SFWR ENG 3A04: Software Design II

Dr. Ridha Khedri

Department of Computing and Software, McMaster University
Canada L8S 4L7, Hamilton, Ontario

Term 2

Acknowledgments: Material based on *Software Architecture Design* by Tao et al. (Chapter 10)

Outline of Part I

1. Model-View-Controller
   - MVC-I
   - MVC-II
2. Presentation-Abstraction-Control (PAC) Architecture

Outline of Part II

3. Overview
4. Client/Server
5. Multi-tier
6. Broker Architectural Style
7. Service-Oriented Architecture (SOA)
8. Enterprise Service Bus

Part I

Review of Previous Lecture

Part II

Today's Lecture

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Distributed Architecture Overview

- A distributed system is a collection of computers connected through a communication network
  - Data is distributed
  - Software is distributed
  - Users are distributed

- The sub-systems or components within a distributed system communicate with each other via
  - message passing
  - remote procedure call
  - remote method invocation
  - etc.

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Distributed Architecture Overview

Two important issues for designing a distributed system are:

- Topology: the way in which entities connect with each other
- Mode: the method by which they communicate with each other
  - Synchronous
  - Asynchronous
  - message driven
  - callback
  - event-driven

Distributed Architecture

Client/Server

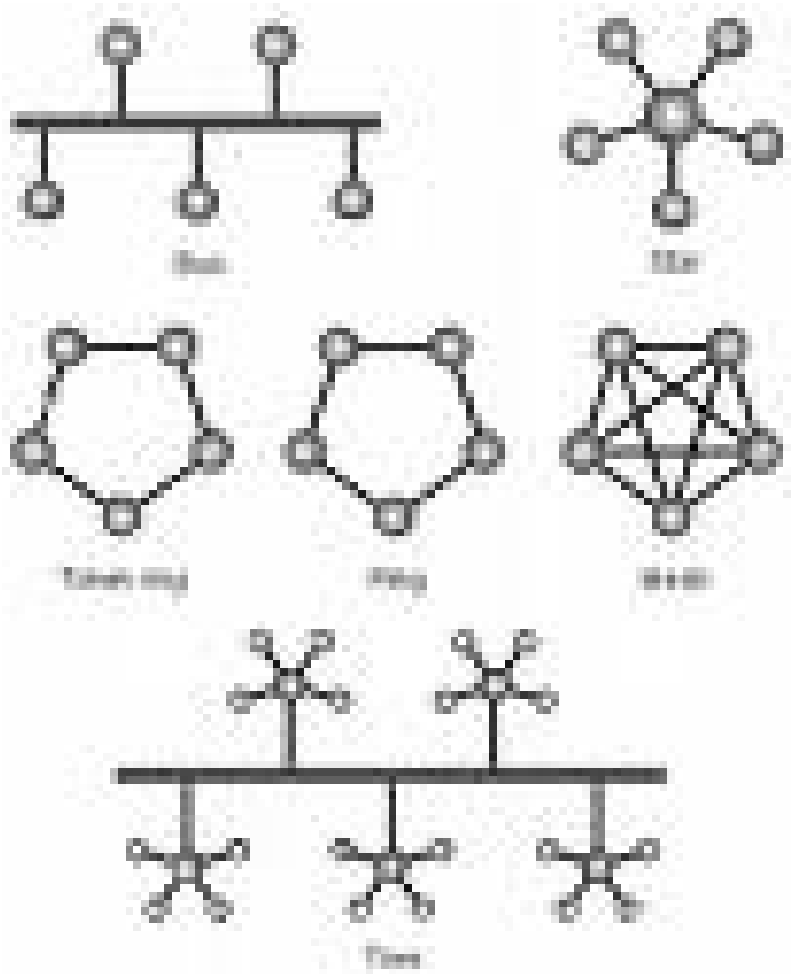


Figure: Examples of network topologies

Distributed Architecture Overview

- A distributed system can be modeled as a
  - Client/server architecture
  - Broker architecture
  - Service-Oriented Architecture (SOA)
- The important features of a distributed architecture include
  - its service location transparency
  - service reliability and availability

Distributed Architecture Client/Server

- The client-server model is the most common distributed system
- It is based on two communicating subsystems (usually running on different processors)
  - Client issues a request to the second process server
  - Server process receives the request, carries it out, and sends a reply to the client

Distributed Architecture

Client/Server

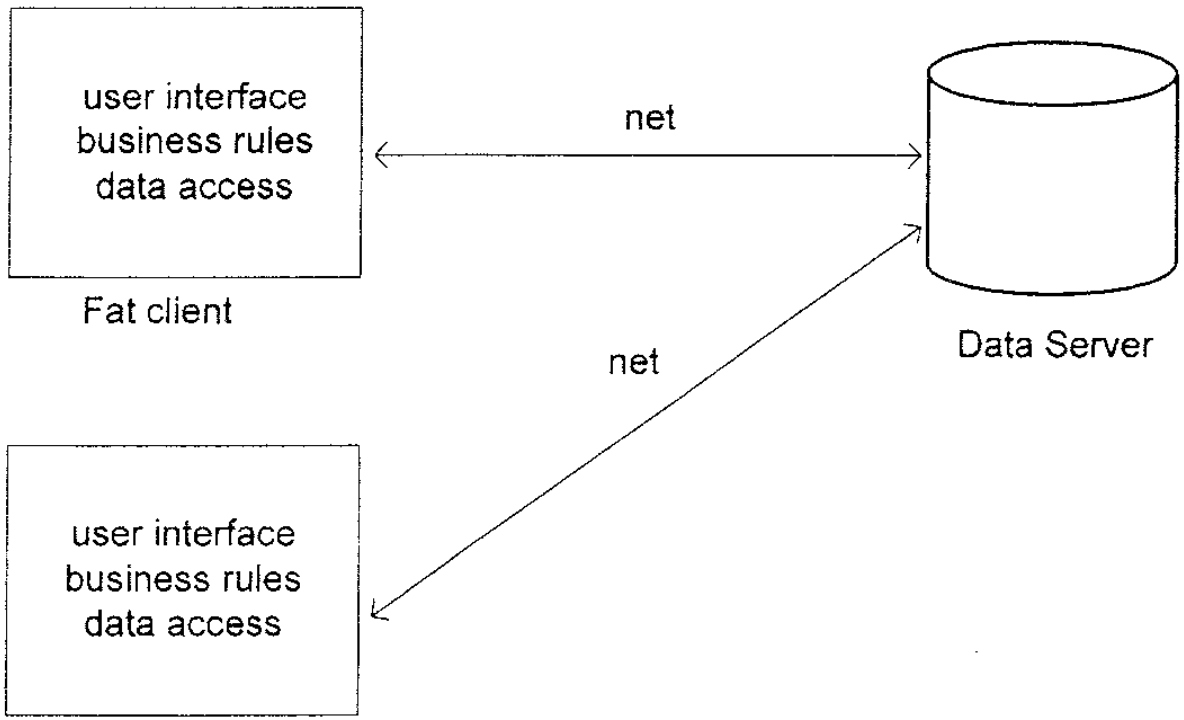


Figure: Two tier client/server architecture

Distributed Architecture Client/Server

- Advantages
  - Separation of responsibilities such as user interface presentation and business logic processing
  - Reusability of server components
- Disadvantages
  - Lack of heterogeneous infrastructure to deal with the requirement changes
  - Security complications
  - Server availability and reliability
  - Testability and scalability
  - Fat-clients/ Thin-clients (depends on the application)

