MANCHESTER
1824
The University of Manchester

Architectures of Distributed Systems

Architectures of Distributed Systems

- An obvious way to distinguish between distributed systems is on the organisation of their software components and how they interact. In other words, their software architecture.
  - Centralised architectures, e.g., traditional client-server, where a single server implements most of the software components (and thus functionality), while remote clients can access that server using simple communication means.
  - Decentralised architectures, e.g., peer-to-peer architectures in which all nodes more or less play equal roles.
  - Hybrid architectures, i.e., combining elements from centralized and decentralized architectures.

The final instantiation of the software components of a distributed system is what we call system architecture.

Software Architectural Styles

- A software architectural style is formulated in terms of components, the way that components are connected to each other, the data exchanged between components, and finally how these elements are jointly configured into a system.
- A variety of architectural styles result from the creation of systems configurations using components and connectors, including:
  - Layered architectural styles
  - Object-based architectural styles
  - Resource-centered architectural styles
  - Event-based architectural styles

A component is a modular unit with well-defined required and provided interfaces that is replaceable within its environment.

A connector is a mechanism that mediates communication, coordination, or cooperation among components. In other words, a connector allows for the flow of control and data between components.

The Layered Architectural Style

- In this architectural style, components are organized in a layered fashion where a component at layer Lj can make a downcall to a component at a lower-level layer Li (with i < j) and generally expects a response.
- The layers on the bottom provide a service to the layers on the top. The request flows from top to bottom, whereas the response is sent from bottom to top. In this approach, the calls always follow a predefined path.
  - Example: communication protocol stacks, where each layer implements one or several communication services allowing data to be sent from a destination to one or several targets.

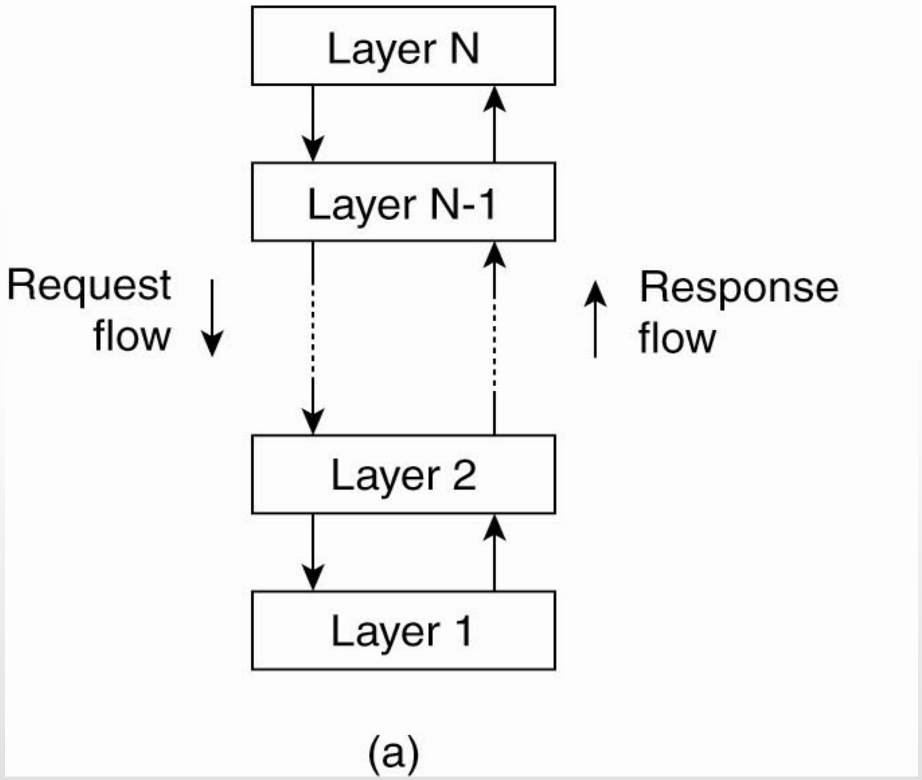


Figure 2.1(a) of book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

The Object-Based Architectural Style

- In this architectural style, each object corresponds to a component, and these components are connected through a procedure call mechanism, that can take place over a network, if the calling object is not on the same machine as the called object.
- Object-based architectures provide a natural way of encapsulating data and the operations that can be performed on that data into a single entity.
- And so, communication between objects happen as method invocations. These are generally called Remote Procedure Calls (RPC).
  - Some argue that object-based architectures form the foundation of encapsulating services into independent units. By clearly separating various services such that they can operate independently, the road toward service-oriented architectures (SOAs) are paved.

