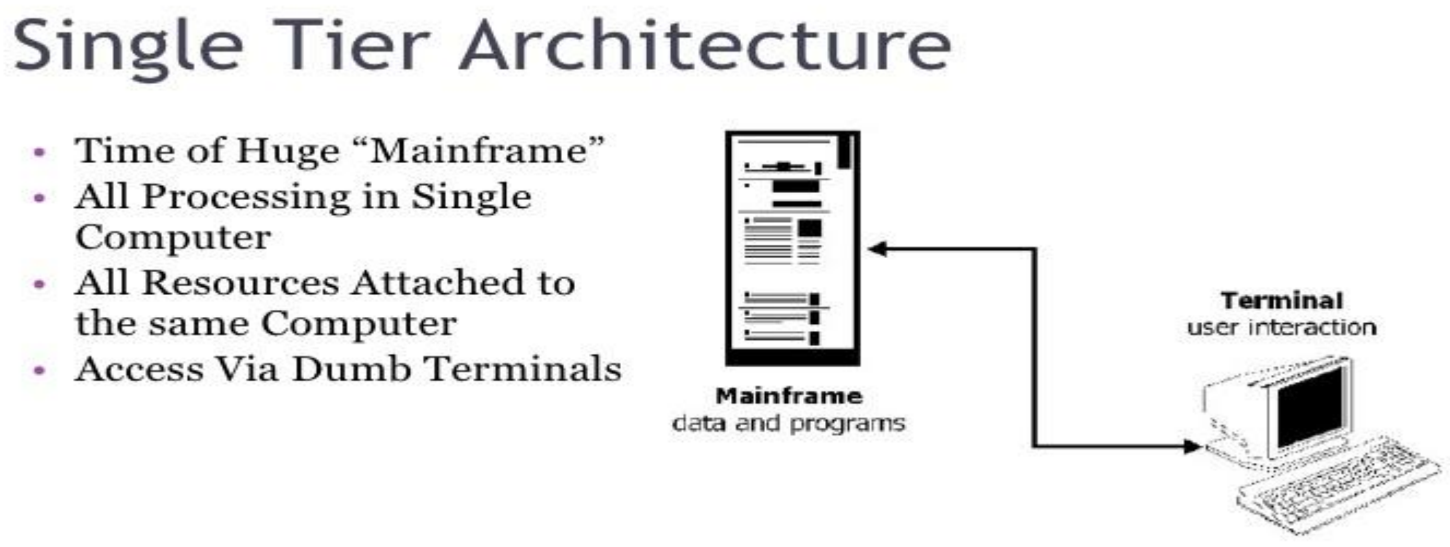
SOA Vs Past Architectures

UNIT-III

SOA Vs C/S Architecture

- Originally monolithic mainframe
- Single-tier client server architecture
- Bulky mainframe back-ends with thin clients
- Supports synchronous and asynchronous

