Distributed Systems

Models

Architectural elements

- ***Processes*** - the entities that communicate in a distributed system are typically *processes* coupled with appropriate interprocess communication paradigms (*nodes* in sensor networks); processes are supplemented by ***threads*** - are the endpoints of communication.
- ***Objects*** - in distributed object-based approaches, a computation consists of a number of interacting objects representing natural units of decomposition for the given problem domain. Objects are accessed via interfaces, with an associated interface definition language (or IDL) providing a specification of the methods defined on an object.
- ***Components*** - software component is a unit of composition with contractually specified interfaces and explicit context dependencies .
- ***Web services*** - software application identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered; supports direct interactions with other software agents via Internet-based protocols.

Communication paradigms

- interprocess communication
  - low-level support for communication between processes in distributed systems, including message-passing primitives, direct access to the API offered by Internet protocols (socket programming) and support for multicast communication.
- remote invocation
  - represents the most common communication paradigm in distributed systems, covering a range of techniques based on a two-way exchange between communicating entities and resulting in the calling of a remote operation, procedure or method;
- indirect communication

Communication paradigms

- interprocess communication
- remote invocation
  - Request-reply protocols are effectively a pattern imposed on an underlying message-passing service to support client-server computing;
  - Remote procedure calls (RPC) - procedures in processes on remote computers can be called as if they are procedures in the local address space
  - Remote method invocation (RMI) - supporting object identity and the associated ability to pass object identifiers as parameters in remote calls.
- indirect communication
  - decoupling between senders and receivers

Communication paradigms

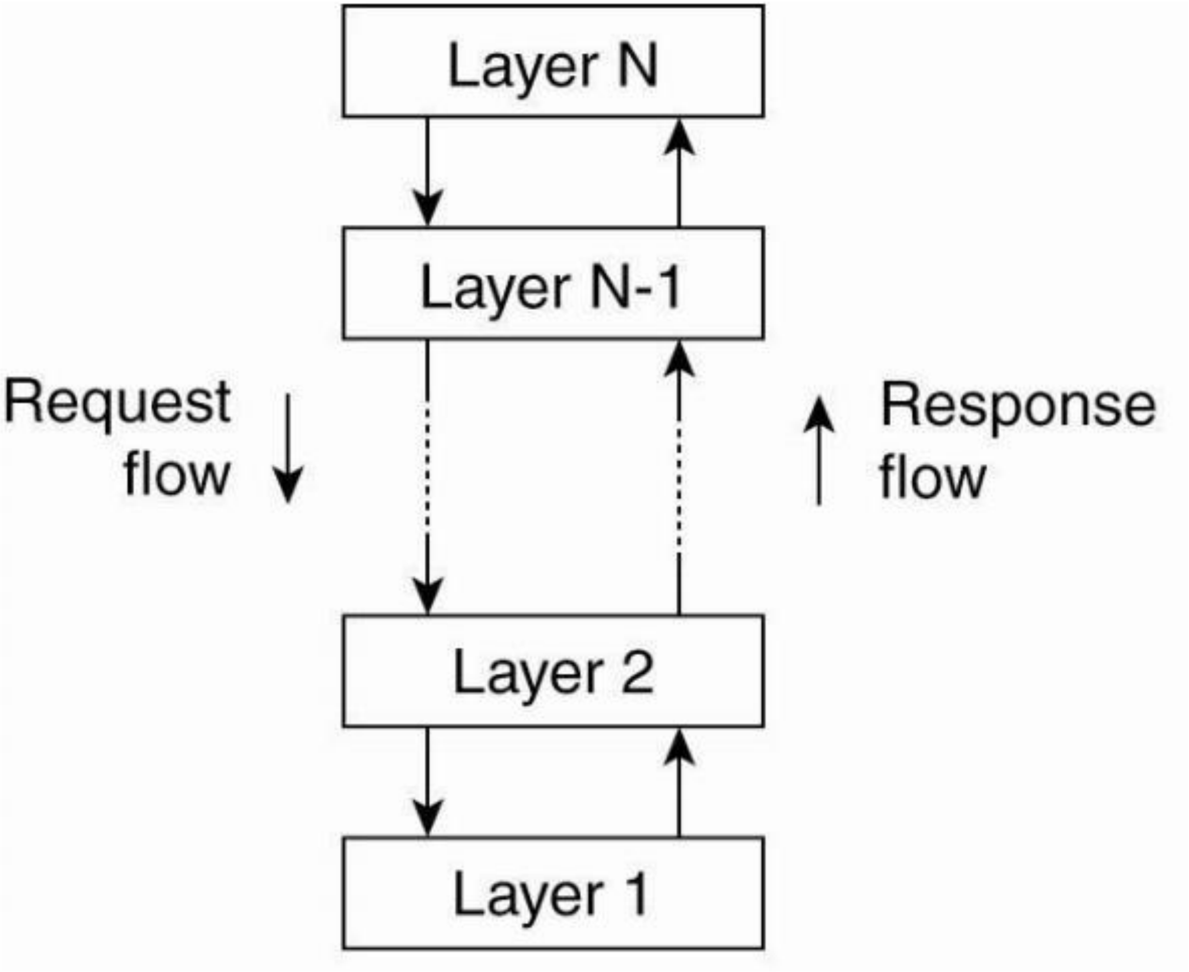
- interprocess communication
- remote invocation
- **indirect communication** - decoupling between senders and receivers
  - Senders do not need to know who they are sending to (space uncoupling).
  - Senders and receivers do not need to exist at the same time (time uncoupling).
  - decoupling processes in space ("anonymous") and also time ("asynchronous") has led to alternative styles.