Distributed Architecture Multi-tier

- The front tier in a multi-tier architecture is the user interface presentation tier
- The middle-tier(s) take(s) care of business logic, application decision, and execution
- The back-end tier usually works on database management, or on a (virtual) machine
- The advantages of multi-tier over the two-tier architecture are
  - the enhancement of reusability
  - scalability by the middle tier
    - The middle tier can also provide multi-threading supports for scalability
  - Multi-tier architecture also reduces the traffic on the network
- Disadvantage: complex testability

Distributed Architecture

Multi-tier

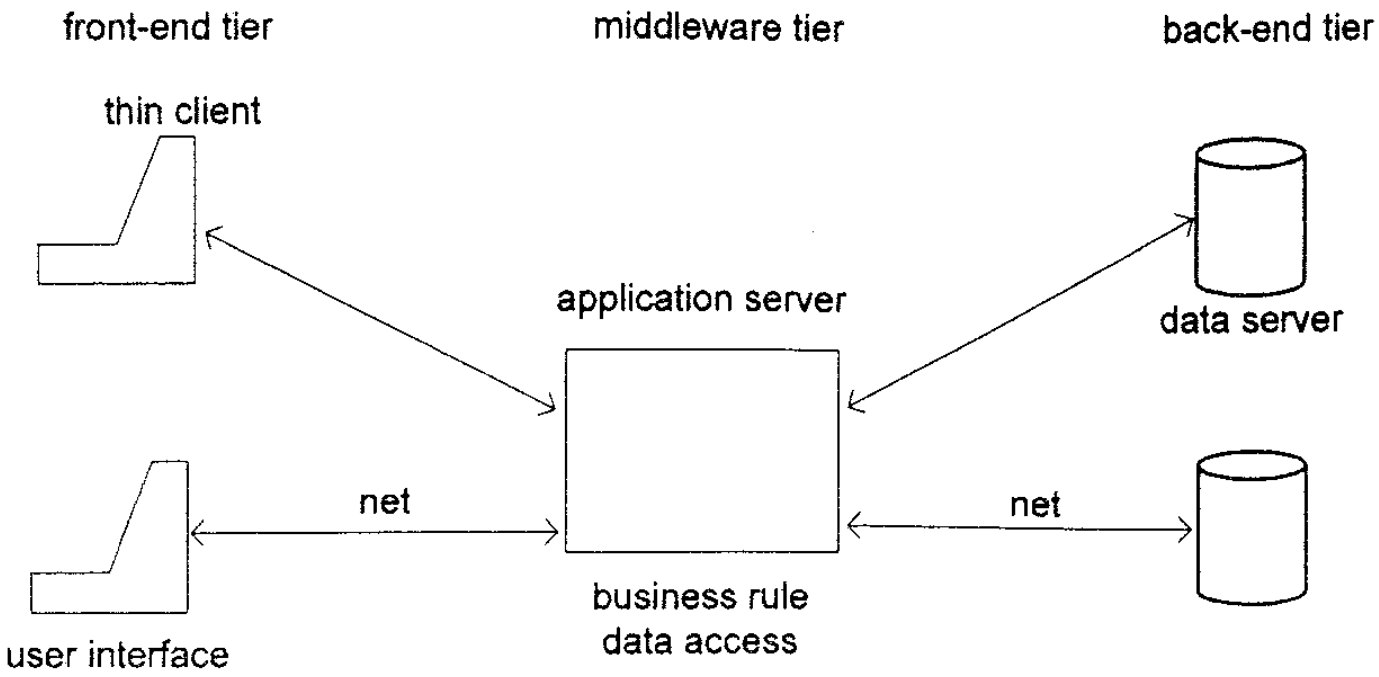


Figure: Three tier architecture

Distributed Architecture
Broker Architectural Style

- The broker architecture is a middleware architecture widely used in distributed computing
- It is suitable for distributed computing that coordinates and facilitates communication
  - brokering the service requests
  - locating proper server
  - forwarding and dispatching requests
  - sending responses or exceptions back to clients
- It can be used to structure distributed software systems with decoupled components that interact by remote service invocations
- The most important quality of this architecture = better decoupling between clients and servers

Distributed Architecture
Broker Architectural Style

- Servers make their services available to their clients by registering and publishing their interfaces with the broker
- Clients can request the services of servers from the broker statically or dynamically by look-up
- A broker acts as a policeman in a busy intersection who controls and interacts with the client components and server components
- The connection between clients and servers is maintained by the broker

Broker Architectural Style

- A distributed client can access distributed services simply by calling a remote method of a remote object
- This concept is similar to unix Remote Procedure Call (RPC) and Java Remote Method Invocation (RMI)
- The clients can dynamically invoke the remote methods even if the interfaces of the remote objects are not available at the compilation time

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Distributed Architecture
Broker Architectural Style

- Client has a direct connection to its client-proxy
- Server has direct connection to its server-proxy
- The proxy talks to the mediator-broker
- The proxy is a well known pattern for hiding low-level detailed communication processing
  - It intercepts the client's request
  - gets all arguments
  - packets it
  - marshals (streamlines) and formats the package in the format of communication protocol
  - sends it to the broker
- A broker system is also called proxy-based system

Distributed Architecture
Broker Architectural Style

Sub-components of a broker architecture

- Broker
- Stub (client-side proxy): It mediates between client and broker
- Skeleton (server-side proxy)
  - It is statically generated by the service interface compilation and then deployed to the server side
  - It receives the requests, unpacks the requests, unmarshals the method arguments, and calls the appropriate service
  - It also marshals results from the sever before it sends it back to the client
- Bridges (Optional)
  - Used to hide implementation details when two brokers interoperate
  - Can connect two different networks based on different communication protocols

