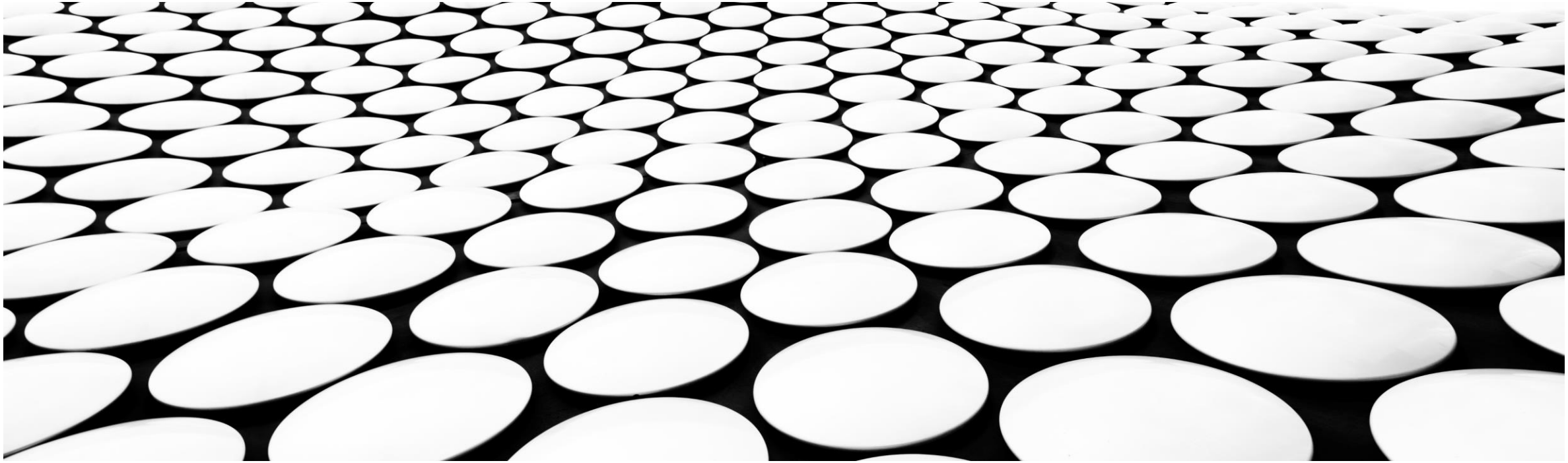
Architecture of Distributed Systems

COEN-317: Distributed Systems
Robert Bruce
Department of Computer Science and Engineering
Santa Clara University



Architecture of distributed systems

System architecture in distributed systems:

An implementation of a distributed system in regards to the specific software components and their interoperation [1].

[1] p. 55, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Architecture of distributed systems

How is architectural style related to distributed systems?

The system architecture is the implementation of a distributed system.

Why be concerned about architectural styles in distributed systems?

The architectural implementation could impact the distributed system in the following ways:

- performance (scalability)
- reliability (resilience)
- maintainability (dependency libraries and vendor lock-in)

Distributed systems examples

COMMUNICATION:

- Whatsapp
- WeChat
- Skype
- Zoom

FILE SHARING:

- BitTorrent

CLOUD COMPUTING:

- Amazon Web Services (AWS)

Types of distributed systems architecture

Centralized system:

- *client-server model* [1]

Decentralized system:

- *peer-to-peer model* [2]

Hybrid system (a combination of centralized and decentralized systems):

- *Edge-server systems* [3]
- *Collaborative systems* [4]

[1] pp. 76-79, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] pp. 80-88, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[3] pp. 90-91, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[4] pp. 91-92, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: client-server model

Client-server model:

- A model in which one or more client machines communicate with a centralized server machine.

Distributed systems architecture: client-server model

Client-server model:

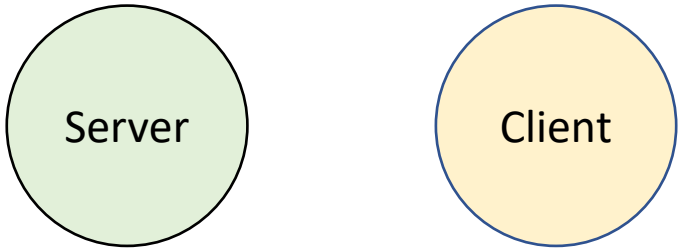
- A model in which one or more client machines communicate with a centralized server machine.
- Utilizes an interactive communication model called *request-reply behavior* [1].

[1] p. 76, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: client-server model

Client-server model:

- A model in which one or more client machines communicate with a centralized server machine.
- Utilizes an interactive communication model called *request-reply behavior* [1].

