Course Contents

- Introduction to WSDL, SOAP, JAX-WS
- WS-Addressing specification
- WS-XOP specification
- MTOM mechanism in web services
- Wrapped web services
- Handling complex data
- Asynchronous web services
- WS-Security specification
- WS-Metadata specification
- JAXB 2.0 a binding technology
- STAX parsing
- Different JAX-WS clients
- Servlet and EJB endpoints
- Integrating Web Services with Tomcat
- Different types of JAX-WS clients
- Fault generation in a web service

seed
beyond the obvious

Objectives

- At the end of this chapter you will be able to Understand
  - The definition of distributed computing
  - The importance of distributed computing
  - Core distributed computing technologies such as the following:
    - Client/server
    - CORBA
    - Java RMI
    - Microsoft DCOM
    - Message-Oriented Middleware
  - Common challenges in distributed computing
  - The role of J2EE and XML in distributed computing
  - Emergence of Web services and service-oriented architectures

3

seed
beyond the obvious

What Is Distributed Computing?

- *Distributing Computing is a type of computing in which different components and objects comprising an application can be located on different computers connected to a network**

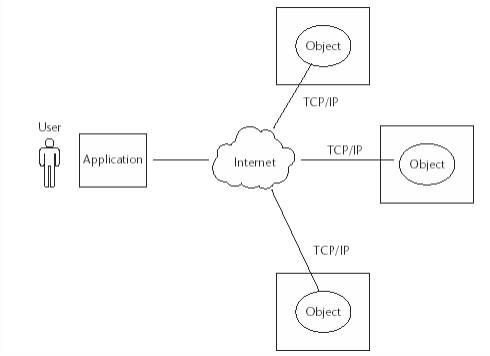
- * Ref:- (www.webopedia.com, May 2001).

4

seed
beyond the obvious

What Is Distributed Computing?

- A distributed computing model provides an infrastructure enabling invocations of object functions located anywhere on the network.
- The objects are transparent to the application and provide processing power as if they were local to the application calling them.



5

seed
beyond the obvious

The Importance of Distributed Computing

- The ***advantages*** of distributed computing are as given
  - **Higher performance.** Applications can execute in parallel and distribute the load across multiple servers.
  - **Collaboration.** Multiple applications can be connected through standard distributed computing mechanisms.
  - **Higher reliability and availability.** Applications or servers can be clustered in multiple machines.
  - **Scalability.** This can be achieved by deploying these reusable distributed components on powerful servers.
  - **Extensibility.** This can be achieved through dynamic (re)configuration of applications that are distributed across the network.

6

seed
beyond the obvious

The Importance of Distributed Computing

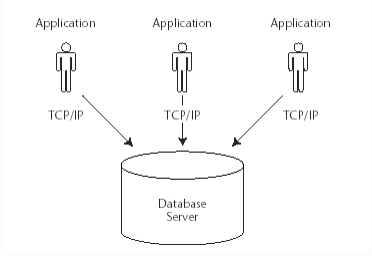
- The ***advantages*** of distributed computing continued…
  - **Higher productivity and lower development cycle time.** By breaking up large problems into smaller ones, these individual components can be developed by smaller development teams in isolation.
  - **Reuse.** The distributed components may perform various services that can potentially be used by multiple client applications. It saves repetitive development effort and improves interoperability between components.
  - **Reduced cost.** Because this model provides a lot of reuse of once developed components that are accessible over the network, significant cost reductions can be achieved.

7

seed
beyond the obvious

Distributed Computing- Client Server technology

- An architectural model of a typical client server system in which multiple desktop-based business client applications access a central database server.