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Distributed Architecture

Broker Architectural Style

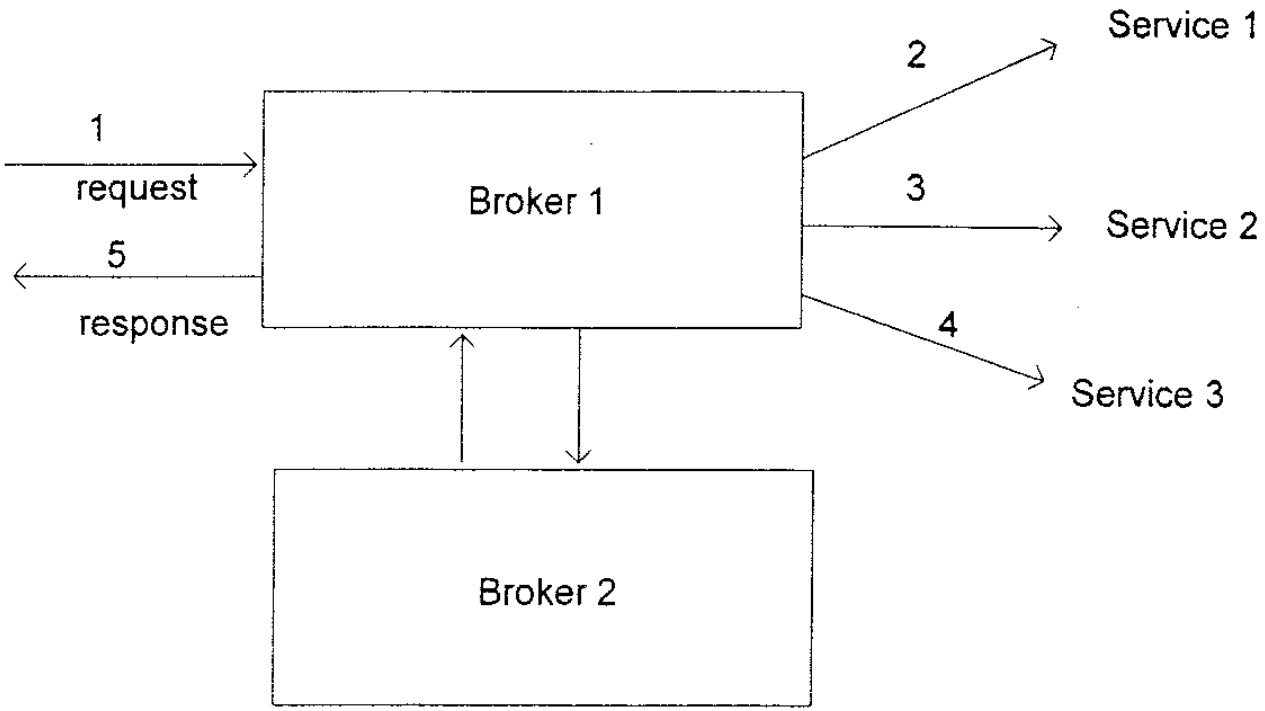


Figure: Broker model

Distributed Architecture

Broker Architectural Style

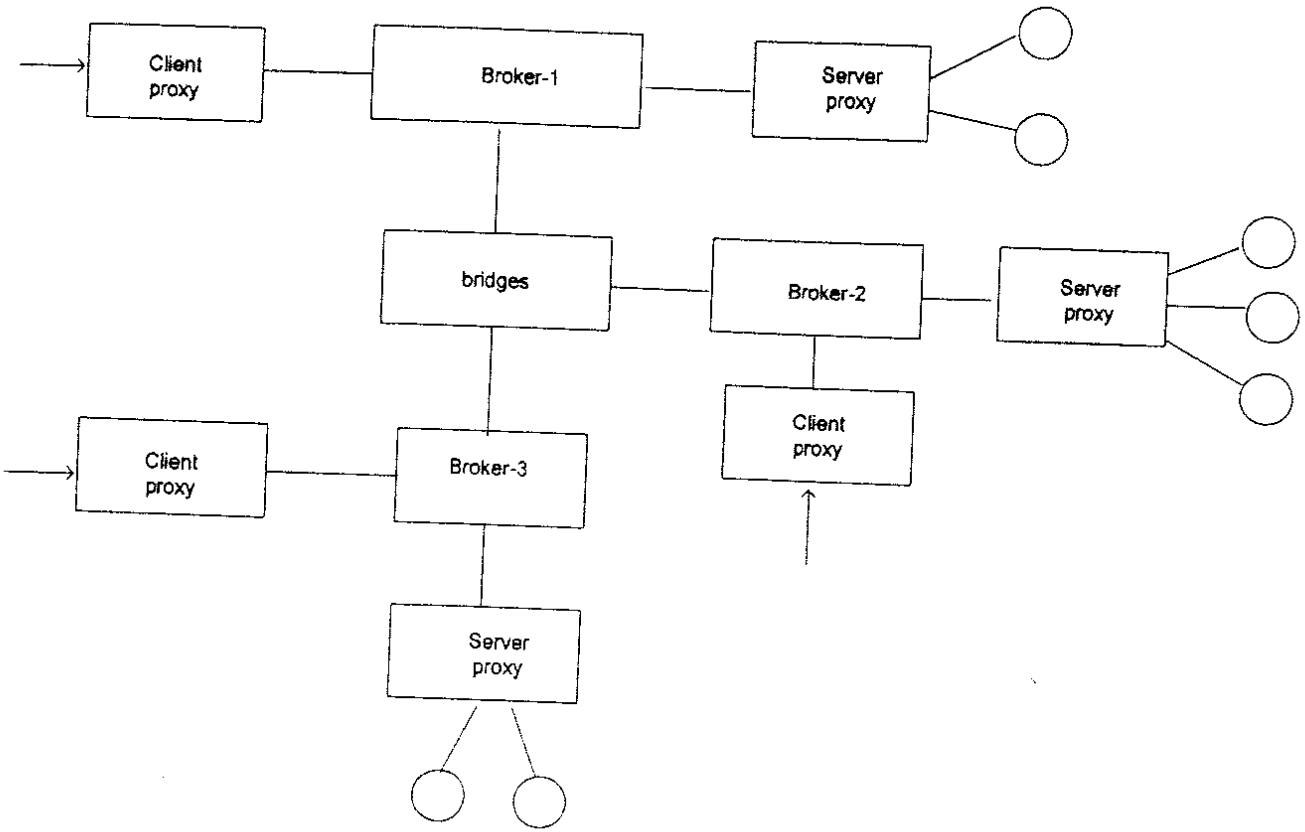


Figure: Connected brokers with client/server proxy

Distributed Architecture

Broker Architectural Style

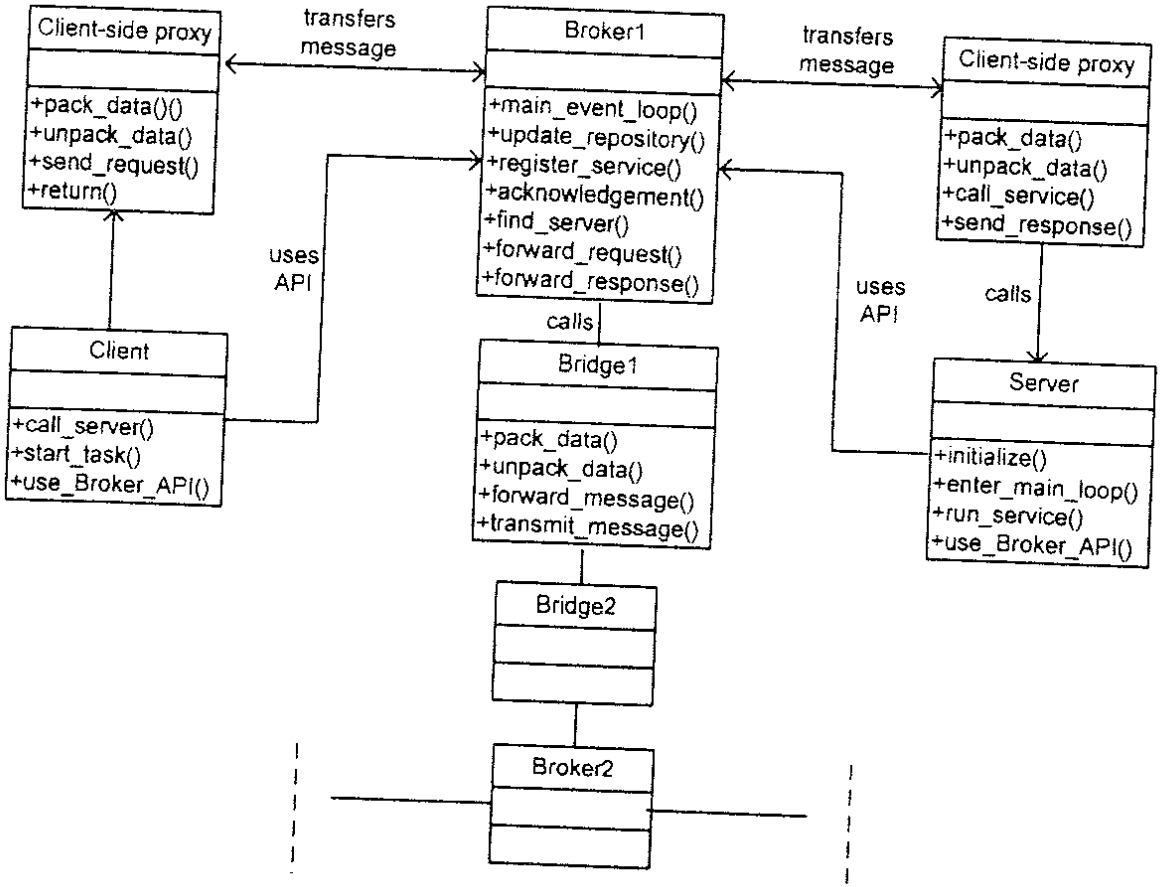


Figure: Class diagram for broker architecture

Distributed Architecture

Broker Architectural Style

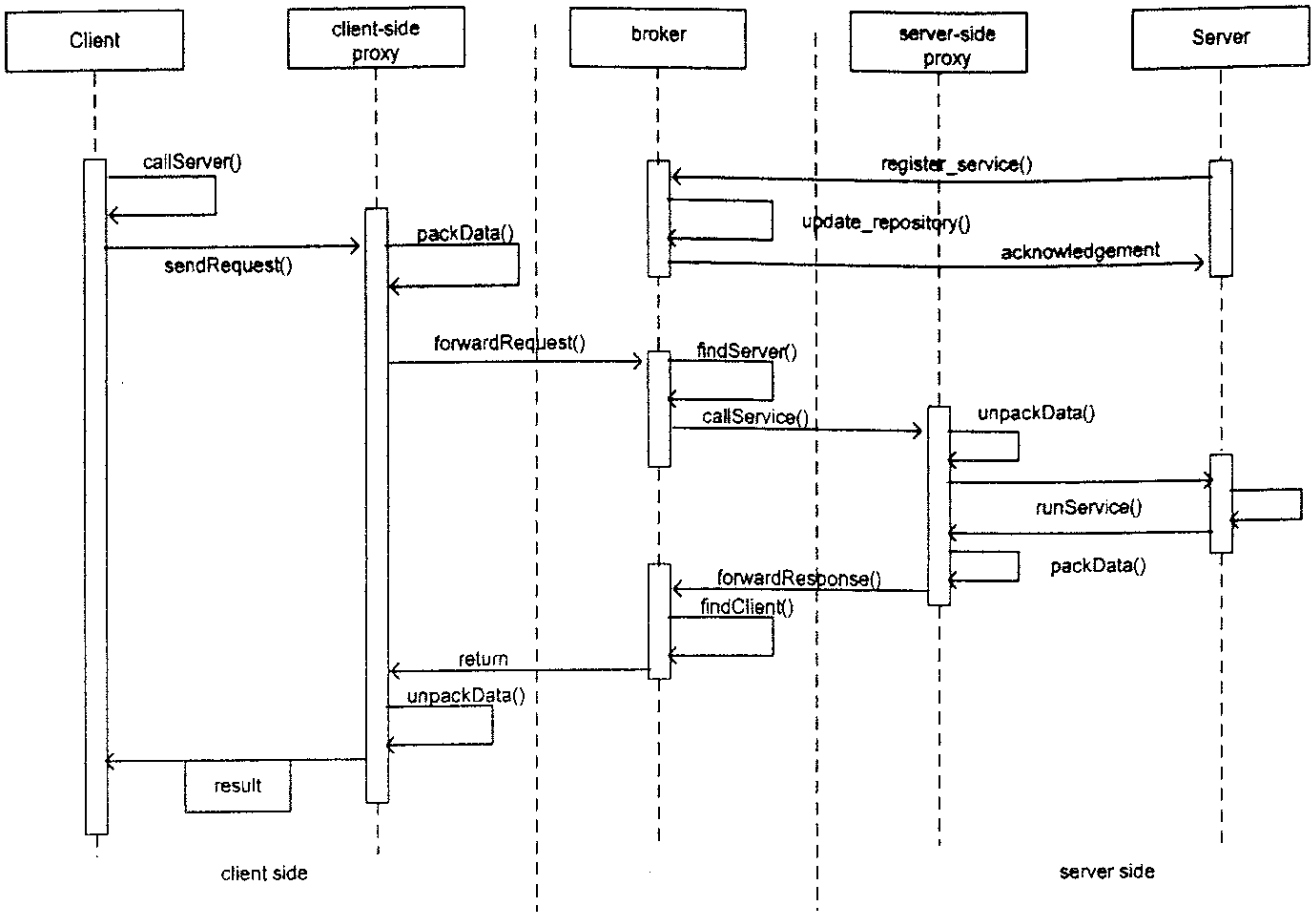


Figure: Sequence diagram for broker architecture

Distributed Architecture
Broker Architectural Style

- Advantages
  - Server component implementation and location transparency
  - Changeability and extensibility
  - Simplicity for clients to access server and server portability
  - Interoperability via broker bridges
  - Reusability
  - Feasibility of runtime changes of server components (add or remove server components)

- Disadvantages
  - Inefficiency due to the overhead of proxies
  - Low fault-tolerance
  - Difficulty in testing

Distributed Architecture
Service-Oriented Architecture (SOA)