Single-tier Architecture



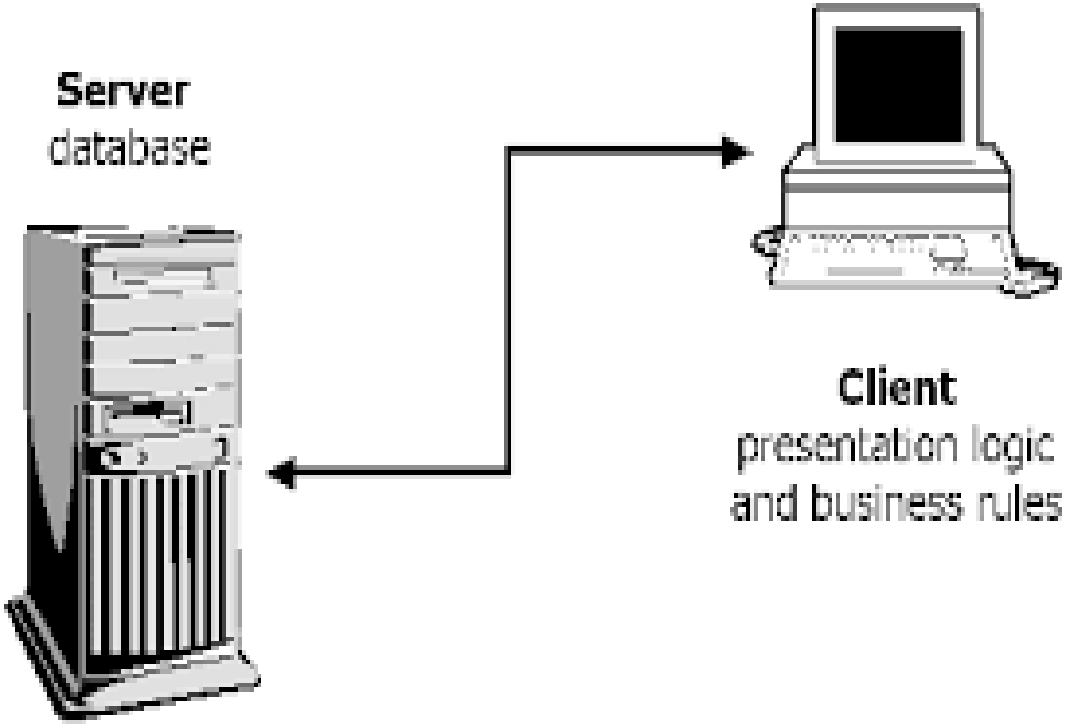
Single Tier Architecture

- Time of Huge "Mainframe"
- All Processing in Single Computer
- All Resources Attached to the same Computer
- Access Via Dumb Terminals

Two-tier client server architecture

- Variation - Two-tier client server architecture
- Fat client – logic +processing units on individual workstation
- Supports GUI
- Ex. Multiple fat clients each has 1 database connection to server

Two-tier C/S Architecture



SOA Vs C/S - Application Logic

- **C/S Architecture:**
- Majority on client side – business rules distributed
- Or business rules stored in DB as procedures and triggers
- **SOA:**
- Presentation layer - Any s/w capable of exchanging SOAP message
- Commonly requestors also service
- Server side – no DB triggers, follows SO design principles (statelessness, interoperability, composability, resusability) partitioning logic into autonomous units
- Solution Agnotisic
- Promote reuse, loose coupling across applications

SOA Vs C/S – Application Processing

- **C/S:**
- Bulk processing at client – 20% DB server used, so performance bottleneck
- Individual DB connection, persistent and expensive due to proprietary
- Client side – fully stateful, eatsup resources
- **SOA:**
- Highly distributed
- Choices to position and deploy services optimally
- Enterprise solutions – multiple servers, no fixed processing
- Communication synchronous /asynchronous
- Intelligent message – supports stateless, autonomous services

SOA Vs C/S – Security

- **C/S:**
- Centralised server – secured, DB – manage user level security, client – part of UI, selective users possible, OS level system login
- **SOA:**
- Distributed, so single point authentication not possible
- Security becomes complexity
- Multiple technologies involved – WS-Security framework involved

SOA Vs C/S – Administration

- **C/S:**
- Maintenance cost more – due to distribution of s/w across clients and updation
- Issues spans on clients and servers – different s/w & h/w on client side, DB enlargement – to meet server-side demand
- **SOA:**
- Requestors not subject to client side challenges
- Scalability in DB servers and applications can be introduced

SOA Vs Distributed Internet Architecture

- Distributing application logic over multiple components reduces centralized deployment problems
- Sever side components share resource pools, concurrent usage is possible – but increases complexity
- Replacing c/s with RPC using CORBA / DCOM faces same problem – resource use, persistent connections
- After the advent of WWW distributed internet architecture , replaces RPC with web server and HTTP
- Becomes defacto for custom developed enterprise solutions

