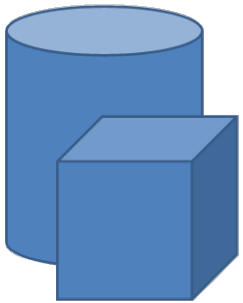
Distributed computing

Definition

Distributed computing is a field of computer science that studies distributed systems. A ***distributed system*** is a system whose components are located on different networked computers, which communicate and coordinate their actions by **passing messages** to one another. The components interact with one another in order to achieve a common goal. Three significant characteristics of distributed systems are:

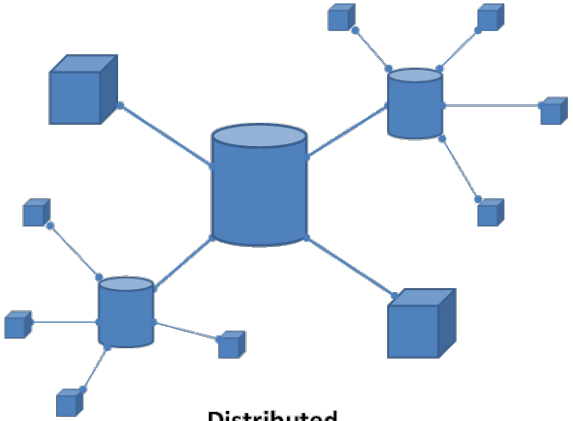
1. concurrency of components,
2. lack of a global clock, and
3. independent failure of components.

source: https://en.wikipedia.org/wiki/Distributed_computing



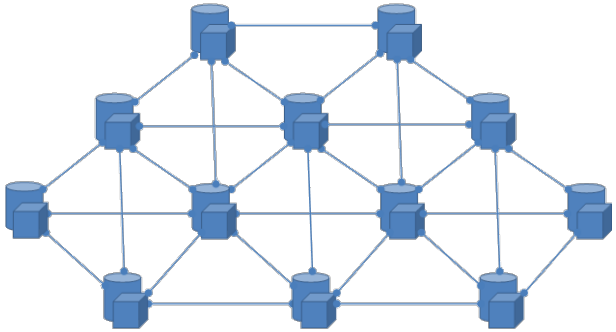
Centralized

one node does everything



Distributed

nodes distribute work to sub-nodes



Decentralized

nodes are only connected to peers

Alert: multiple definitions!

(not always coherent with each other…)

Definitions

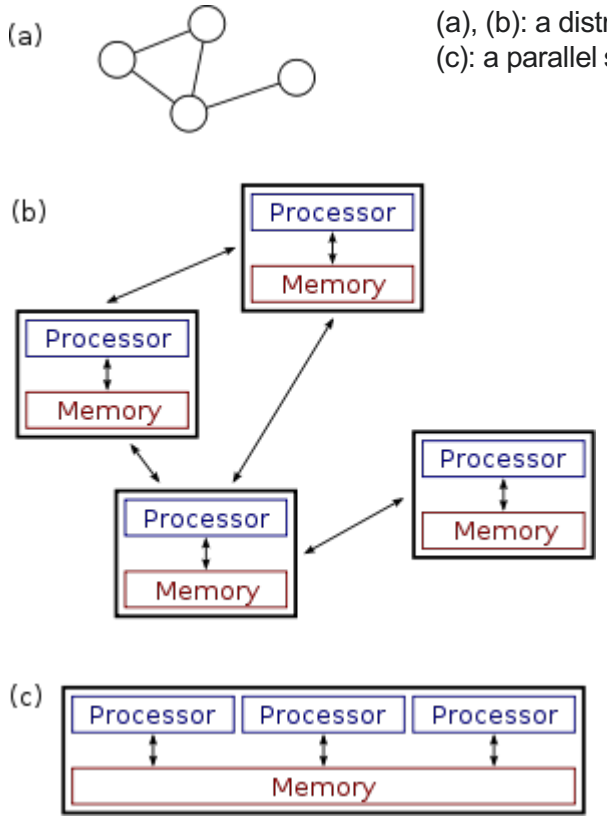
- [van Steen & Tanenbaum] A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system
  - Autonomous computing elements, also referred to as **nodes**, be they hardware devices or software processes
  - Users or applications perceive a single system: nodes need to collaborate
- [Coulouris & Dollimore] A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages
- [Lamport] A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable
  - Emphasis on fault tolerance

Parallel and distributed computing

The terms "*concurrent computing*", "*parallel computing*", and "*distributed computing*" have much overlap, and no clear distinction exists between them.