- A service is a business functionality that is
  - well-defined and self-contained
  - independent from other services
  - published and available to be used via an interface

- SOA services can be reused extensively regardless of whether they are based on new or legacy applications
- Loose coupling of service-orientation architecture provides a great flexibility for enterprises to make use of all available service resourses

- The connections between services are conducted by common and universal message oriented protocols such as the SOAP Web service protocol
- A connection can be established statically or dynamically

Distributed Architecture

Service-Oriented Architecture (SOA)

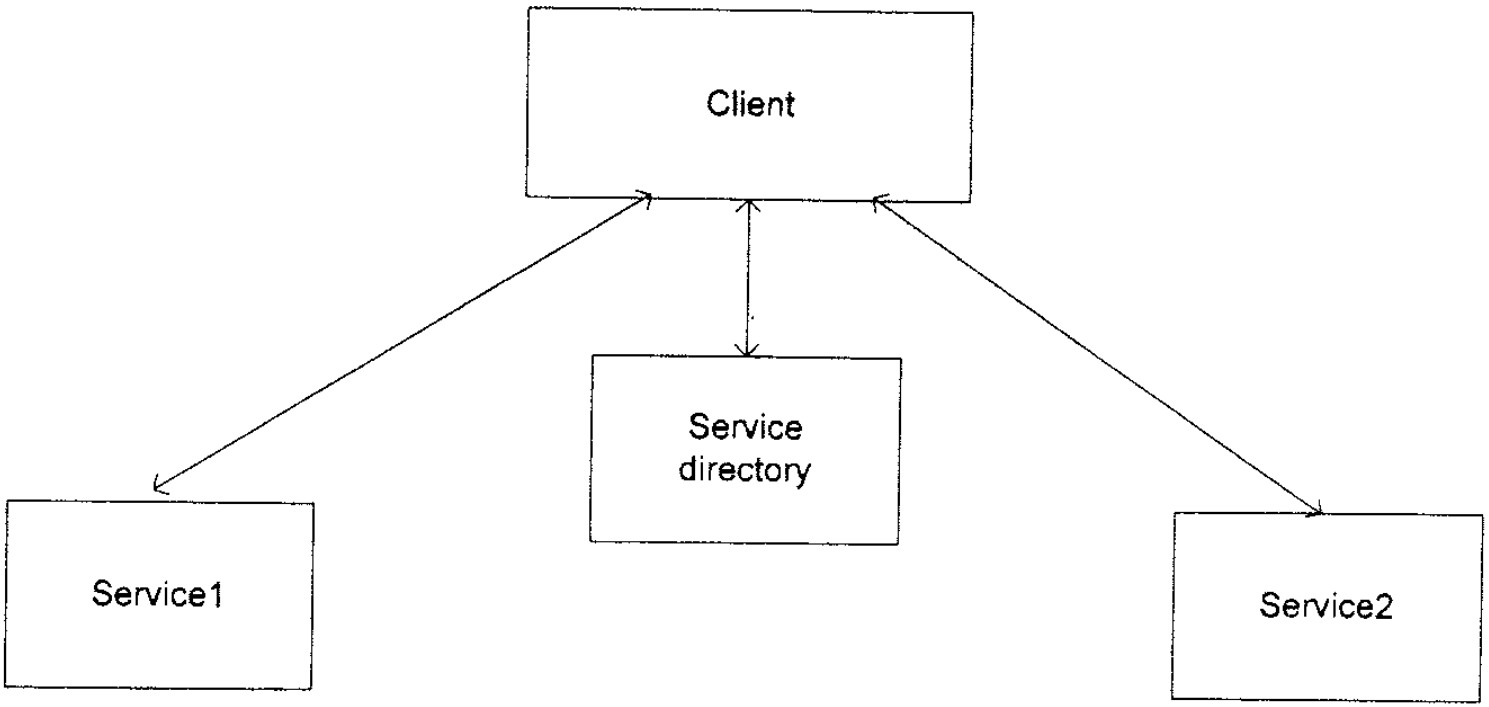


Figure: Client with services and service directory

Distributed Architecture
Service-Oriented Architecture (SOA)

- A service-oriented application might make use of many available services
- For that one needs a flow control language
  - allows specifying the sequence and logical order of the business executions based on the business logic
- Some services can be reused by other applications that they are not originally designed for
- We can build a new service out of existing services
  - aggregation: extends one endpoint of a service to make a new interface of the new service
  - containment structure: has one interface that wraps all used services
- Services can be recursively constructed to satisfy a more complex business needs (through aggr. and cont.)

Distributed Architecture

Service-Oriented Architecture (SOA)

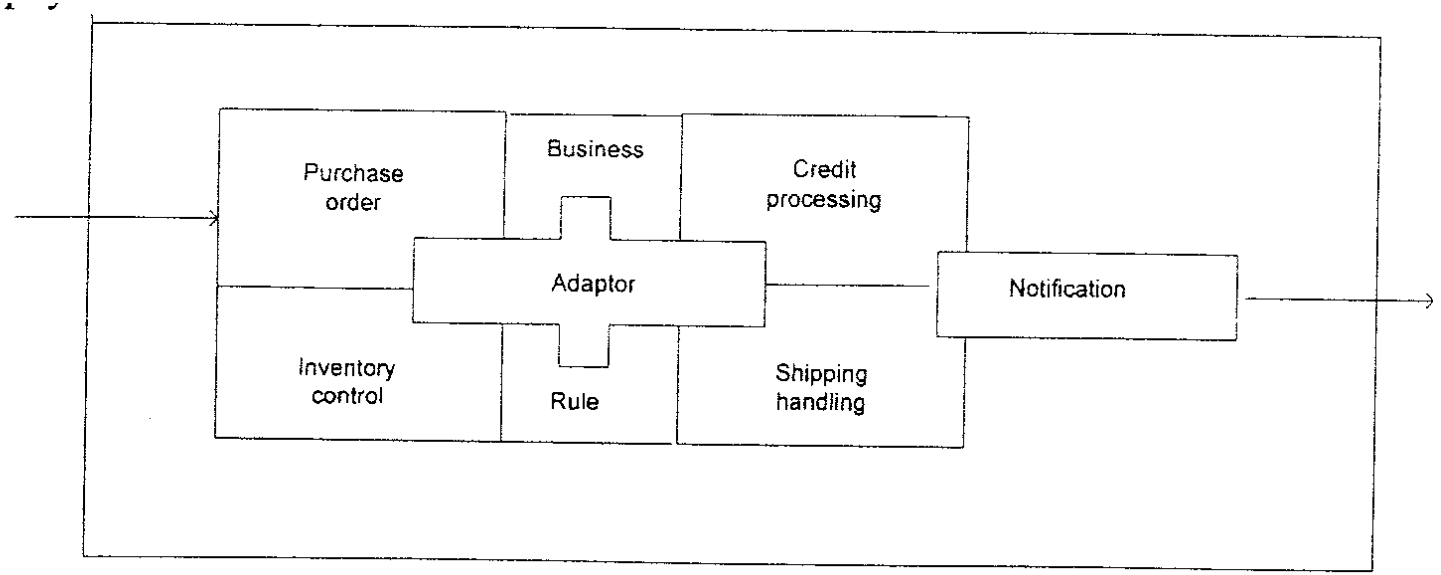


Figure: Service composition

Distributed Architecture

Service-Oriented Architecture (SOA)