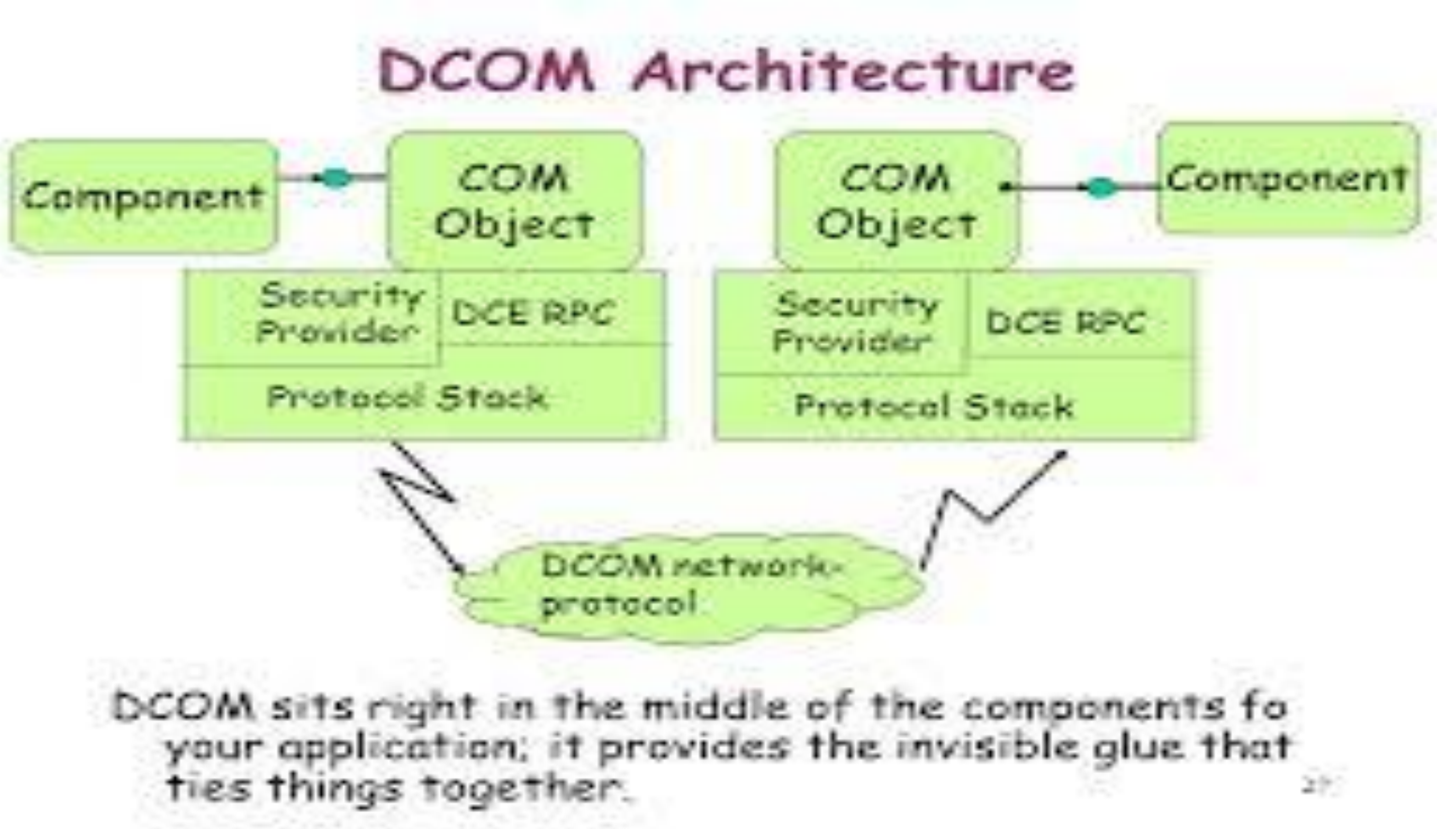
Distributed Component Object Model

- Distributed Component Object Model (DCOM)

- Protocol that enables s/w components to communicate directly over the n/w , in a reliable, secure manner