We will talk more about SOAs later in this course.

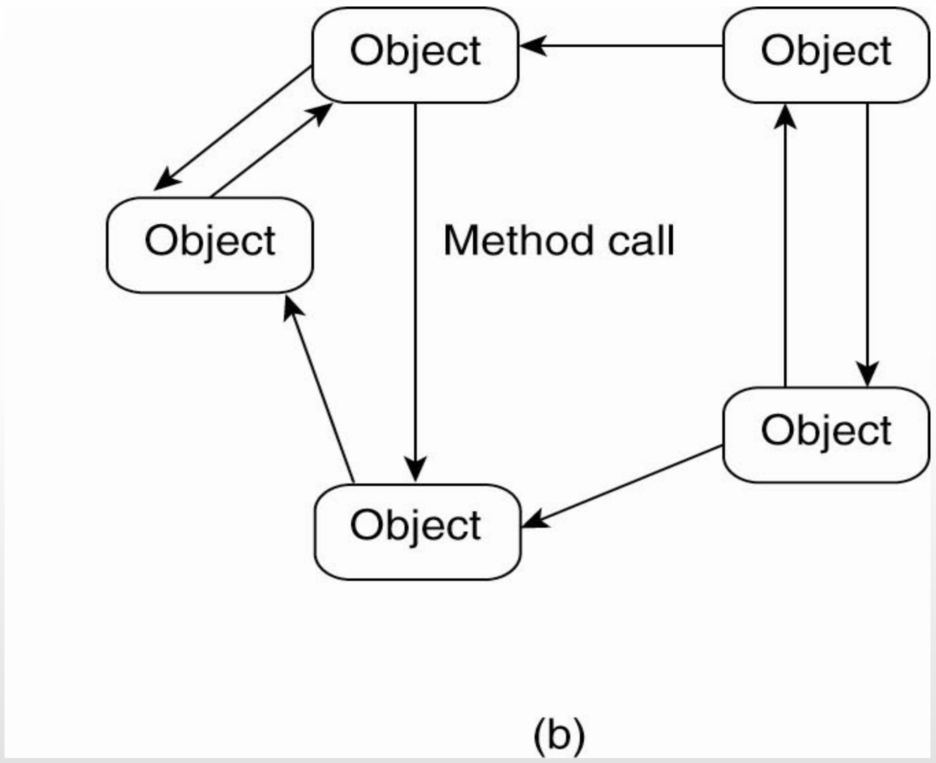


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

Resource-Centered Architectural Style

- In this architectural style, a distributed system is viewed as a huge collection of resources that are individually managed by components. Resources may be added or removed by (remote) applications, and likewise can be retrieved or modified.
- It is based on a data center, where the primary communication happens via a central data repository.
  - This approach has been widely adopted for the Web.

