Distributed System Architectures

Distributed systems are often complex pieces of software in which components are dispersed across multiple machines in a network. In order to understand such complexity of a distributed system, it is always crucial to organise such components. ***The organisation of a distributed system is mostly about the constitutes of the software components that tells us how various software components are organised and how they interact.***

Distributed systems can be organized in many ways. We can make a distinction between **software architecture** and **system architecture**.

The **software architecture** considers where the components that constitute a distributed system are placed across the various machines. **System architecture** is more concerned about the logical organization of the software: how do components interact, it what ways can they be structured, how can they be made independent, and so on.

As explained earlier, the goal of a distributed system is to separate applications from the underlying platforms by providing a **middleware layer** for the purpose of providing distribution transparency.

However, trade-offs need to be made to achieve this transparency, this leads to adoption of various techniques to make middleware adaptive.

Adaptability in distributed systems can be achieved by having the system monitor its own behaviour and taking appropriate corrective measures. This is referred to as **autonomic systems**. Which are organised with some form of feedback control loops which form an important architectural element during system design.

Architectural styles

A key idea when talking about architectures is **architectural style**. A style reflects the basic principle that is followed in organizing the interaction between the software components comprising a distributed system.

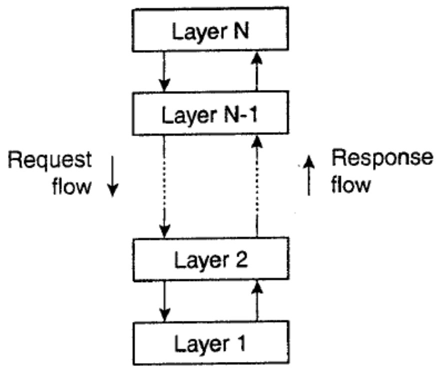
Let's consider the logical organisation of distributed systems into software components that is referred to as **software architecture**. For a successful designing, development or adopting of a large system, the choice of an architecture to use is crucial.

Architectural styles are formulated based on how components are connected to each other, how data is exchanged between the components and how these components are jointly configured to form a system.

Several styles have been identified for distributed systems. This include;

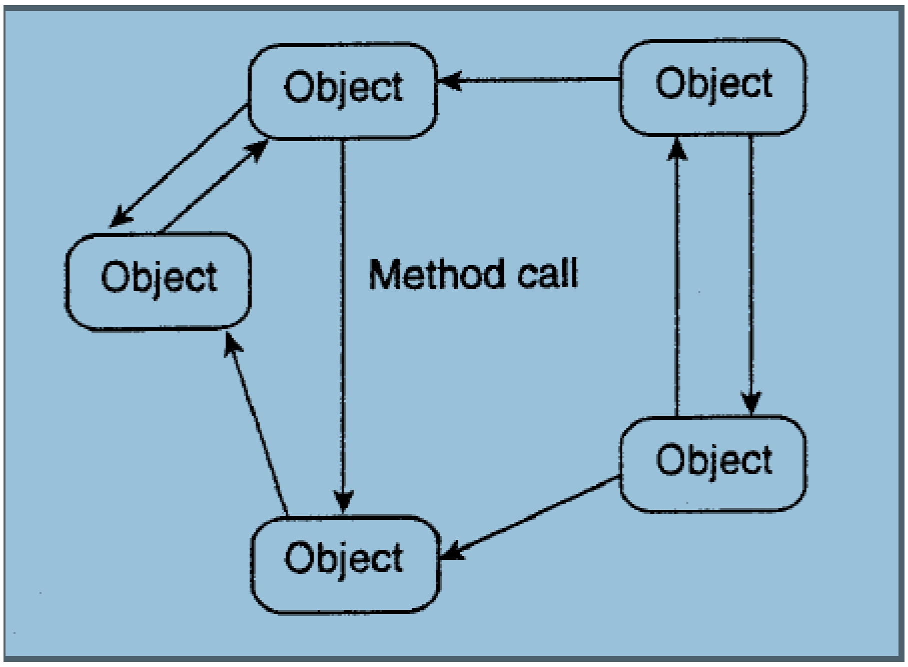
- Layered architectures
- Object-based architectures
- Data-centred architectures
- Event – based architectures

For **layer architecture**, components are organised into a layered fashion where components are allowed to call components in layers above them. This approach has widely been adopted by the networking community.



For **object-based architecture**, each object corresponds to a component and these components are connected through a **remote procedure call (RPC)** mechanism. This architecture matches **client-server system architecture**.

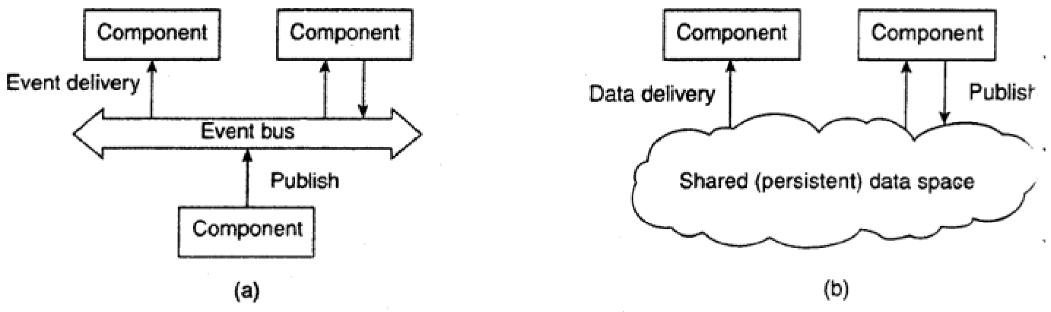
The layered and object-based architecture forms the most important styles for large software systems.