- It is developed to use across multiple different transports such as HTTP

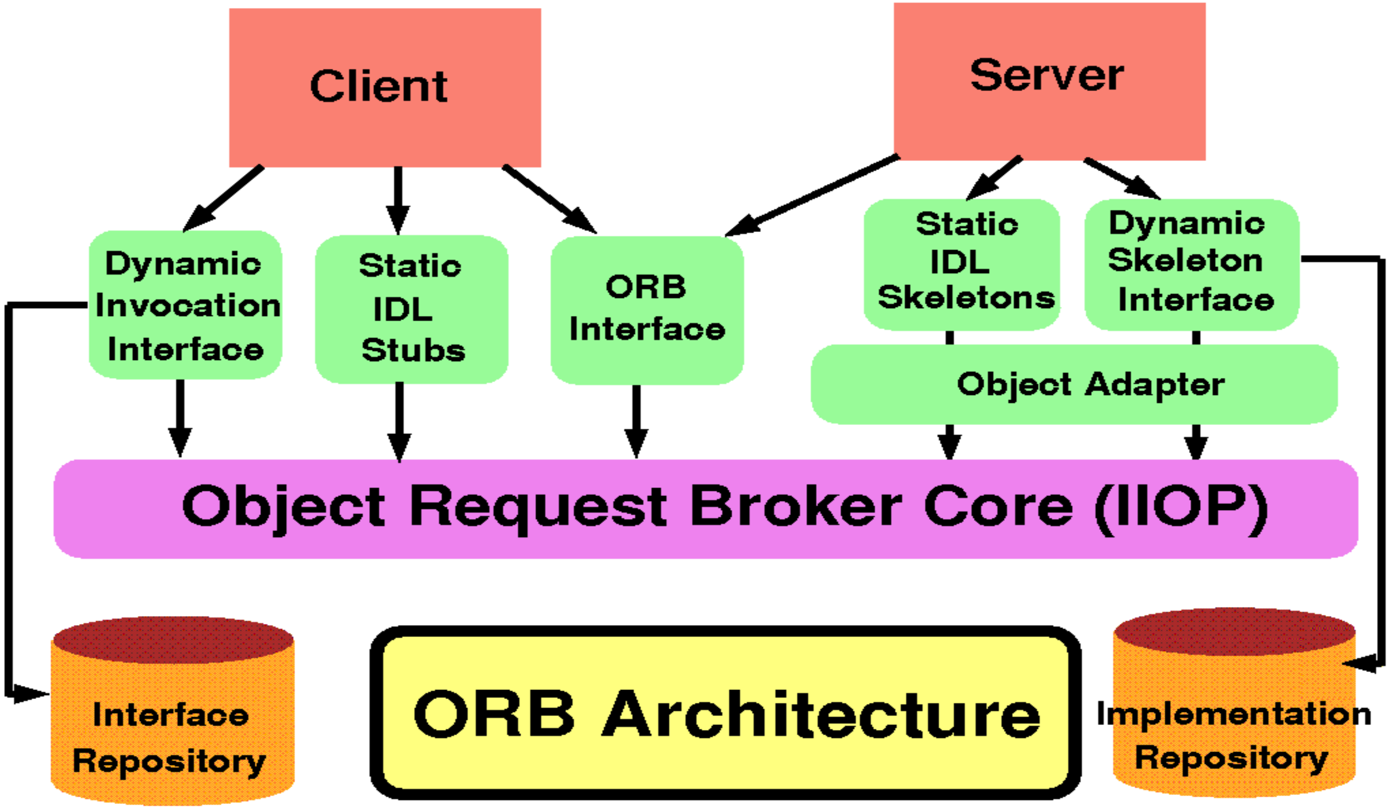
DCOM Architecture



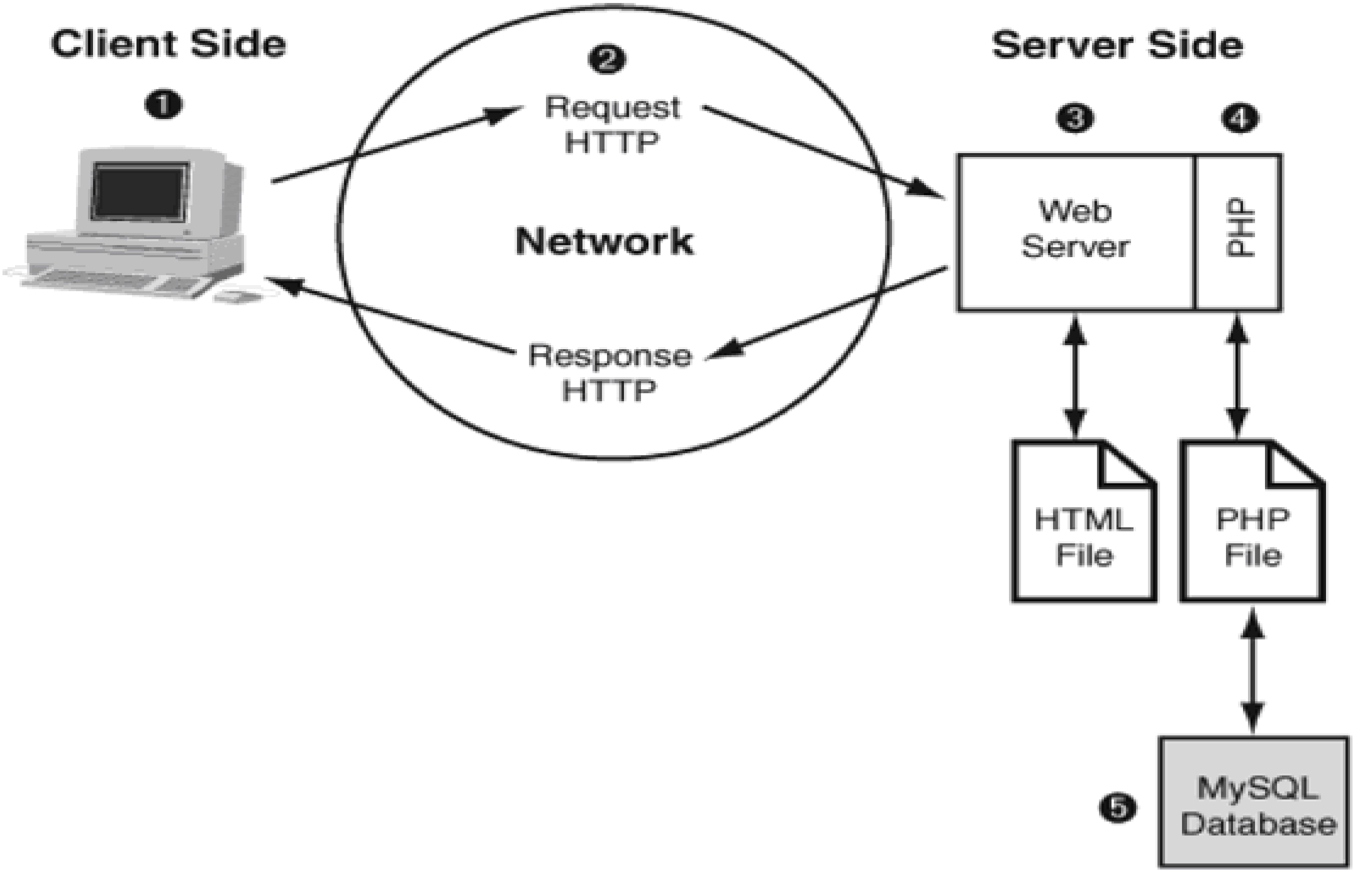
DCOM Architecture

DCOM sits right in the middle of the components fo your application; it provides the invisible glue that ties things together.

Common Object Request Broker (CORBA) Architecture


Client
Server
Dynamic Invocation Interface
Static IDL Stubs
ORB Interface
Static IDL Skeletons
Dynamic Skeleton Interface
Object Adapter
Object Request Broker Core (IIOP)
Interface Repository
ORB Architecture
Implementation Repository

Disributed Internet Architecture



Distributed Architecture – Application Logic

- Distributed Architecture:
- Application logic – server side, client respond to event on web page download from webserver
- Emphasis on
- How application logic partitioned
- Where partitioned unit should reside
- How they interact

Contd…

- **Traditional Distributed Applications:**
- Components resides on 1 or more application servers in various granularity
- Same server uses proprietary API otherwise RPC
- Actual reference to physical components is embedded in the code – tight coupling
- **SOA:**
- Rely on components, based on service orientation principles
- Positioned to expose set of functionalities
- Open standardize interface
- Loosely coupled
- Document style message self sufficient
- Solution agnostics services

Distributed Internet Architecture – Application Processing

- Distributed Internet Architecture:
- Use proprietary communication protocol (DCOM, CORBA)
- Active connection – relaible
- Support stateful and stateless
- SOA:
- Message based communication – involves serialization, transmission and de-serialization of SOAP message
- RPC is faster than SOAP
- Synchronous and asynchronous communication pattern supported
- To support statelessness, supported through WS-* spec., WS-Coordiantion, WS-BPEL and custom solutions