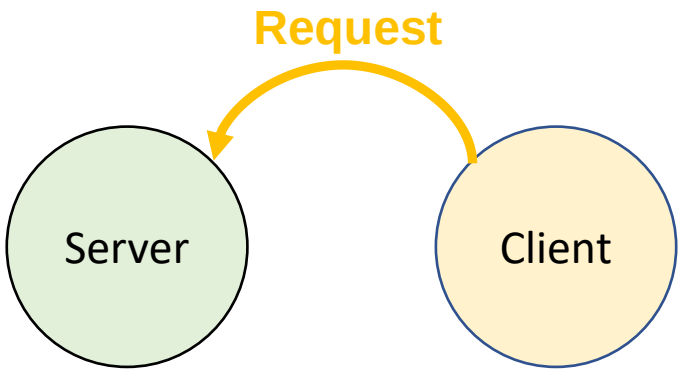


[1] p. 76, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: client-server model

Client-server model:

- A model in which one or more client machines communicate with a centralized server machine.
- Utilizes an interactive communication model called *request-reply behavior* [1].

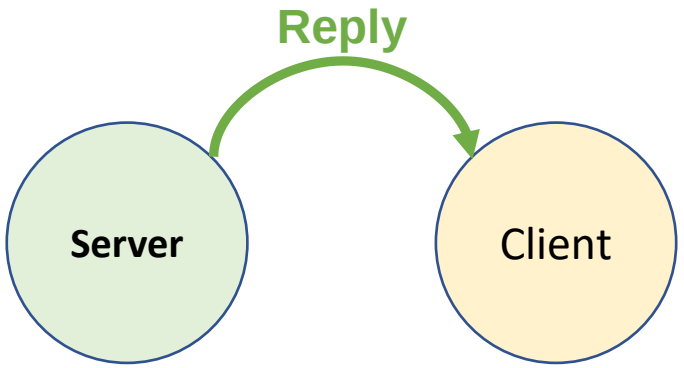


[1] p. 76, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: client-server model

Client-server model:

- A model in which one or more client machines communicate with a centralized server machine.
- Utilizes an interactive communication model called *request-reply behavior* [1].



[1] p. 76, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: client-server model

Client-server models:

- The functional components of a client-server model are installed on different machines.
- This is called *vertical distribution* [1].

[1] p. 81, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: client-server model

Client-server models:

- The functional components of a client-server model are installed on different machines.
- This is called *vertical distribution* [1].

Example of a client-server model:

- Network File System (NFS) [2].

[1] p. 81, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 94, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: peer-to-peer model

Peer-to-peer model:

- The client and server machines have logically equivalent functionality.
- Each machine serves a dualistic role as both a client and a server: a *servant* [1].
- This arrangement of client and server machines is called a *horizontal distribution* [1].
- Each machine processes a unique subset of the workload in a distributed system [1].

[1] p. 81, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: peer-to-peer model

Peer-to-peer systems can be structured or unstructured [1].

Structured peer-to-peer systems:

The nodes are arranged in a deterministic network topology [1].

Unstructured peer-to-peer systems:

The nodes resemble a random graph of nodes with no pre-defined path between nodes [1].

[1] pp. 82-84, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: edge-server system

Edge-server system:

- A hybrid architecture utilizing both centralized (client-server) and decentralized (peer-to-peer) architectures [1].
- Edge-servers are placed at the perimeter of an enterprise network [1].
- This technique reduces network latency for end-users (clients).

[1] p. 90, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: collaborative system

Collaborative system:

Bootstrapped as a centralized client-server model but transforms into a decentralized peer-to-peer model once a sufficient number of collaborative nodes join [1].

Example:

- BitTorrent is a peer-to-peer file sharing system. As more users participate in the BitTorrent sharing, each user's machine becomes a servant to download and share files with other users [1].

[1] p. 91, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Distributed systems architecture: Tips and tricks

Wrappers:

A means of encapsulating the components and details of specific interface(s) within a common middle layer interface [1].

Advantage:

Wrappers keep your distributed system architecture implementation agnostic. For example, a distributed system might have generic interface capable of simultaneously:

- Authenticating and connecting to an Amazon EC2 compute node.
- Authenticating and connecting to a Google Compute node.
- Authenticating and connecting to a Microsoft Azure node.

By making the interface generic, you avoid hardcoding the idiosyncratic details of each vendor's application programmer interface throughout your distributed system architecture. This saves you from vendor lock-in and massive refactoring of your distributed systems architecture.

[1] p. 72, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.