Architecture Styles in Distributed Systems

Architecture Styles in Distributed Systems

Distributed systems are a set of autonomous computers that appears to be a single coherent system to its users from outside. There are actually two different types of systems that exist in prospective of computers:

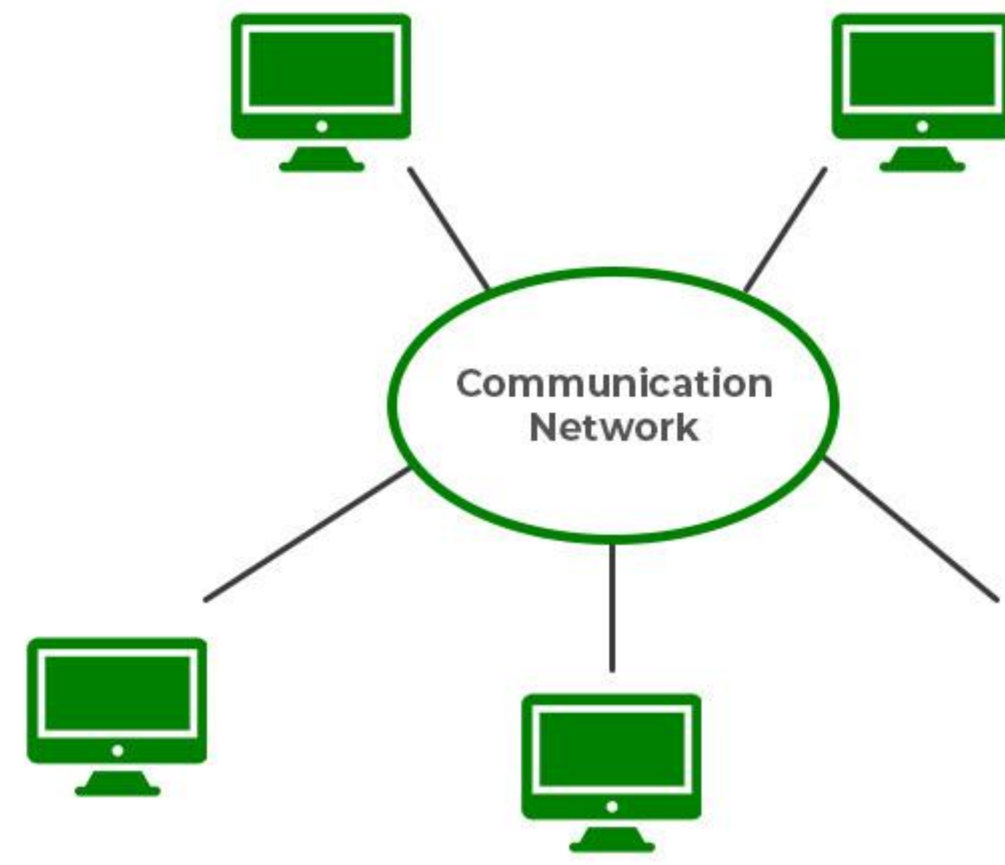
Centralized System :

- A centralized system consists of a single machine.
- All calculations are done by a particular computer.
- Its performance is low as the workload is not divided.
- There is also no machine present in backup if the original computer system fails.

Distributed Systems :

- A distributed system consists of multiple machines.
- All computation work is divided among the different systems.
- Its performance is high as the workload is divided among different computers to efficiently use their capacity.
- There are systems present in backup, so if the main system fails then work will not stop.

A distributed system contains multiple nodes that are physically separate but mixed together using the communication networks.