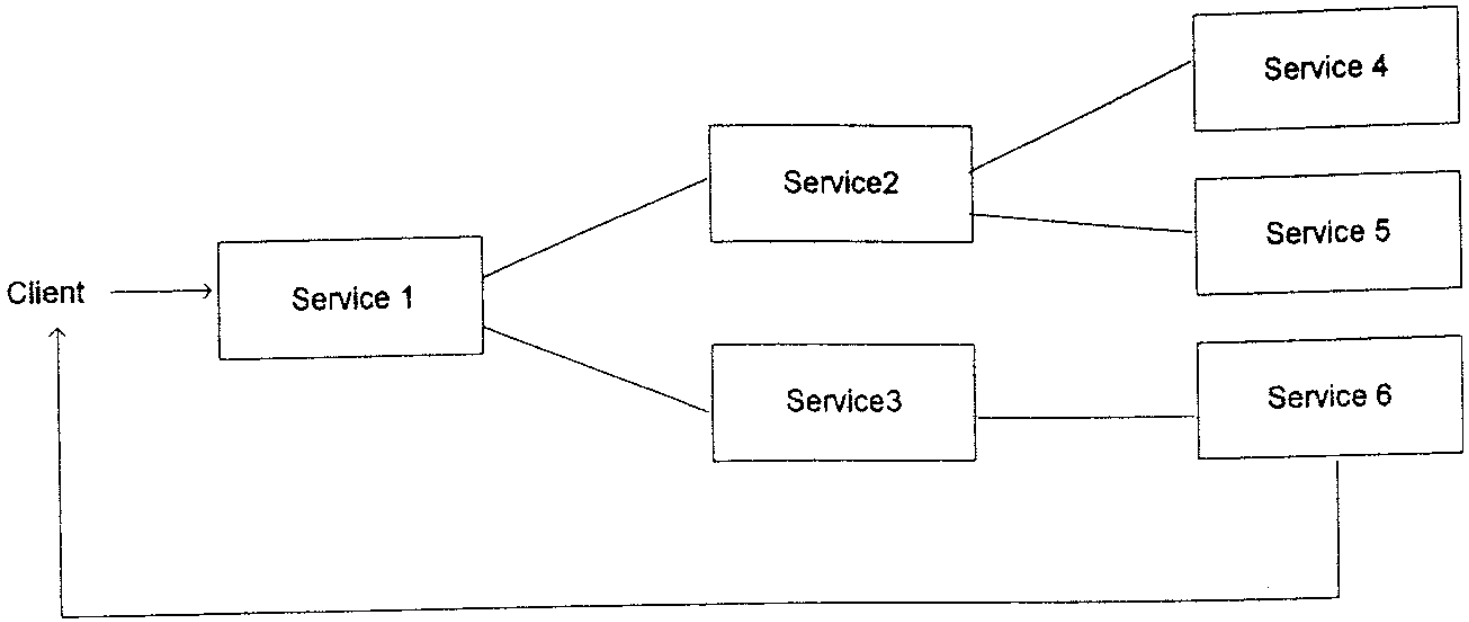


Figure: Service reuse

Service-Oriented Architecture (SOA)

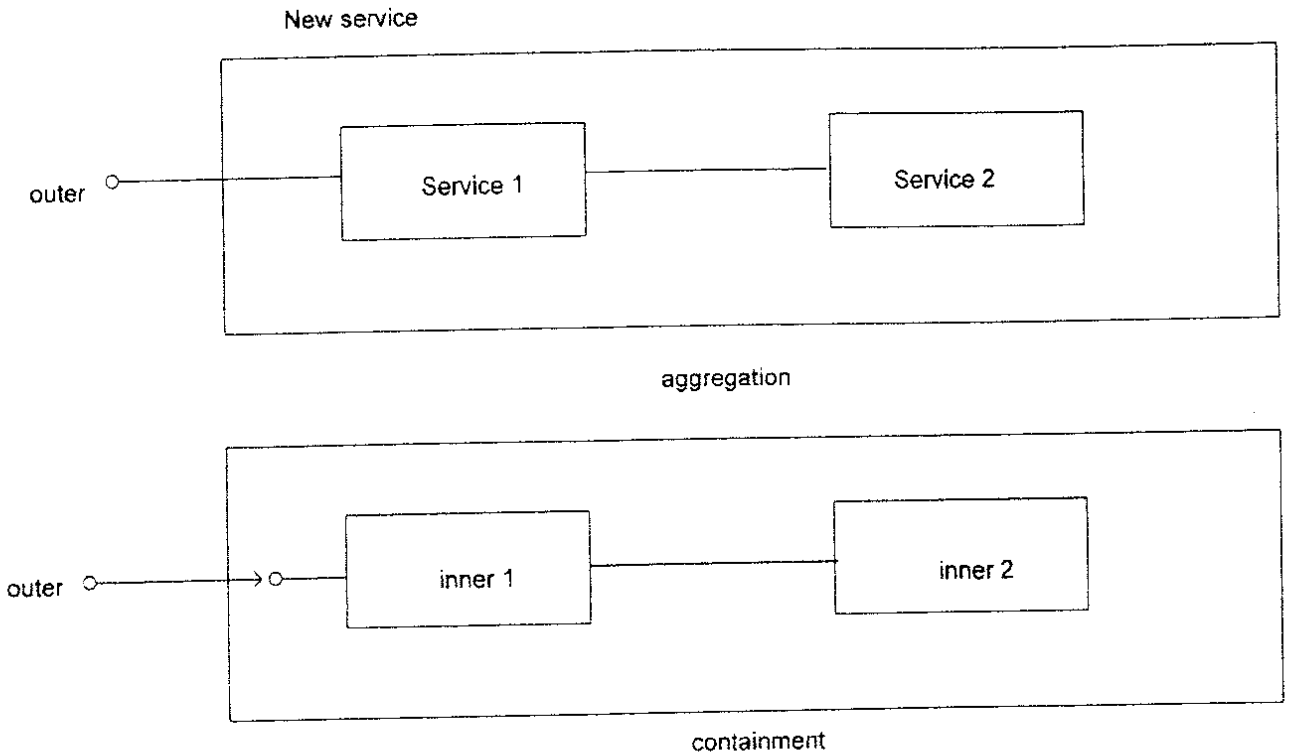


Figure: Service composition model

SFWR ENG 3A04: Software Design II
Dr. R. Khedri
Overview
Client/Server
Multi-tier
Broker Architectural Style
Service-Oriented Architecture (SOA)
Enterprise Service Bus