Data-centred architecture evolves around the idea of process communication through a common (active/passive) repository.

In **event-based architectures**, processes communicate through the propagation of events which are associated with publish/subscribe systems. The basic idea is that processes publish events after which the middleware ensures that only those processes that subscribed to those events will receive them.

Event-based architectures can be combined with data-centred architectures to form what is referred to as **share data space**.



The (a) event-based and (b) shared data-space architectural style.

System architectures

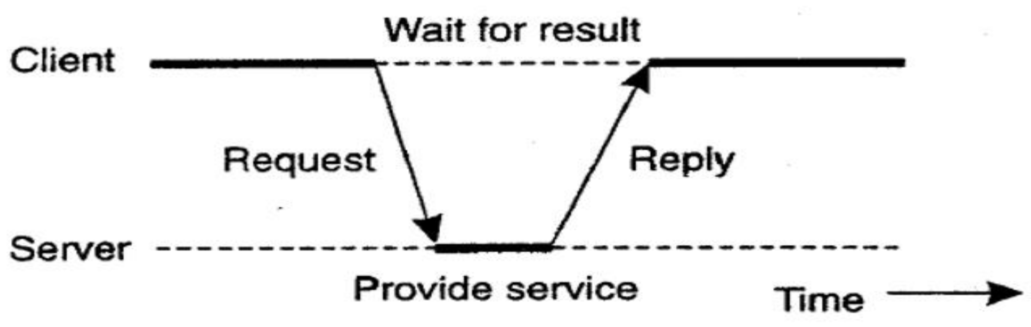
There are many different organizations of distributed systems. Let's look at how distributed systems are organised by considering where software components are placed. System architecture is also referred to as software architecture and is concerned with software components in a distributed system, their interaction and placement.

This shall be divided into 3;

1. Centralised organisation
2. Decentralise organisation
3. Hybrid organisation

Centralised architecture

An important class is where machines are divided into **clients and servers**. A client sends a request to a server, who will then produce a result that is returned to the client. The client-server architecture reflects the traditional way of modularizing software in which a module calls the functions available in another module. By placing different components on different machines, we obtain a natural physical distribution of functions across a collection of machines. Client-server architectures are often highly **centralized**. In this, a **server** is a process implanting a specific service e.g. a file system or database service, a **client** is a process that requests a service from a server by sending it a request and subsequently waiting for the server to reply. This client server interaction is also known as **request-reply behaviour**.



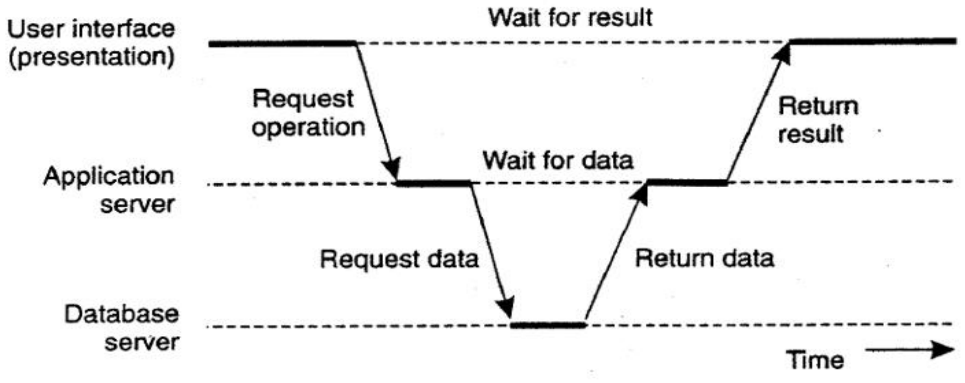
General interaction between a client and a server.

The figure above is an example of two-tier client-server architecture

Client-servers architectures has had debates and contrivances over years over how to draw a distinction between the client and the server. For instance, in some cases, a server for a distribution database can also act as a client. This results to server itself doing more than just process queries.

