DC Experiment No.10

Aim: Write a case study on CORBA in enabling communication within distributed file systems.

Theory: CORBA (Common Object Request Broker Architecture), a standard by the Object Management Group (OMG), facilitates communication between distributed objects across heterogeneous systems, enabling applications written in different languages to interact seamlessly. The central component of CORBA is the **Object Request Broker (ORB)**, which acts as a middleware that manages communication and data exchange between distributed objects.

CORBA provides the following key features:

- **Language and Platform Independence:** Allows objects written in different programming languages to interact.
- **Location Transparency:** Clients can invoke methods on remote objects as if they were local.
- **Interoperability:** Supports communication across systems with different architectures.
- **Scalability and Reusability:** Facilitates development of scalable and reusable distributed components.

CORBA uses **Interface Definition Language (IDL)** to define object interfaces in a language-neutral manner, which are then mapped to specific programming languages (e.g., Java, C++).

How CORBA is useful in distributed file system?

A Distributed File System (DFS) allows users to access and manage files stored on multiple servers as if they were on a single local system. In such systems, components like file servers, clients, metadata managers, and security services may be distributed across various locations and built using different technologies.

CORBA (Common Object Request Broker Architecture) is a middleware standard developed by the Object Management Group (OMG) that supports communication between distributed objects in a language- and platform-independent manner. It allows objects in a DFS—such as file handlers, directory services, and access controllers—to communicate regardless of their underlying implementation.

Case Study Example: Airline Reservation System:

Imagine a distributed airline reservation system where different components (e.g., flight search, ticket booking, seat allocation) are implemented on different machines and use different technologies. CORBA can be used to facilitate communication between these components, allowing them to interact as if they were local objects, regardless of their underlying implementation or location. In summary, CORBA provides a robust and flexible framework for building distributed applications, enabling communication between heterogeneous systems and components, and supporting a wide range of applications, including airline reservation systems, e-commerce backends, and financial systems.

System Architecture in CORBA:

1. Flight Search Engine
   - Technology: Java
   - Platform: Linux
   - Role: Allows users to search available flights by source, destination, date, time, airline, etc.
   - CORBA Role: Exposes a SearchService object registered with the ORB, allowing other modules to invoke it remotely.

2. Ticket Booking System
   - Technology: C++
   - Platform: Windows
   - Role: Handles the ticket booking logic including fare calculation, PNR generation, and validation.
   - CORBA Role: Offers BookingService object via CORBA for client or flight search engine to interact with.

3. Seat Allocation Module
   - Technology: Python
   - Platform: Cloud (e.g., AWS, GCP)
   - Role: Manages seat assignment, layout updates, and availability tracking.
   - CORBA Role: Publishes SeatManager object with the ORB for other services to call.

4. User Interface / Web Portal
   - Technology: Could be web-based or mobile app
   - Role: Frontend for user interaction

- CORBA Role: Acts as a client calling CORBA services via stubs generated from IDL definitions.

Workflow Example (Booking a Flight):

1) User searches for flights via UI → SearchService (Java) is called via CORBA.
2) User selects a flight → UI invokes BookingService (C++) to begin booking.
3) BookingService checks seat availability by calling SeatManager (Python) via CORBA.
4) If available, booking is confirmed, and SeatManager is updated to mark the seat as reserved.
5) Ticket is generated, and a confirmation is sent to the user.

All these operations happen via CORBA, with the Object Request Broker (ORB) handling:

- Object discovery
- Communication protocols
- Data marshaling/unmarshaling
-
- Method invocations

IDL (Interface Definition Language): Used to define operations like searchFlights(), bookTicket(), allocateSeat().Enables stub/skeleton generation in multiple languages.

ORB (Object Request Broker): Responsible for locating services and handling method calls between clients and objects. Examples include ORB implementations like TAO, OmniORB, or JacORB.

Naming Service / Trader Service: Helps clients dynamically find services (e.g., find the nearest BookingService).

Advantages of Using CORBA in Airline Systems:

1. **Technology Independence:** Allows using the best language/platform for each module.
2. **Interoperability:** CORBA enables seamless communication between components developed independently.
3. **Maintainability:** Each service/module can be maintained, upgraded, or redeployed without affecting others.

4. **Reusability:** CORBA objects like BookingService or SeatManager can be reused in mobile apps, travel agency portals, etc.
5. **Scalability:** Modules can be horizontally scaled (e.g., multiple SeatManager instances in the cloud).
6. **Loose Coupling:** Modules only depend on interface definitions (IDL), not internal implementations.
7. **Dynamic Service Discovery:** CORBA supports naming and trading services, which allow components to discover each other at runtime.
8. **Quality of Service (QoS) Management:** CORBA can prioritize certain operations (e.g., emergency flight bookings or VIP customers) over others, helping maintain service-level agreements (SLAs) and improve customer satisfaction.
9. **Support for Event and Notification Services:** CORBA provides publish/subscribe mechanisms (via the CORBA Event and Notification Services), allowing modules to react to events like booking confirmations, cancellations, or flight delays without polling.
10. **Extensibility:** The system can be extended easily by **adding new services** (e.g., meal selection, in-flight entertainment booking) without disrupting the existing architecture.

Conclusion: In this case study, how CORBA is useful in DFS was studied.