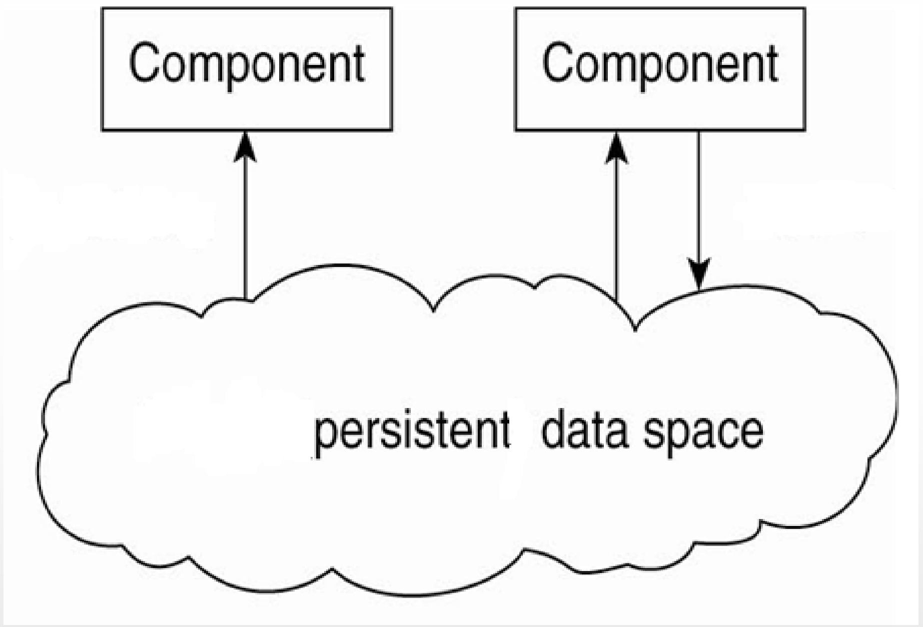


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

The Event-Based Architectural Style

- In this architectural style, processes running on the various components are both referentially decoupled and temporally coupled. In other words, one process does not explicitly know any other process, but for coordination to take place processes need to be *running at the same time*.
- In such scenarios, the only thing a process can do is **publish** a **notification** describing the occurrence of an event).
- Assuming that notifications come in all sorts and kinds, processes may **subscribe** to a specific kind of notification.

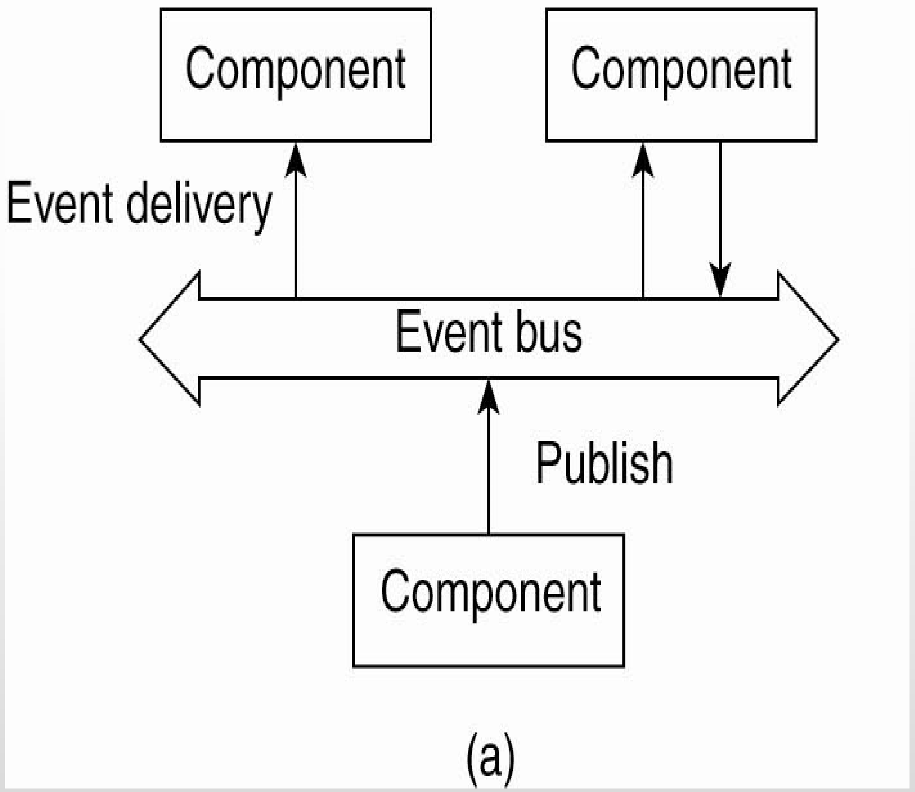
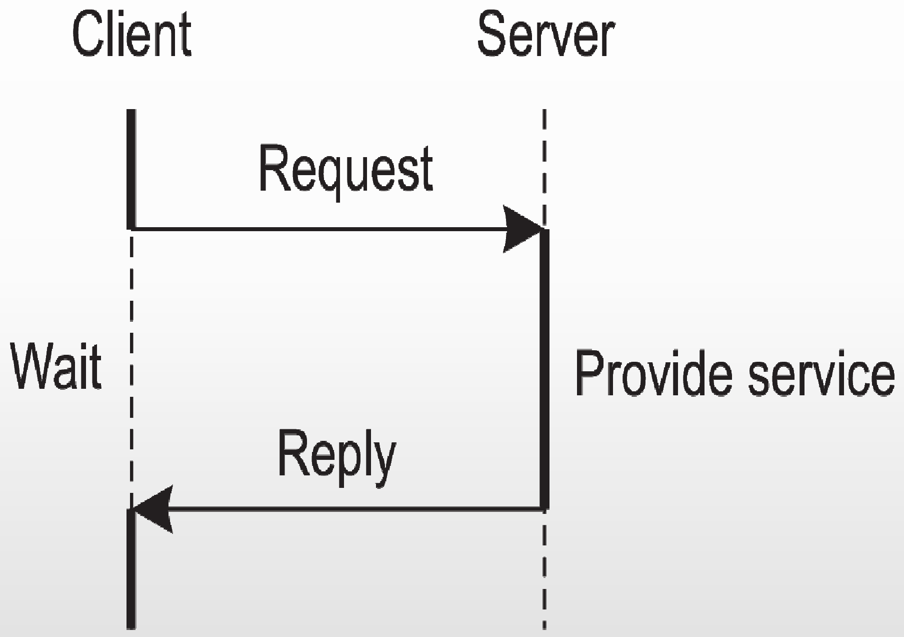