Distributed Architecture

Service-Oriented Architecture (SOA)

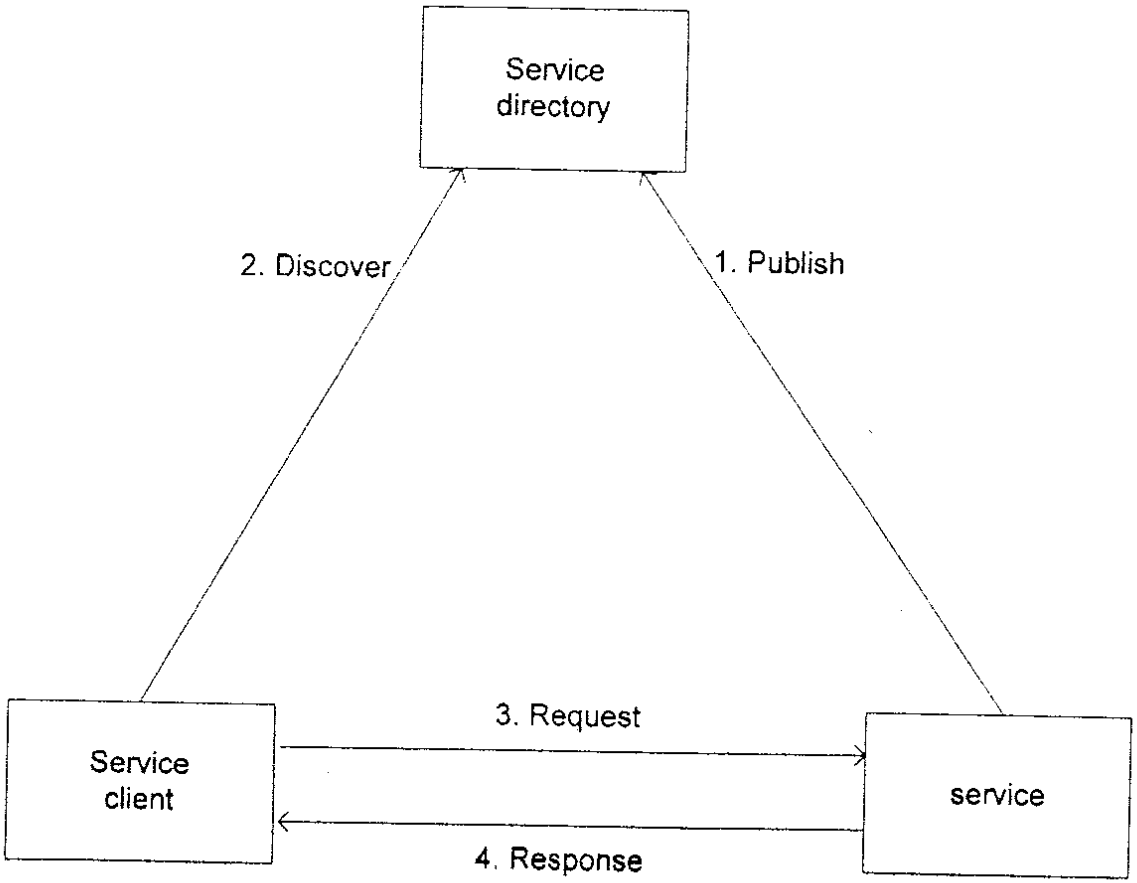


Figure: Service working model

Service-Oriented Architecture (SOA)

Advantages of SOA

- Loosely-coupled connection
- Each service component is independent from other services due to the stateless service feature
- Interoperability: Technically any client or any service can access other services regardless of their platform, technology, vendors, or language implementations
- Reusability: Any service can be reused by any other services and service is developed to be reused as well
- Scalability: Loosely coupled services make themselves easy to scale

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Enterprise Service Bus (ESB)

Motivation

- Enterprise information systems have dramatically changed and progressed
- A lot of changes to enterprise information systems is brought by
  - Corporation merger
  - Internal control and regulatory compliance control for personal information protection
- Sometimes, a quick change is needed by a new business that ought to be added because of tough competition

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Enterprise Service Bus (ESB)

Motivation (Continued)

- These changes lead to having various systems exist in silos of different departments that are operated and managed separately (some kind of total independence of these systems)
- The cost for the overall system maintenance increase significantly
- Also we observe a diminished enterprise management efficiency
- Enterprise information systems require highly adaptable and highly flexible system architecture

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Enterprise Service Bus (ESB)

Motivation (Continued)

- On the top of these requirements, there is a strong demand for low-cost and quick system development
- These overall requirements can be satisfied by
  - Utilising the existing systems as well as the Web services and Web API's that are available to the public
  - Minimising newly developed parts
  - The proper combination of the elements above
- To achieve this, SOA and its underlying middleware are necessary

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Enterprise Service Bus (ESB)

Enterprise Service Bus (ESB): What is it?

- Enterprise Service Bus (ESB) is the SOA integration technology that provides a unified architecture for high reusability
- It is an environment designed to foster sophisticated interconnectivity between services
- It establishes an intermediate layer of processing that can help overcome common problems associated with reliability, scalability, and communications disparity

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Distributed Architecture
Enterprise Service Bus (ESB)

What is it?

- It is an architectural layer that connects a bunch of software capabilities (referred to as services) to high level controlling entities (referred to as choreographers)

- The aim is that the services capabilities can work in concert to accomplish tasks

Enterprise Service Bus (ESB)

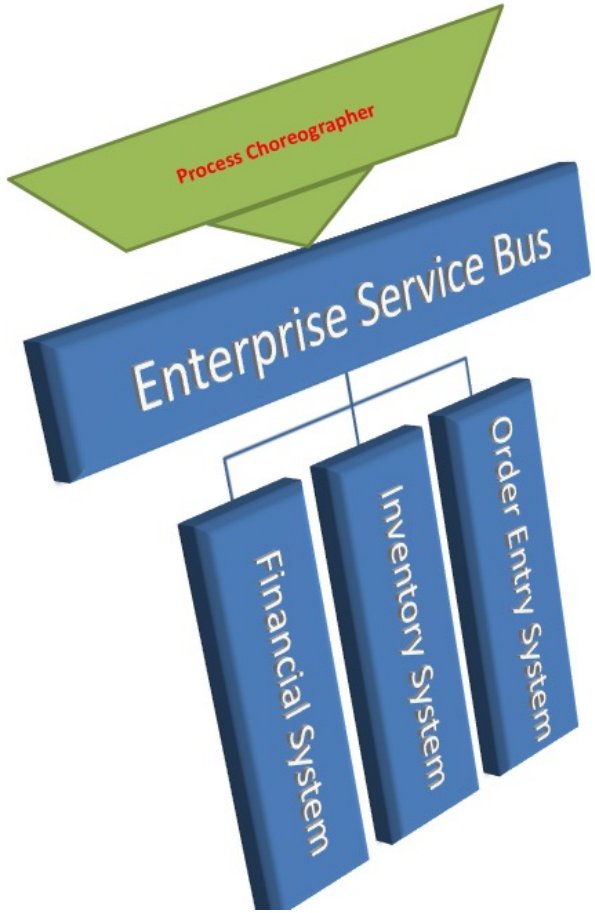


Figure: Simple Representation of an ESB

Enterprise Service Bus (ESB)

What is it?