Distributed Internet Architecture – Technology

- **Distributed Internet Architecture (DIA):**
- Web technologies used
- XML is used for data representation & content transfer
- Allows the use of web services
- **SOA:**
- SOA uses same XML and web services but with governance
- Traditional DIA, XML and Web Services is optional

Distributed Internet Architecture – Security

- Distributed Internet Architecture :
- Server side security using login and safe transportation ensured
- When connection is removed, to handle security breaches, delegation, impersonation and Encryption is added
- SOA:
- Security is added as extension through WS-Security framework
- Message level security is done for autonomy and loose coupling

Distributed Internet Architecture – Administration

- Distributed Internet Architecture:
- Maintaining and tracking components, communication problems, web server monitoring
- Web server scalability – web server farms
- SOA:
- Exception handling with message frameworks are more complex
- Need Ws-* extension to deal with
- Maintenance task – resource management – done using UDDI

SOA Vs Hybrid Web Service Architecture

- Hybrid Web Service Architecture:
- Primary role of Web service (WS) – integration layer – as a wrapper WS
- Synchronous communication via SOAP complaint messages
- Mimic RPC style communication with other applications / utilize third party WS
- It mimics component interfaces of point-to-point connections with web services
- **It is simply a distributed Internet architecture that uses Web Service not an SOA**

Contd…

- SOA:
- Supports various message models (synchronous or asynchronous)
- WS in SOA build on service orientation principles
- No point-to-point communication – loosely coupling - single service, handle any no. of requestors
- SOA build with WS automate one/more business process
- WS organized into specialized layers – abstract specific part of enterprise logic
- Standardizing SOA allows interoperability across an enterprise