However, usually:

- In **parallel computing**, all processors may have access to a shared memory to exchange information between processors.
- In **distributed computing**, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.



(a), (b): a distributed system.
(c): a parallel system.

source: https://en.wikipedia.org/wiki/Distributed_computing

Fallacies of distributed computing

Wrong assumptions by architects and designers of distributed systems

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology does not change
6. There is one administrator
7. Transport cost is zero
8. The network environment is homogeneous

source: A. Rotem-Gal-Oz, "Fallacies of distributed computing explained"

Architectural patterns

Architectural patterns for distributed systems

- *Pattern*: a commonly adopted solution for a class of problems

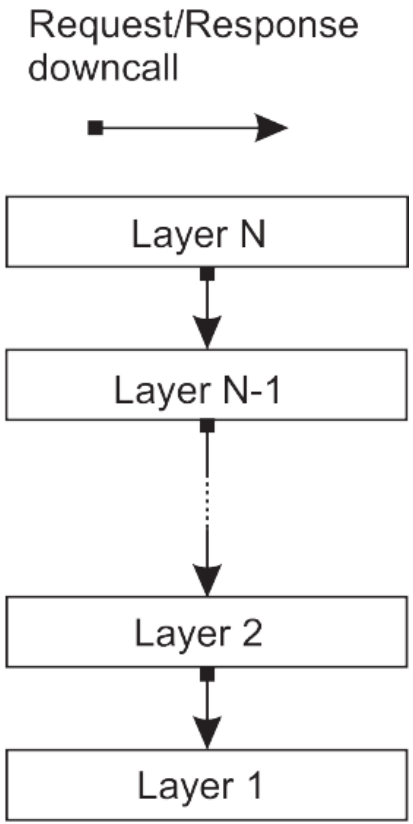
- **Component** (building block)
  - modular unit with well-defined required and provided interfaces
  - fully replaceable with its environment
- **Connector**
  - Mechanism that connects components among each other, mediating communication, coordination or cooperation among components