- It is comprised of the co-existent application of
  - Asynchronous Queuing: It is about how a service and its consumers accommodate isolated failures and avoid unnecessarily locking resources
  - Event-Driven Messaging: The consumer establishes itself as a subscriber of the service. The service, in turn, automatically issues notifications of relevant events to this and any of its subscribers
  - Intermediate Routing: Various types of intermediary routing logic can be incorporated to create message paths based on message content or runtime factors

SFWR ENG 3A04: Software Design II

Dr. R. Khedri

Overview

Client/Server

Multi-tier

Broker Architectural Style

Service-Oriented Architecture (SOA)

Enterprise Service Bus

Enterprise Service Bus (ESB)

- Composition (CONTINUED)
  - Policy Centralization: Policies that apply to multiple services can introduce redundancy and inconsistency within service logic and contracts. Global or domain-specific policies can be isolated and applied to multiple services.
  - Reliable Messaging: Reliable messaging is the concept of communicating messages across an unreliable infrastructure whilst being able to make certain guarantees about the successful transmission of the messages

Enterprise Service Bus (ESB)

- Composition (CONTINUED)
  - Rules Centralization: The same business rules may apply across different business services, leading to redundancy and governance challenges. The storage and management of business rules are positioned within a dedicated architectural extension from where they can be centrally accessed and maintained.
  - Service Broker

Distributed Architecture

Enterprise Service Bus (ESB)

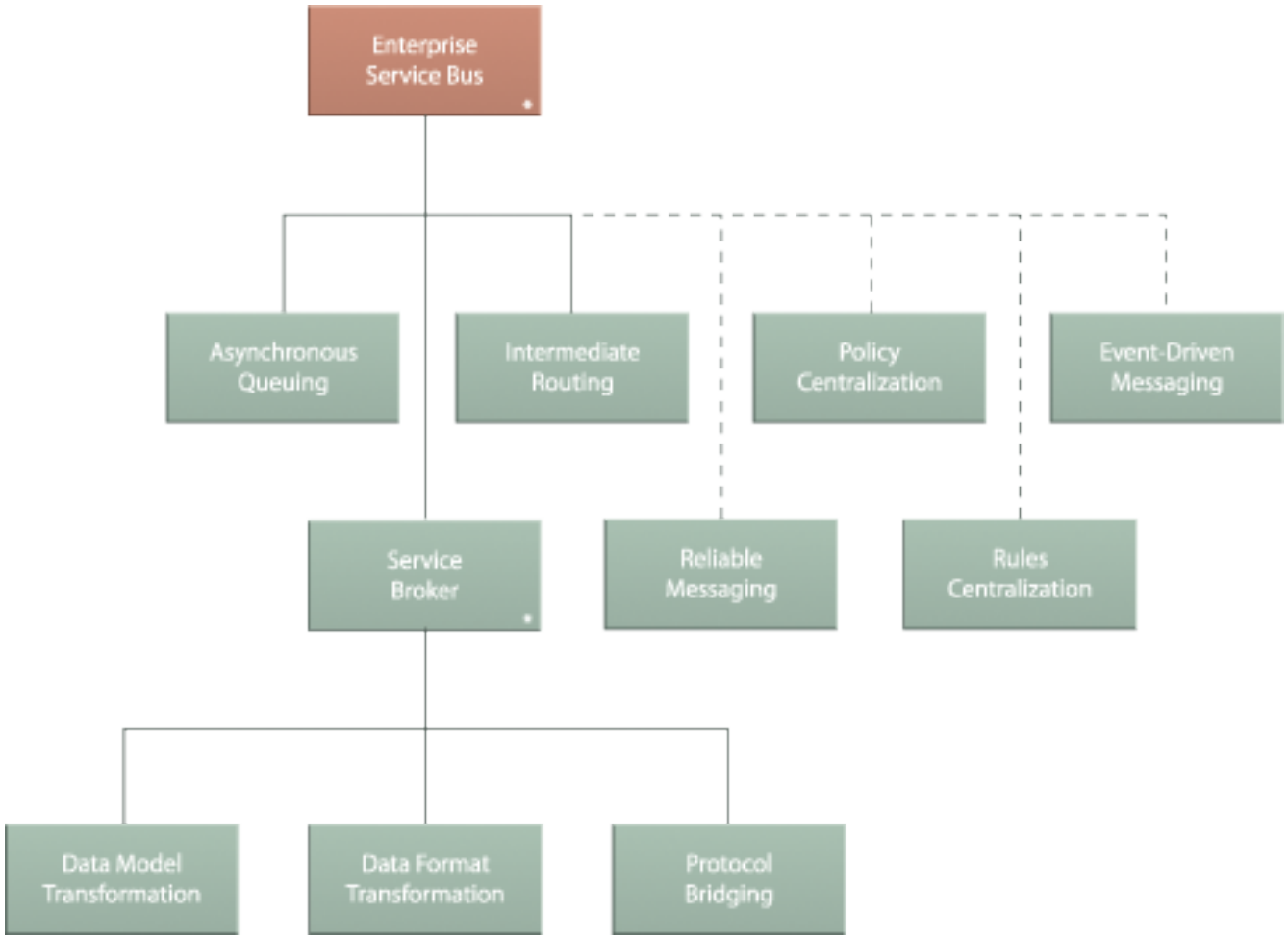
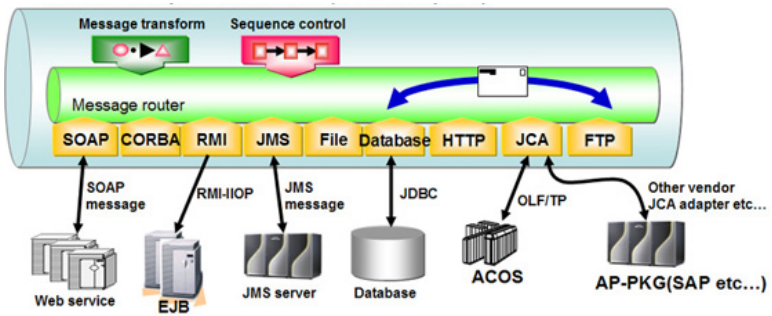


Figure: The co-existent applications that comprise an ESB

Distributed Architecture

Enterprise Service Bus (ESB)



Aknowledgment: Borrowed from https://www.nec.com/en/global/prod/webotx/en/soa.html

Figure: Example of system mediation using ESB

Overview

Client/Server

Multi-tier

Broker
Architectural Style

Service-Oriented
Architecture (SOA)

Enterprise Service
Bus