  - Group communication
  - Publish-subscribe systems
  - Message queues
  - Tuple spaces - processes can place arbitrary items of structured data, called tuples, in a persistent tuple space and other processes can either read or remove such tuples from the tuple space by specifying patterns of interest
  - Distributed shared memory - abstraction of reading or writing (shared) data structures as if they were in their own local address spaces, thus presenting a high level of distribution transparency.

Architectural Styles

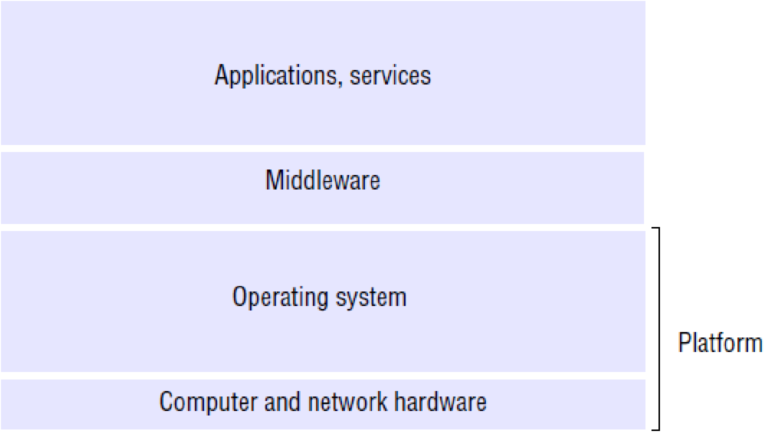
- Architecture - structure in terms of separately specified components and their interrelationships.
- An **architectural style** describes a particular way to configure a collection of components and connectors.
  - **Component** - a module with well-defined interfaces; reusable, replaceable
  - **Connector** – communication link between modules (RPC, RMI)
- Architectures suitable for distributed systems:
  - **Layered** architectures
  - **Object-based** architectures
  - **Data-centered** architectures
  - **Event-based** architectures