Main architectural patterns for DS

- Layered
- Object-based
- RESTful
- Event-driven and publish/subscribe

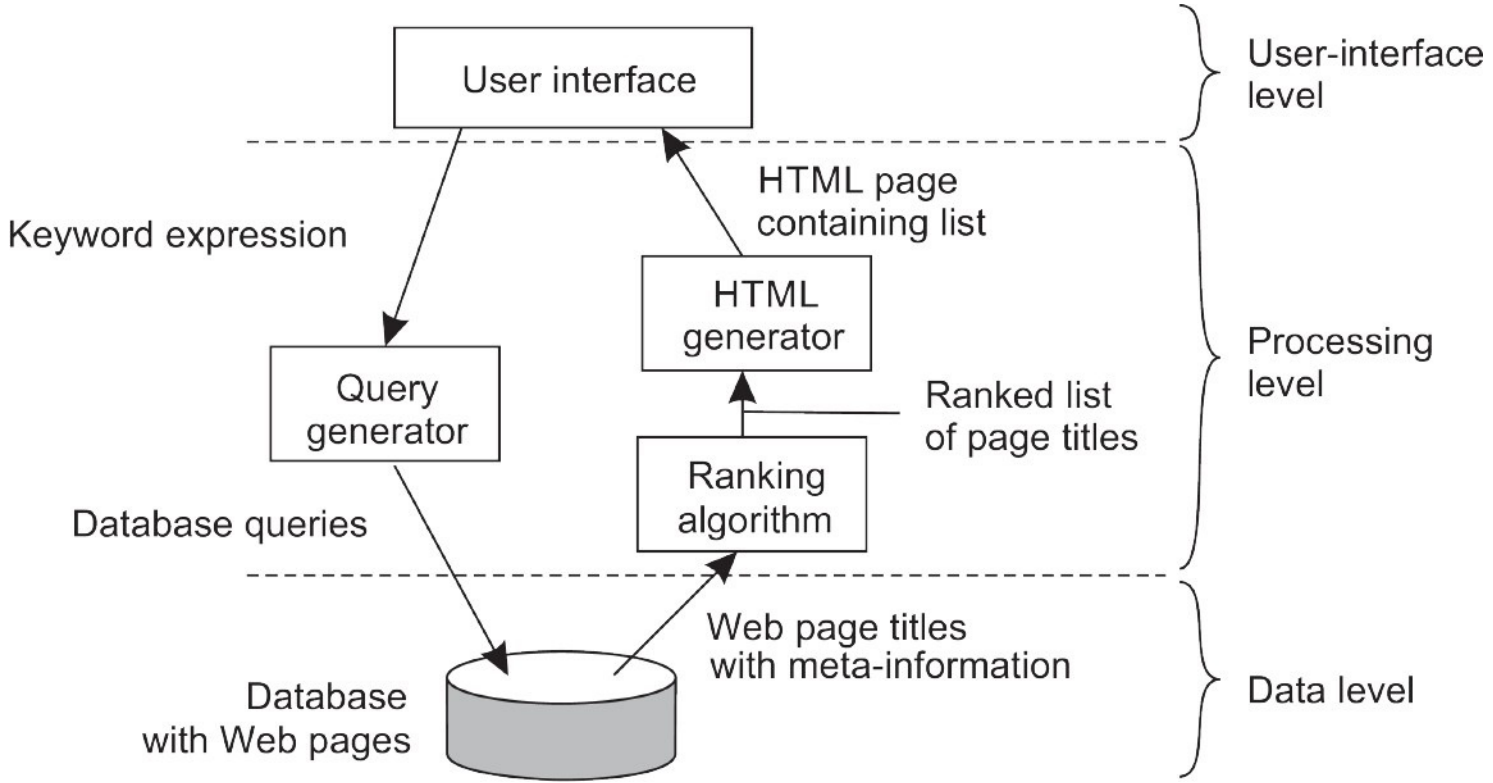
Layered style

- Components organized in layers
- Component at layer I invokes component at layer j (with j<i)
- Components communicate by exchanging messages



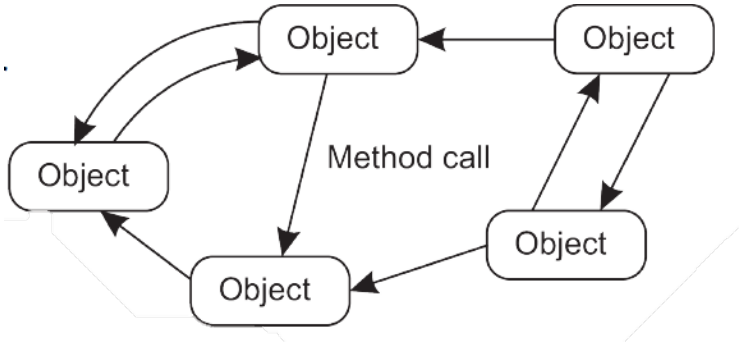
Application layering example

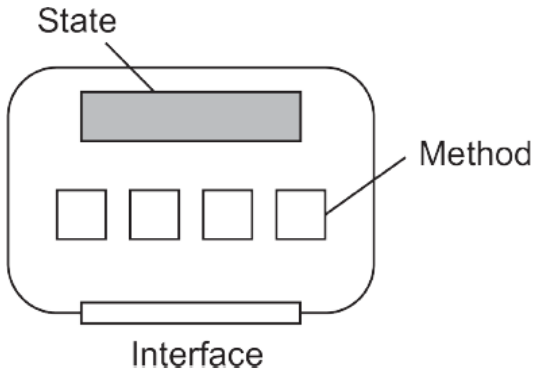
simple web search engine



Object-based style

- **Component=object**: encapsulates a data structure and provides API to access and modify data
  - Encapsulation and information hiding reduce management complexity
  - Object reusability among different apps
  - Wrapping of legacy components
- Communication between components: through remote procedure call (RPC) or remote method invocation (RMI)