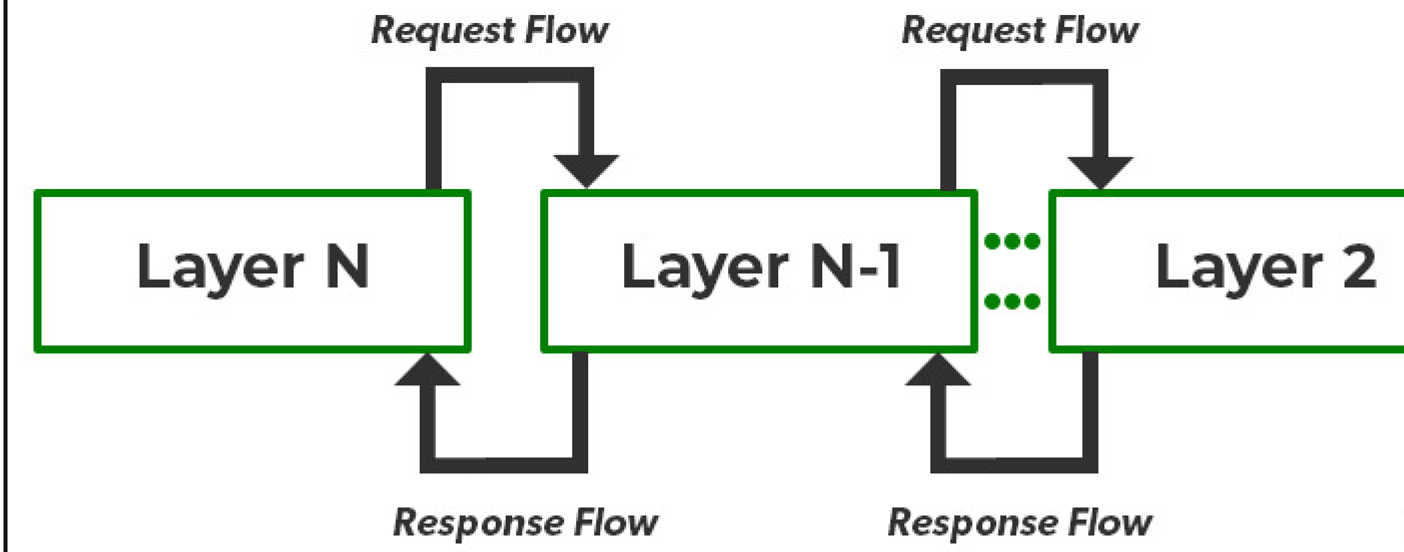
Architecture Styles:
To show different arrangement styles among computers Architecture styles are proposed.

1. Layered Architecture:

In Layered architecture, different components are organised in layers. Each layer communicates with its adjacent layer by sending requests and getting responses. The layered architecture separates components into units. It is an efficient way of communication. Any layer can not directly communicate with another layer. A layer can only communicate with its neighbouring layer and then the next layer transfers information to another layer and so on the process goes on.

In some cases, layered architecture is in cross-layer coordination. In a cross-layer, any adjacent layer can be skipped until it fulfils the request and provides better performance results. Request flow from top to bottom(downwards) and response flow from bottom to top(upwards). The advantage of layered architecture is that each layer can be modified independently without affecting the whole system. This type of architecture is used in Open System Interconnection (OSI) model.