Layered style



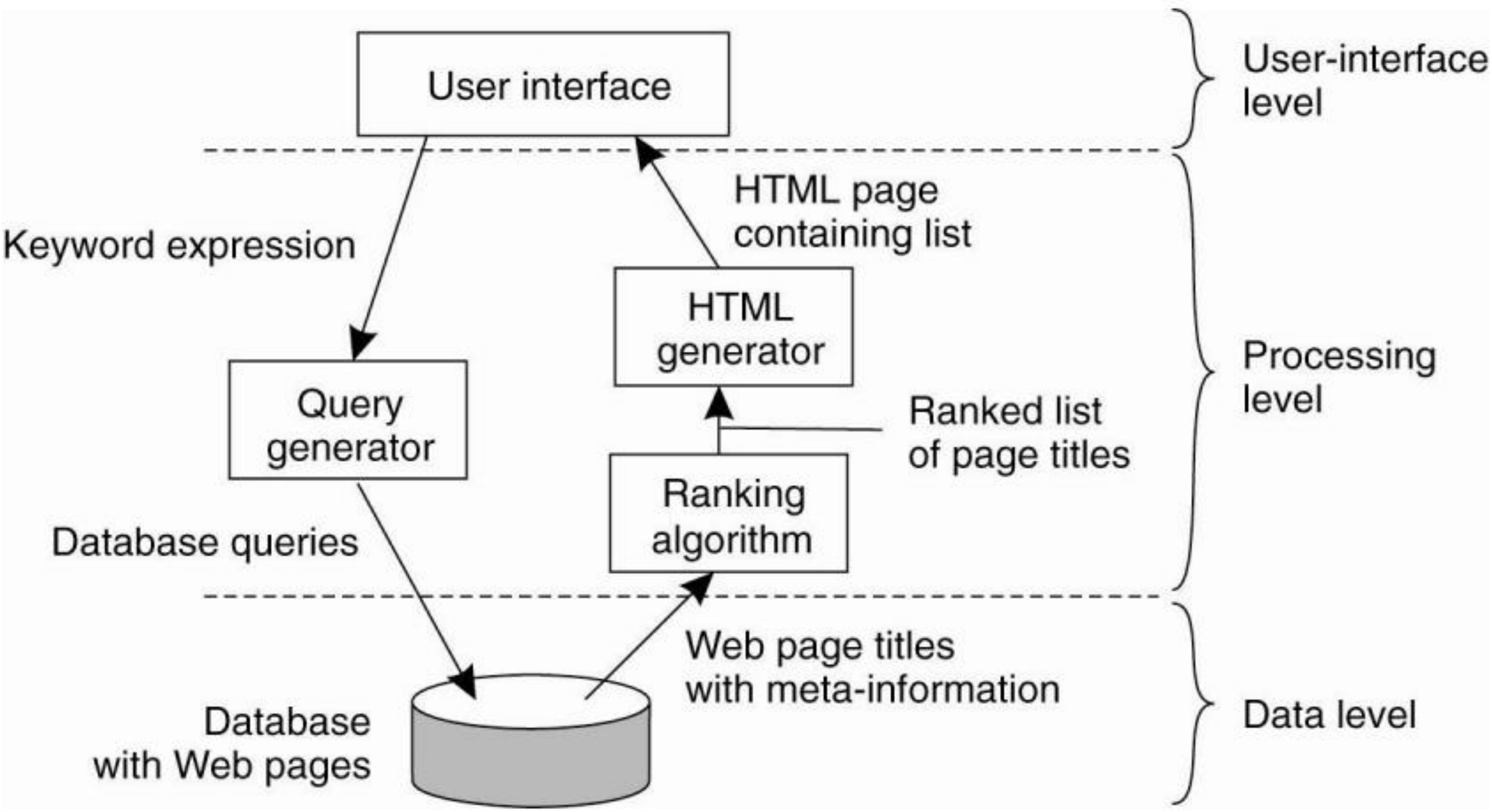
Software and hardware service layers in distributed systems

- A given layer offers a software abstraction, with higher layers being unaware of implementation details
- **Platform** for distributed systems and applications consists of the lowest-level hardware and software layers; provide services to the layers above them, which are implemented independently in each computer (see PaaS)



