Middleware

software agents acting as an intermediary between different application components

Middleware layers

- Application
- Domain Specific middleware
- Common middleware
- Distribution middleware
- Host Infrastructure middleware
- Operating System & Protocols
- Hardware Devices

Infrastructure middleware

Encapsulates OS currency and inter-process communication (IPC) mechanisms

E.g. Java Packages, ACE

Distribution middleware

automates common network programming tasks

focuses on managing end-system resources

Object Request Broker (ORB)

E.g. COM+, Java RMI, CORBA

Common middleware services

defines higher-level domain-independent services, focuses on allocating, scheduling, and coordinating resources throughout a distributed system

Domain-specific middleware

provide services for specific vertical markets

E.g. telecommunications, e-commerce, health care, process automation, or avionics

Networked application design dimensions

- Communication dimensions
  - How networked applications interact
- Concurrency dimensions
  - How processes and threads are used
- Service dimensions
  - Duration and structure of services
- Configuration dimensions
  - How networked services are identified and the time that they are bound together to form complete applications

Communication Design Dimensions

- Connectionless vs Connection-oriented
- Synchronous vs Asynchronous
- Message-passing vs Shared memory

Connection dimension

- **Connectionless** → each message can be routed and delivered independently
- **Connection-oriented** → guarantee that a series of messages will arrive in a particular order at their destination

Connection-oriented Strategies

- Data framing strategies
- Connection multiplexing strategies

Synchronous vs Asynchronous

- Synchronous → requests and responses are exchanged in a lock-step sequence
- Asynchronous → stream requests from client to server without waiting for responses synchronously. Multiple requests can be transmitted before any responses arrive from a server.

Message Passing & Shared Memory

- ***Message Passing*** explicitly exchanges bytestream and record-oriented data
- ***Shared Memory*** allows multiple processes on the same or different hosts to access and exchange data as though it were local to the address space of each process