RESTful style

- DS as collection of resources, individually managed by components
- Representational State Transfer (REST): proposed by Roy Fielding, co-author of HTTP/1.1
  - Resources may be added, removed, retrieved, and modified by (remote) applications (HTTP methods)
  - Resources are identified through a single naming scheme (Uniform Resource Identifier, URI)

    URI = scheme:[//authority]path[?query][#fragment]
    authority = [userinfo@]host[:port]
- Components expose a **uniform interface**
- Messages sent to/from component are self-described
- Interactions are **stateless**
- State must be transferred from clients to servers

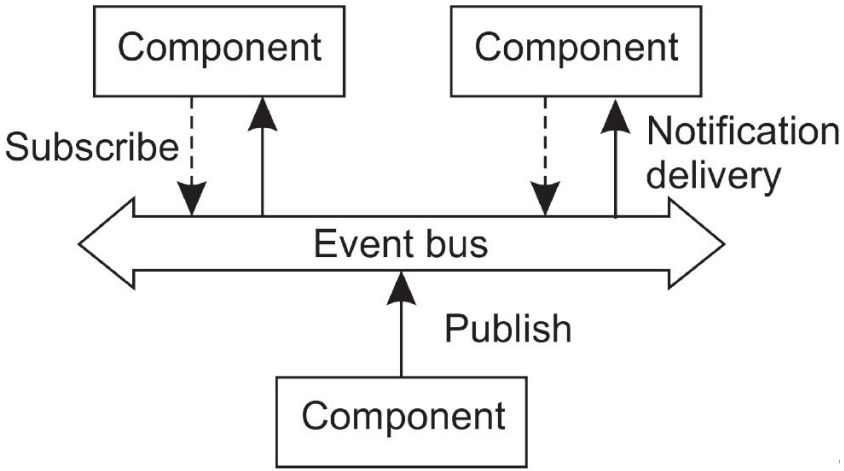
REST operations

- Basic operations with HTTP methods

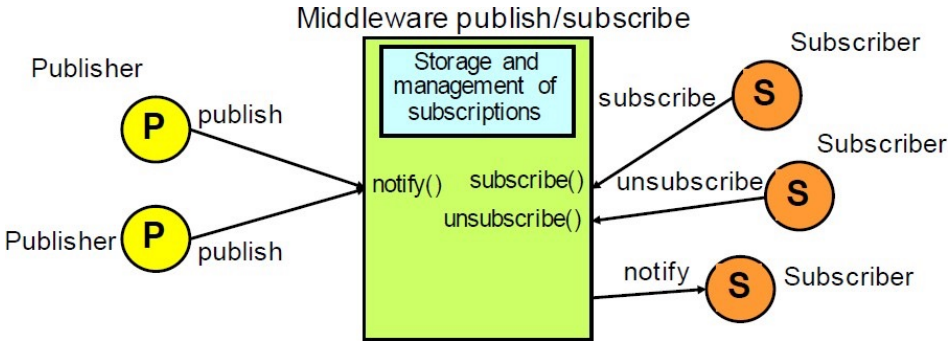
Operation	Description
PUT	Create a new resource
GET	Retrieve the state of a resource in some representation
DELETE	Delete a resource
POST	Modify a resource by transferring a new state

Event-driven style and publish/subscribe

- Components communicate by means of **event propagation**
  - Event: a significant change in state (e.g., change in temperature, opening a door)
- Components
  - Subscribe to events they are interested in being notified
  - Receive notifications about events
- Communication
  - Based on message exchange
  - Asynchronous
  - Multicast
  - Anonymous



- Producers generate events (**publish**) and are not interested in their delivery to subscribers (aka consumers)
- Consumers register as interested to events (**subscribe**) and are notified (notify) of their occurrence
- Full decoupling between interacting entities