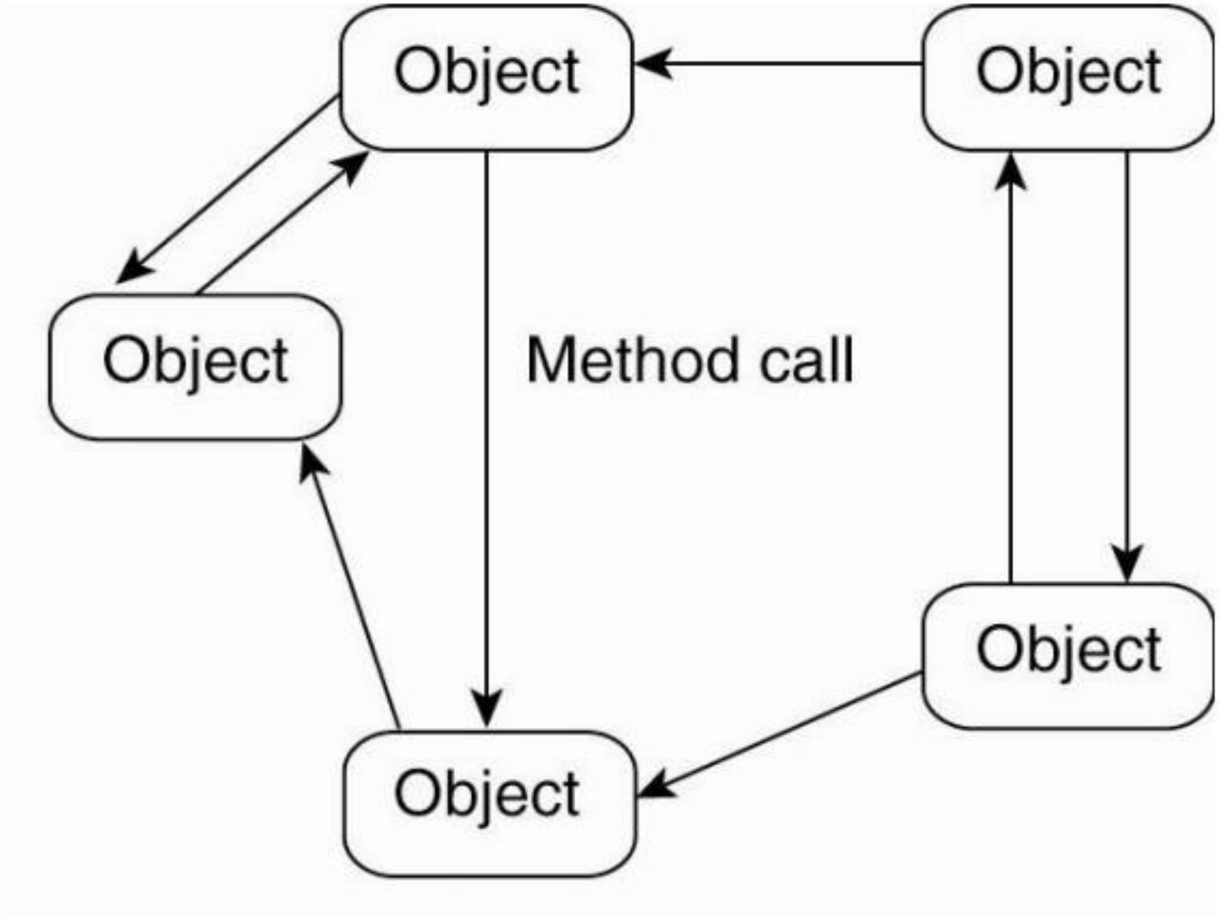
- **Middleware** - a layer of software whose purpose is to mask heterogeneity and to provide a convenient programming model to application programmers.
- Middleware is represented by processes or objects in a set of computers that interact with each other to implement communication and resource-sharing support for distributed applications.
- Middleware provides building blocks for the construction of software components that can work with one another in a distributed system. In particular, it raises the level of the communication activities of application programs through the support of abstractions such as remote method invocation; communication between a group of processes; notification of events; the partitioning, placement and retrieval of shared data objects amongst cooperating computers; the replication of shared data objects; and the transmission of multimedia data in real time.