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

Centralised System Architectures: Examples

Simple Client-Server Architectures

- Processes in a distributed system are divided into two main groups: Clients and Servers.
- A server is a process implementing a specific service, for example, a file system service or a 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's reply.
- The client-server interaction is known as request-reply behaviour.



Request-reply behaviour shown in the form of a message sequence chart.

Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

Multi-Tiered Client-Server Architectures

- Typically presents three *logical* tiers.
- The distinction into three logical tiers suggests a number of possibilities for physically distributing a client-server application across several machines. However, the simplest organization is to have only two types of machines:
    - A client machine containing only (part of) the user-interface level.
    - A server machine containing the rest, i.e., the programs implementing the processing and data management functionalities.
- In the most typical organization, all functionality is handled by the server, while the client is essentially no more than a dumb terminal.

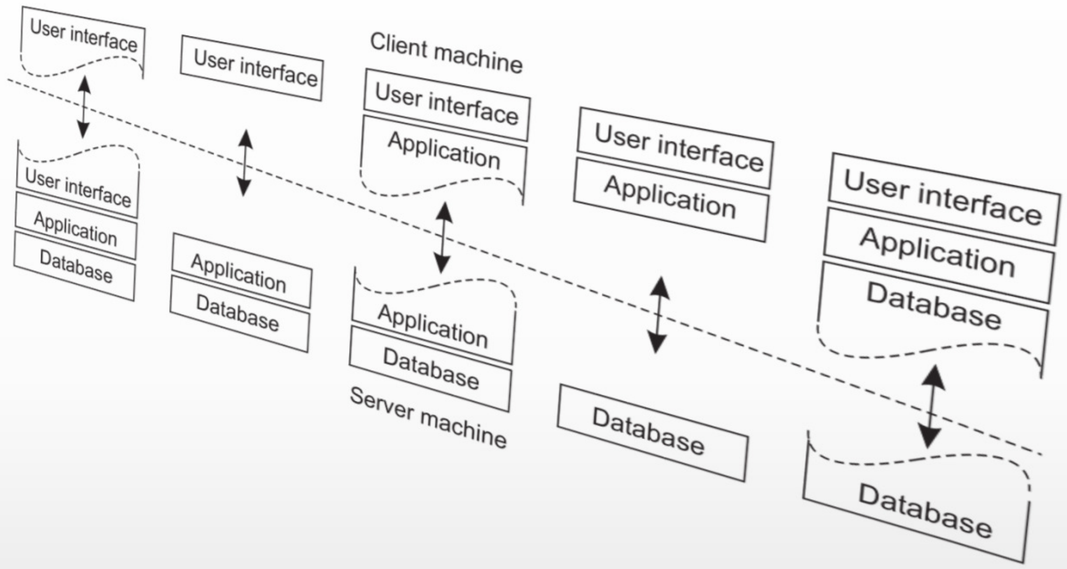


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

- The many other possibilities in functionality distribution between client and server machine are illustrated in the figure.
- Many distributed applications are divided into the three layers.
  - User interface layer,
  - Processing layer, and
  - Data layer.
- Thus, the main challenge to clients and servers is to distribute these layers across different machines.