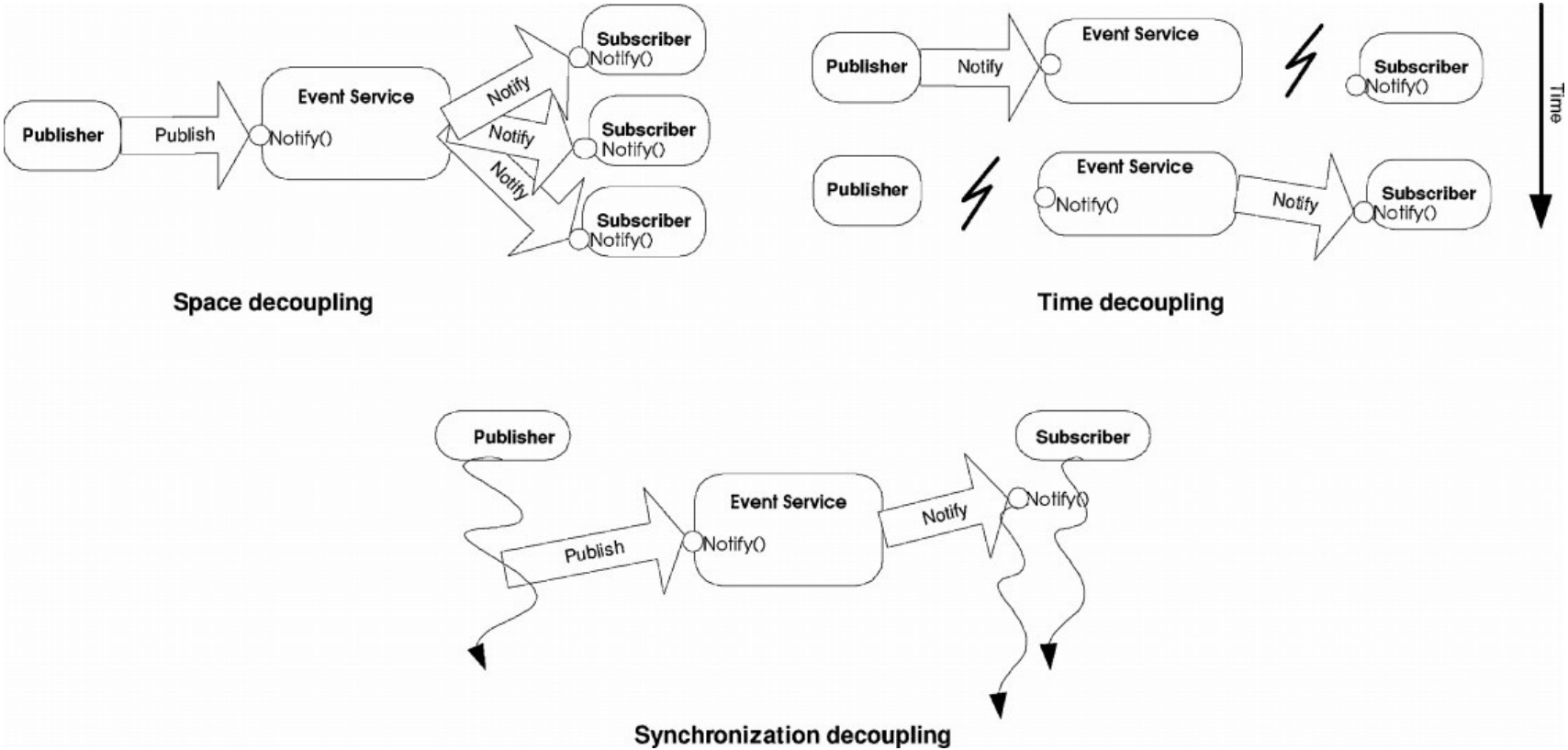


Decoupling

- **Space (or referential) decoupling**
  Components do not need to know each other in order to communicate and cooperate; they are anonymous
- **Time decoupling**
  Interacting components do not need to be present at the same time when communication occurs
- **Synchronization decoupling**
  Interacting components do not need to wait each other and are not reciprocally blocked

source: P.T. Eugster et al., “The many faces of publish/subscribe”