Internet Search Engine in 3 Layers



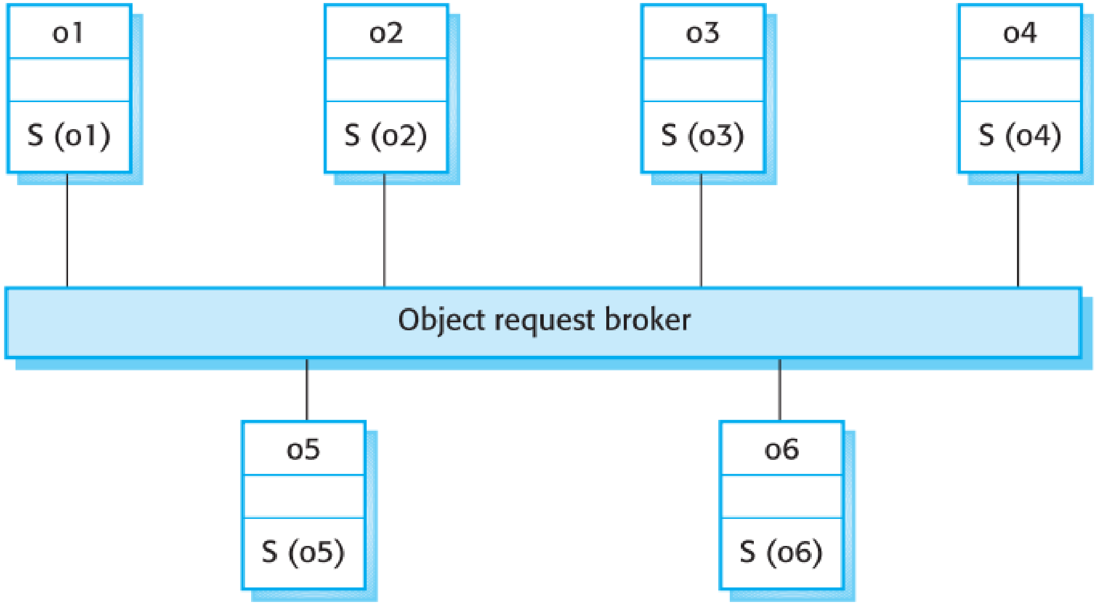
see layers vs tiers

object-based architectures

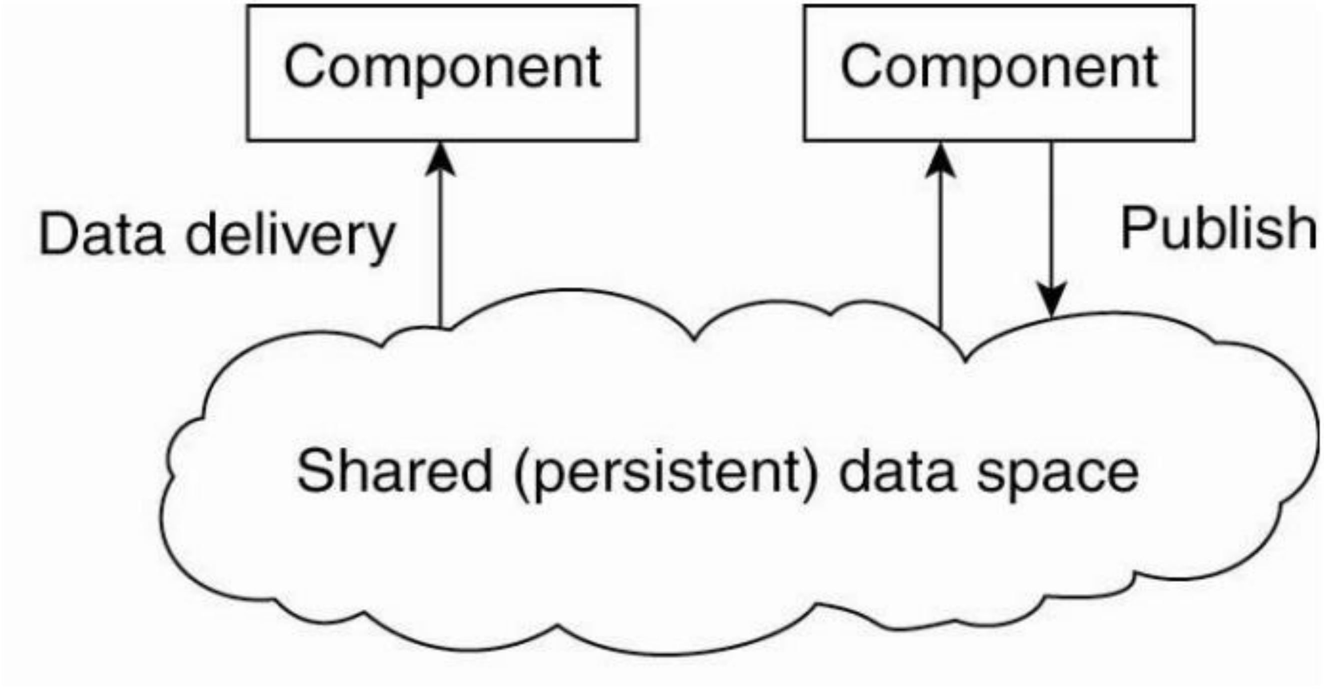


Distributed object architectures

- There is no distinction in a distributed object architectures between clients and servers.
  - Each distributable entity is an object that provides services to other objects and receives services from other objects.
  - Object communication is through a middleware system called an object request broker.
  - more complex to design than C/S systems.