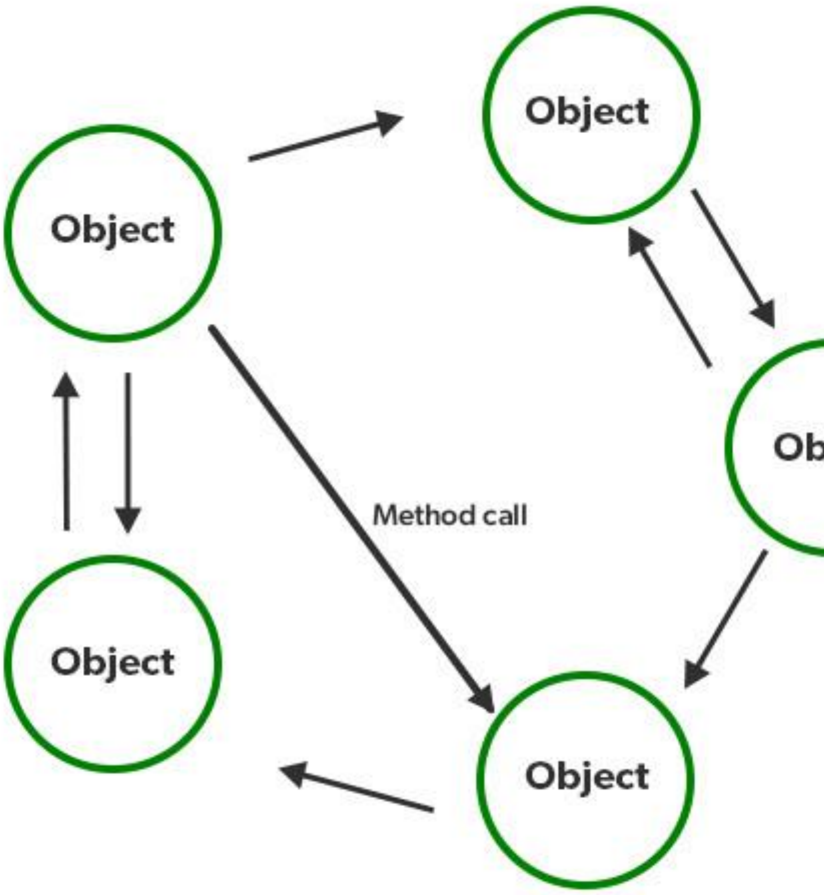
To the layers on top, the layers at the bottom offer a service. While the response is transmitted from bottom to top, the request is sent from top to bottom. This method has the advantage that calls always follow a predetermined path and that each layer is simple to replace or modify without affecting the architecture as a whole.



2. Object-Oriented Architecture:

In this type of architecture, components are treated as objects which convey information to each other. Object-Oriented Architecture contains an arrangement of loosely coupled objects. Objects can interact with each other through method calls. Objects are connected to each other through the Remote Procedure Call (RPC) mechanism or Remote Method Invocation (RMI) mechanism. Web Services and REST API are examples of object-oriented architecture. Invocations of methods are how objects communicate with one another. Typically, these are referred to as Remote Procedure Calls

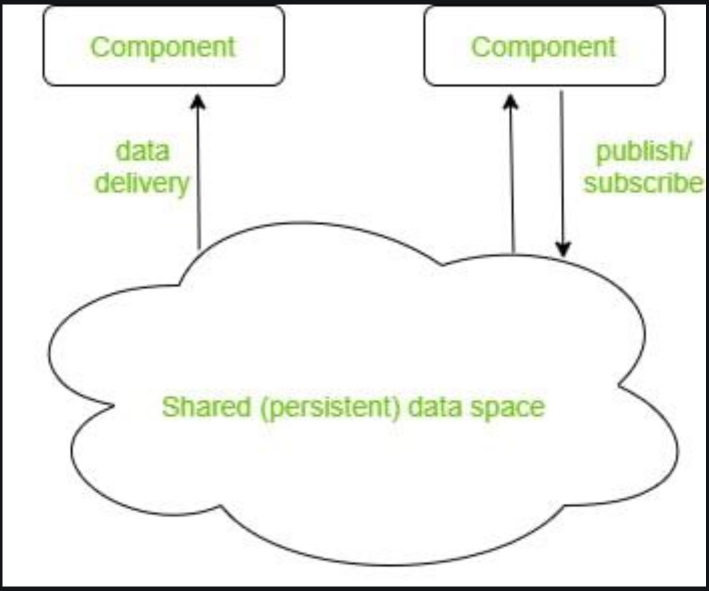
(RPC). REST API Calls, Web Services, and Java RMI are a few well-known examples. These characteristics apply to this.