Publish/subscribe and decoupling

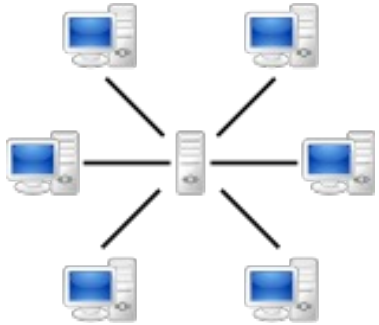


source: P.T. Eugster et al., “The many faces of publish/subscribe”

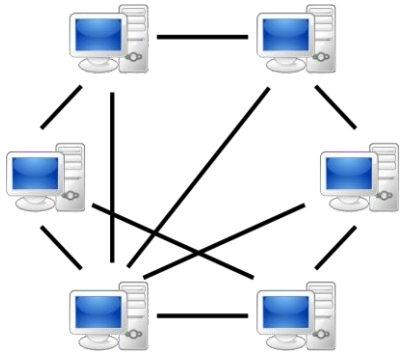
Sample Applications

Peer-to-peer (P2P)

- **Peer-to-peer** (**P2P**) computing or networking is a distributed application architecture that partitions tasks or workloads between peers.
- Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes



A network based on the **client-server model**, where individual clients request services and resources from centralized servers



A **peer-to-peer (P2P) network** in which interconnected nodes ("peers") share resources amongst each other without the use of a centralized administrative system

Blockchain

- A blockchain is a growing list of records, called blocks, that are linked using cryptography.
- Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree).
- By design, a blockchain is resistant to modification of its data. This is because once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks.
- For use as a distributed ledger, a blockchain is typically managed by a **peer-to-peer network** collectively adhering to a protocol for inter-node communication and validating new blocks.

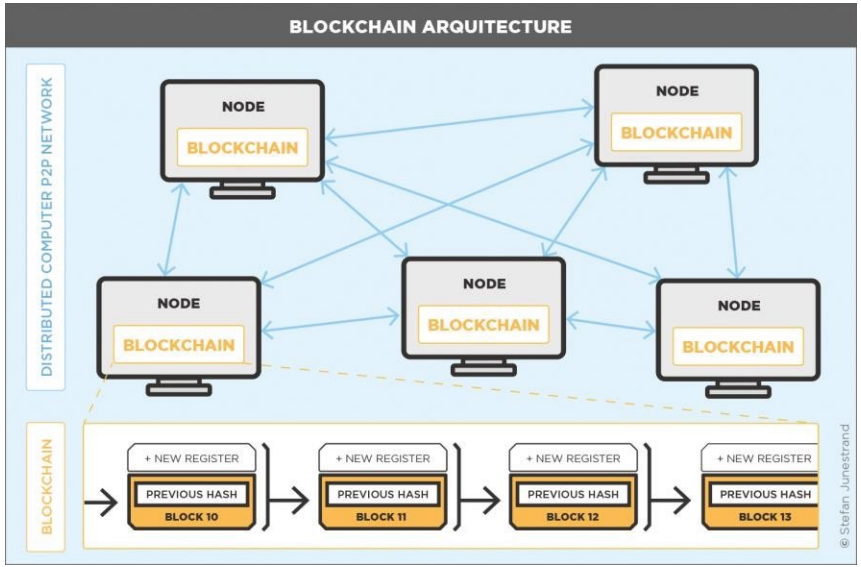


image source: https://www.stefanjunestrand.com/blog/what-is-blockchain/

Reading list

A. Rotem-Gal-Oz, "Fallacies of distributed computing explained“ ()

P.T. Eugster, P.A. Felber, R. Guerraoui, A.-M. Kermarrec, "The many faces of publish/subscribe", *ACM Computing Surveys* (2003)

(documents available on Blackboard)

Questions, comments?