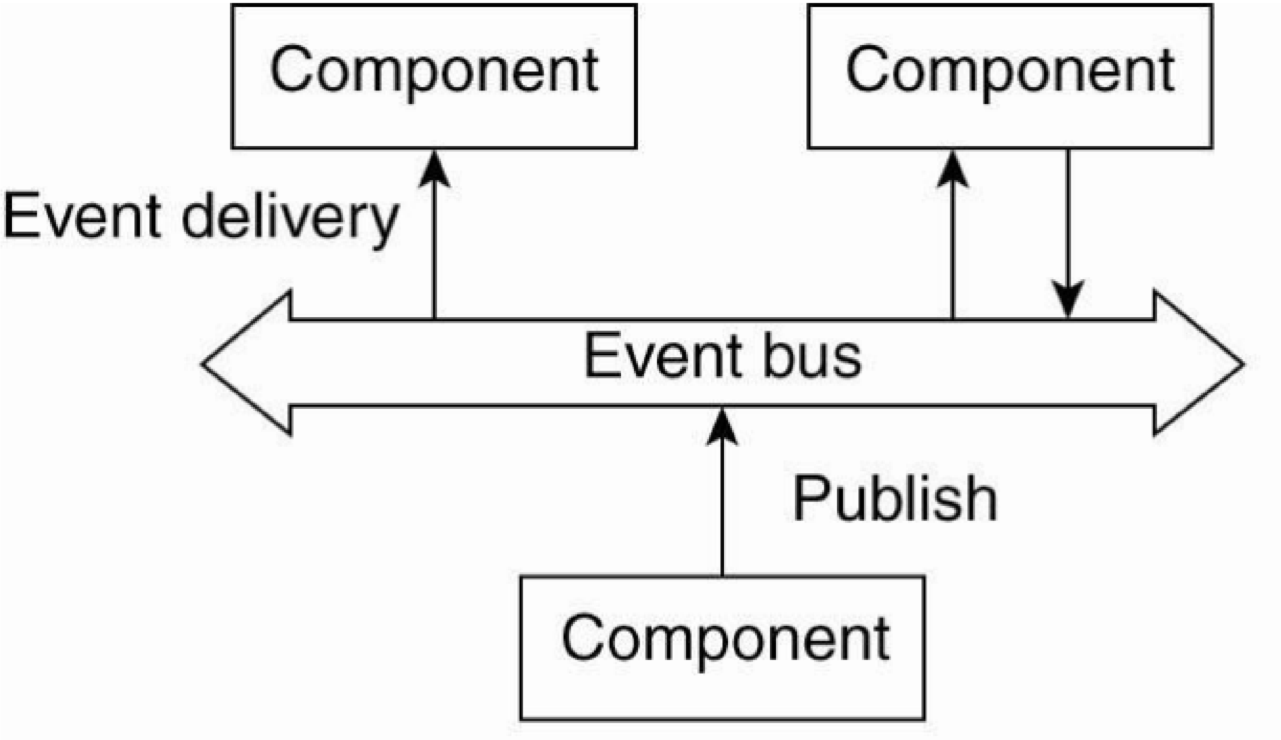


Data-centered architectures



- processes communicate through a common repository (passive or active).
- shared dataspace (decoupled in space and time)

event-based architectures



- processes communicate through the propagation of events
- An event can be defined as "a significant change in state“
- Decouples sender & receiver; asynchronous communication
- publish/subscribe, broadcast

Published-subscribe systems

- An event service system - asynchronous, anonymous and loosely-coupled.