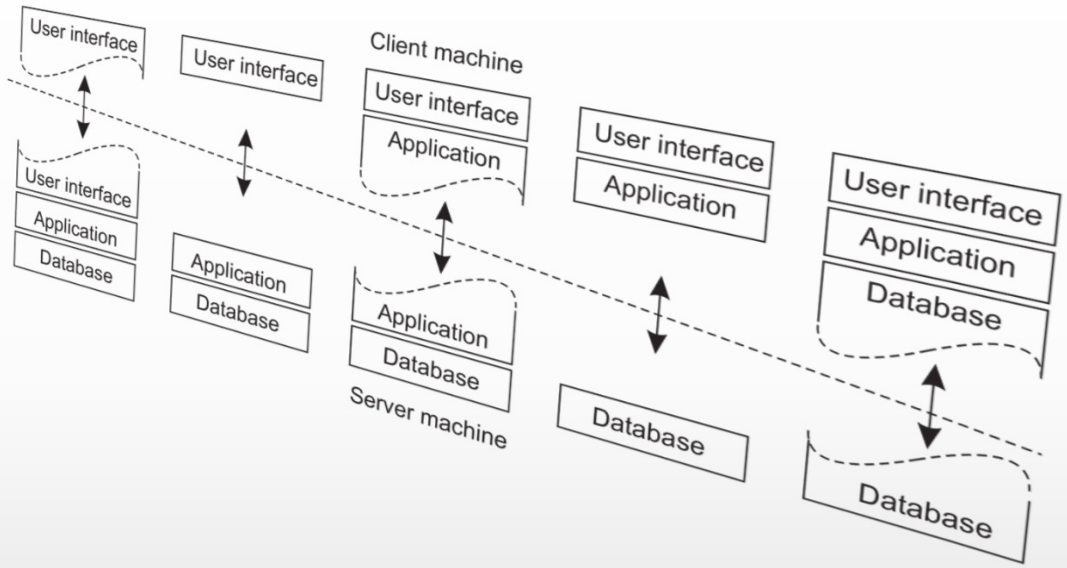
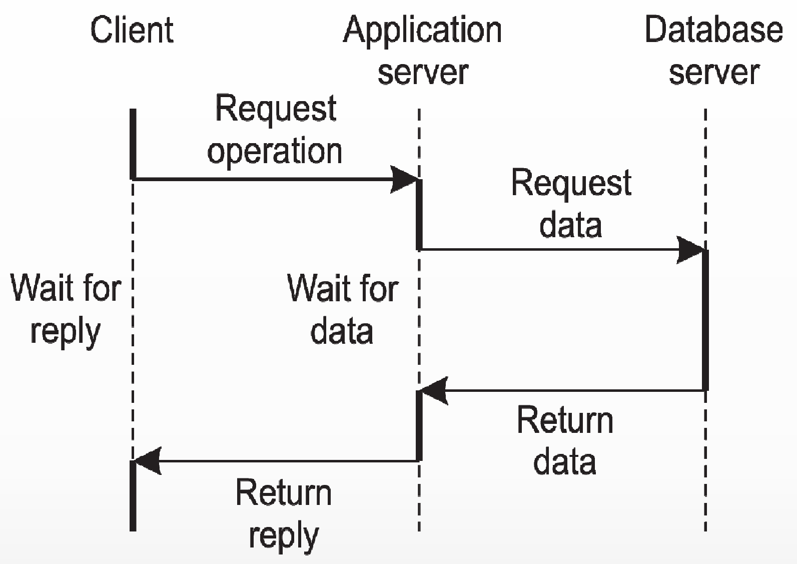


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

- A server may sometimes need to act as a client, as shown, typically leading to a physically three-tiered architecture.
- An example of use of this architecture is in the organisation of Web sites.



- Multi-tiered client-server architectures are a consequence of dividing distributed applications into a user interface, processing, and data-management components, where the different tiers correspond directly with the logical organization of applications. This type of distribution is called **vertical distribution**, achieved by placing logically different components on different machines.

Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'