent transactions do not interfere with each other
- Durable: Once a transaction commits, the changes are permanent

Transactions can be nested. Durability applies to top-level transactions only. For example: an airline and a hotel database. This led to the design of Transaction Processing Monitors (TP Monitors) to coordinate the commitment of multiple subtransactions.

Example EAI: Transaction Processing Systems (2)



Enterprise Application Integration



Middleware as a communication facilitator for enterprise application integration. This can be done in multiple ways. For example:

- Remote Procedure Call (RPC): Requests are sent through local procedure call, packaged as message, processed, responded through message, and result returned as return from call.
- Message Oriented Middleware (MOM): Messages are sent to logical contact point (published), and forwarded to subscribed applications.

Distributed Pervasive Systems

Characteristics of pervasive systems (aka Internet Of Things):

- Blends into the environment
- Encourage ad hoc composition
- Naturally distributed
- Nodes are often small and battery powered
- Wireless/mobile communication is the norm

Examples: Ubiquitous Computing Systems, Mobile Computing Systems, Sensor (and actuator) networks

In-class Exercise

Classify the distributed system examples we have seen so far into the three categories: Distributed Computing Systems, Distributed Information Systems, Distributed Pervasive Systems.

Chapter 1: Recommended Exercises

Attempt to answer the following questions on your own (as if you are in an interview). Then ask your favorite AI virtual assistant the same questions – explore the answers by interacting with it. The idea is to deepen your understanding of distributed systems by comparing your answers with those provided by the AI assistant.

- **Problem 1**. Explain the difference between centralized, decentralized, and distributed systems with real-world examples.
- **Problem 2**. What is the role of middleware in a distributed system?
- **Problem 3**. Explain what is meant by transparency, and give examples of different types of transparency.
- **Problem 4**. Scalability in a distributed system can be achieved by applying different techniques. What are these techniques?
- **Problem 5**. Give further examples of distributed pervasive systems.