- Publishers : Publishers generate event data and publishes them.
- Subscribers : Subscribers submit their subscriptions and process the events received
- P/S service: It's the mediator/broker that filters and routes events from publishers to interested subscribers.

- **Centralized Broker model**
  - Consists of multiple publishers and multiple subscribers and centralized broker/brokers (an overlay network of brokers interacting with each other).
  - Subscribers/Publishers will contact 1 broker, and does not need to have knowledge about others.
  - E.g. CORBA event services, JMS, etc…

System Architectures for Distributed Systems

- **Centralized**: traditional client-server structure
  - Vertical (or hierarchichal) organization of communication and control paths (as in layered software architectures)
  - Logical separation of functions into client (requesting process) and server (responder)
- **Decentralized**: peer-to-peer
  - Horizontal rather than hierarchical comm. and control
  - Communication paths are less structured; symmetric functionality
- **Hybrid:** combine elements of C/S and P2P
  - Edge-server systems
  - Collaborative distributed systems.

- Classification of a system as centralized or decentralized refers to communication and control organization, primarily.

Centralized v Decentralized Architectures

- Traditional client-server architectures exhibit **vertical distribution**. Each level serves a different purpose in the system.
  - *Logically* different components reside on different nodes

- **Horizontal distribution** (P2P): each node has roughly the same processing capabilities and stores/manages part of the total system data.
  - Better load balancing, more resistant to denial-of-service attacks, harder to manage than C/S
  - Communication & control is not hierarchical; all about equal
  - Many application areas: file sharing, streaming, process sharing, collaborative applications, web-caching etc

Peer-to-Peer

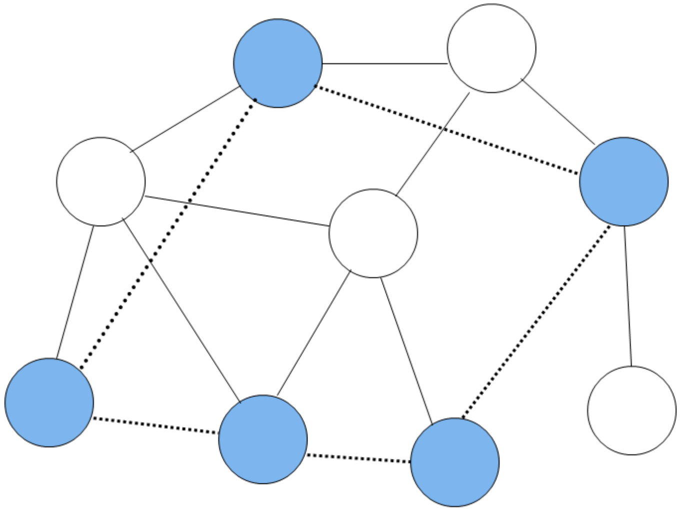
- Nodes act as both client and server; interaction is symmetric
- Every node act both as a client and server, and “pays” for the participation by offering access to some if its resources (typically processing and storage resources, but can also be logical resources (services)
- Each node acts as a server for part of the total system data

- Advantages: no single point of failure, scalability
- Disadvantages: complexity of protocols

- **Overlay networks** connect nodes in the P2P system
  - Each node in a P2P system knows how to contact several other nodes
  - Nodes in the overlay use their own addressing system for storing and retrieving data in the system
  - Nodes can route requests to locations that may not be known by the requester.

Overlay Networks

- Are logical or *virtual* networks, built on top of a physical network
- A link between two nodes in the overlay may consist of several physical links.
- Messages in the overlay are sent to logical addresses, not physical (IP) addresses
- Various approaches used to resolve logical addresses to physical.



Circles represent nodes in the network. Blue nodes are also part of the overlay network. Dotted lines represent virtual links. Actual routing is based on TCP/IP protocols

Fog Computing

- edge computing - servers are placed "at the edge" of the network.

- Fog Computing extends the cloud computing paradigm to the edge of the network to address applications and services that do not fit the paradigm of the cloud including:

- Applications that require very low and predictable latency
- Geographically distributed applications
- Fast mobile applications
- Large-scale distributed control systems (smart grid, connected rail, smart traffic light systems).

http://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing
https://en.wikipedia.org/wiki/Edge_computing
https://en.wikipedia.org/wiki/Fog_computing