3. Data Centered Architecture:

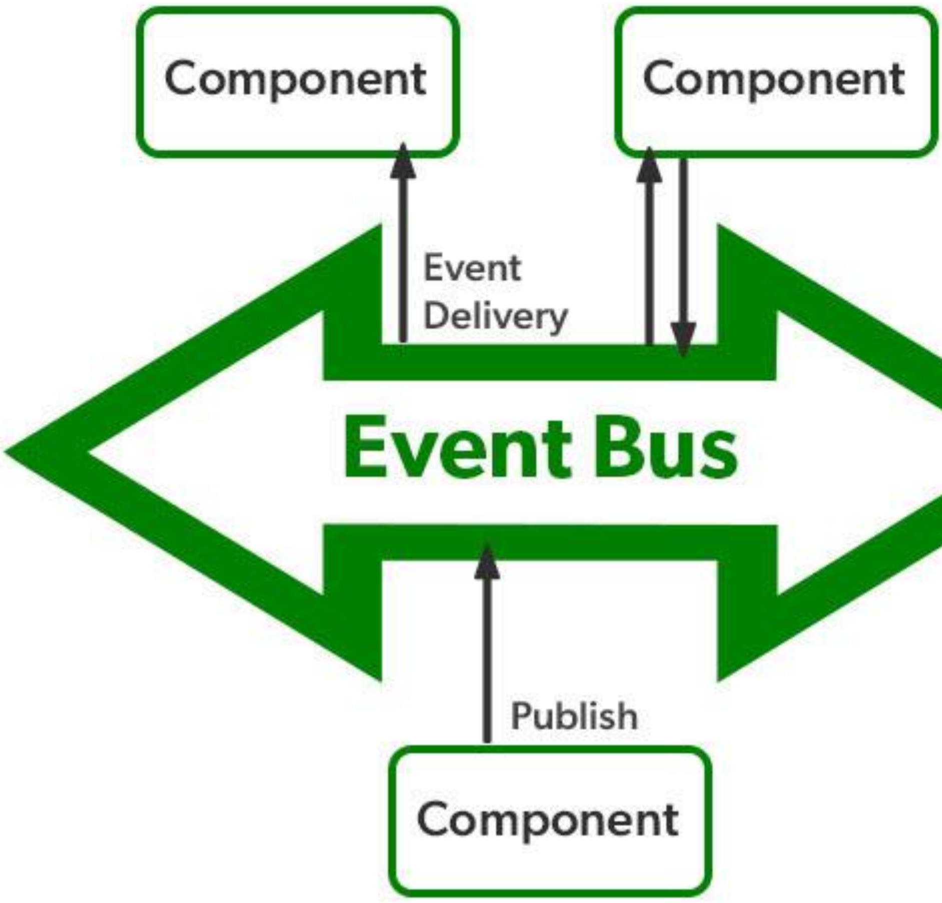
Data Centered Architecture is a type of architecture in which a common data space is present at the centre. It contains all the required data in one place a shared data space. All the components are connected to this data space and they follow publish/subscribe type of communication. It has a central data repository at the centre. Required data is then delivered to the components. Distributed file systems, producer-consumer systems, and web-based data services are a few well-known examples.

For example Producer-Consumer system. The producer produces data in common data space and consumers request data.



4. Event-Based Architecture:

Event-Based Architecture is almost similar to Data centered architecture just the difference is that in this architecture events are present instead of data. Events are present at the centre in the Event bus and delivered to the required component whenever needed. In this architecture, the entire communication is done through events. When an event occurs, the system, as well as the receiver, get notified. Data, URLs etc are transmitted through events. The components of this system are loosely coupled that’s why it is easy to add, remove and modify them. Heterogeneous components can communicate through the bus. One significant benefit is that these heterogeneous components can communicate with the bus using any protocol. However, a specific bus or an ESB has the ability to handle any kind of incoming request and respond appropriately.


Component
Component
Event Delivery
Event Bus
Publish
Component