Considering that many client-server application targets supporting users access to a database, the distinction has been brought be considering a layered architectural design as listed

1. The user-interface level
2. The processing level
3. The data level.

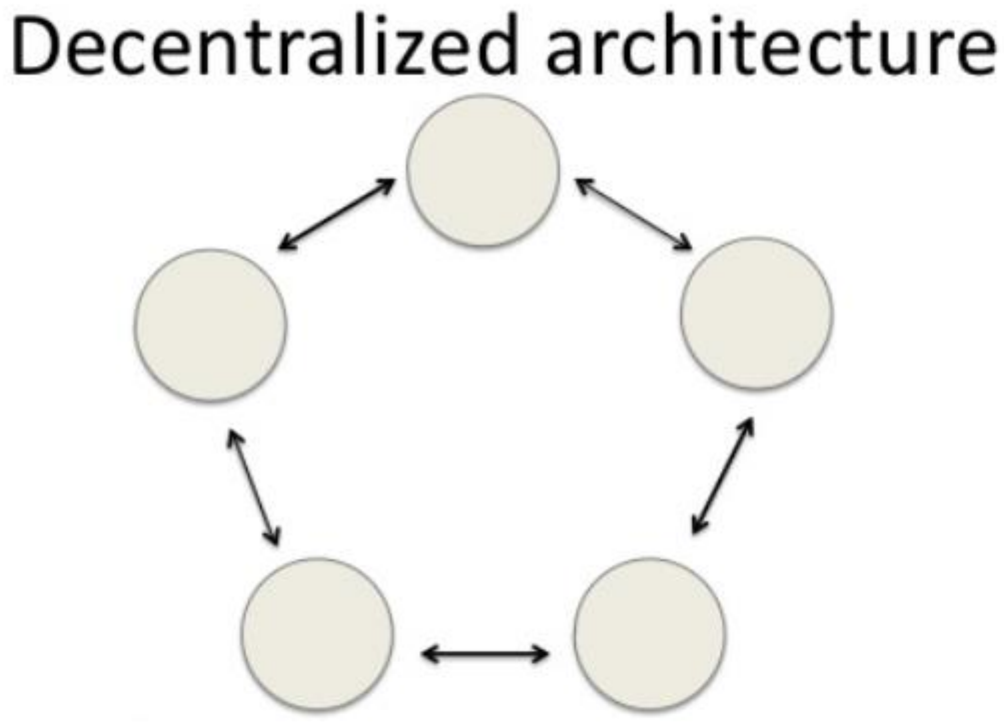


An example of a server acting as client.

The figure above is an example of three-tier client-server architecture

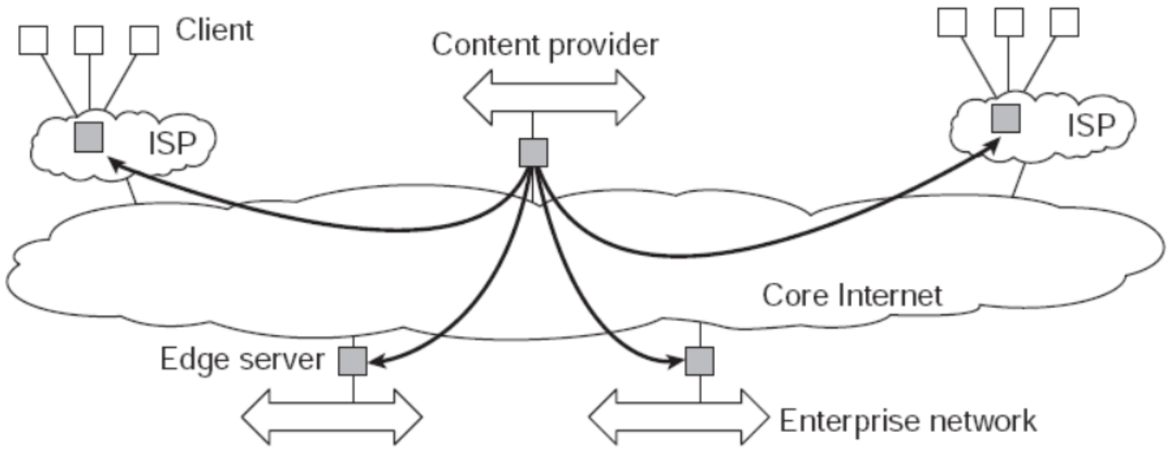
Decentralise organisation

In **decentralized architectures** we often see an equal role played by the processes that constitute a distributed system, also known as **peer-to-peer systems**. In peer-to-peer systems, the processes are organized into an **overlay network**, which is a logical network in which, every process has a local list of other peers that it can communicate with. The **overlay network** can be structured, in which case deterministic schemes can be deployed for routing messages between processes. In unstructured networks, the list of peers is more or less random, implying that search algorithms need to be deployed for locating data or other processes.



Hybrid Architectures: ***Client-server combined with P2P***

An important class of distributed systems that is organized according to a hybrid architecture is formed by edge-server systems. These systems are deployed on the Internet where servers are placed "at the edge" of the network. End users, or clients in general, connect to the Internet by means of an edge server.



Research Questions:

1. **With the aid of a diagram, explain what is meant by THIN and FAT clients in centralized architectures.**
2. **Processes in event-based architectures are said to be decoupled in space or referentially decoupled. Explain what this implies and its advantage.**
3. **Discuss the various design requirements for distributed architectures.**
4. **Discuss the various advantages and disadvantages of centralised architecture.**
5. **Discuss the various advantages and disadvantages of decentralised architecture.**
6. **Using a diagram, describe superpeer architecture as used in distributed systems**
7. **Explain the concept of overlay network as used in decentralised architecture**