8

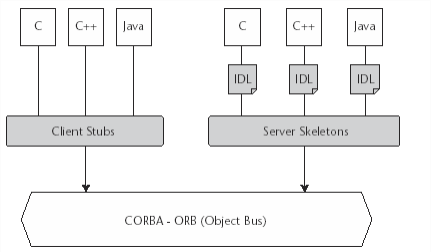
seed
beyond the obvious

Distributed Computing- Client Server technology

- Some of the common **limitations** of the client-server application model are :
  - Complex business processing at the client side demands robust client systems.
  - Security is more difficult to implement because the algorithms and logic reside on the client side making it more vulnerable to hacking.
  - Increased network bandwidth is needed to accommodate many calls to the server, which can impose scalability restrictions.
  - Maintenance and upgrades of client applications are extremely difficult because each client has to be maintained separately.
  - Client-server architecture suits mostly database-oriented standalone applications and does not target robust reusable component oriented applications.

Distributed Computing- CORBA

- The architectural model of CORBA with an example representation of applications written in C, C++, and Java providing IDL bindings.



Distributed Computing- CORBA

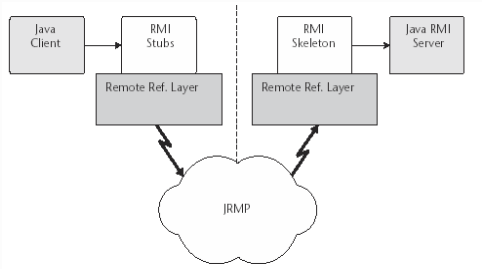
- Some of the **advantages** of the CORBA over a traditional client/server application model are :
  - **OS and programming-language independence**.
  - **Legacy and custom application integration**.
  - **Rich distributed object infrastructure.** CORBA offers developers a rich set of distributed object services, such as the Lifecycle, Events, Naming, Transactions, and Security services.
  - **Location transparency**
  - **Remote callback support.** CORBA allows objects to receive asynchronous event notification from other objects.
  - **Dynamic invocation interface.** CORBA clients can both use static and dynamic methods invocations. They either statically define their method invocations through stubs at compile time, or have the opportunity to discover objects' methods at runtime.

Distributed Computing- CORBA

- Some of the **limitations** of the CORBA are :
  - **High initial investment.** CORBA-based applications require huge investments in regard to new training and the deployment of architecture, even for small-scale applications.
  - **Availability of CORBA services.** The Object services specified by the OMG are still lacking as implementation products.
  - **Scalability.** Due to the tightly coupled nature of the connection oriented CORBA architecture, very high scalability expected in enterprise applications may not be achieved.

Distributed Computing- Java RMI

- RMI provides an architectural model of distributed Java application environment by calling remote Java objects and passing them as arguments or return values.
- It uses Java object serialization—a lightweight object persistence technique that allows the conversion of objects into streams.



13

seed beyond the obvious

Distributed Computing- Java RMI

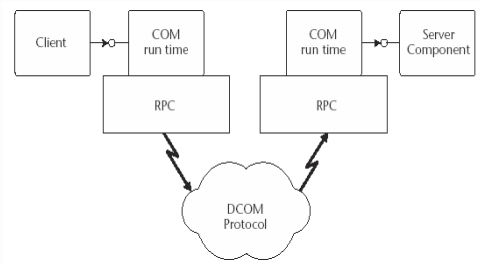
- Although RMI had inherent advantages provided by the distributed object model of the Java platform, it also had some **limitations**:
  - RMI is limited only to the Java platform. It does not provide language independence in its distributed model as targeted by CORBA.
  - RMI-based application architectures are tightly coupled because of the connection-oriented nature. Hence, achieving high scalability in such an application model becomes a challenge.
  - RMI does not provide any specific session management support. In a typical client/server implementation, the server has to maintain the session and state information of the multiple clients who access it. Maintaining such information within the server application without a standard support is a complex task.

14

seed beyond the obvious

Distributed Computing- Microsoft DCOM

- An architectural model of DCOM (The Distributed Common Object Model) developed by Microsoft is an answer to the distributed computing problem in the Microsoft Windows platform.
- DCOM enables COM applications to communicate with each other using an RPC mechanism, which employs a DCOM protocol on the wire.



15

seed beyond the obvious

Distributed Computing- Microsoft DCOM

- The following are some of the common **limitations** of DCOM:
  - Platform lock-in
  - State management
  - Scalability
  - Complex session management issues

16

seed beyond the obvious

Distributed Computing- Message Oriented Middleware

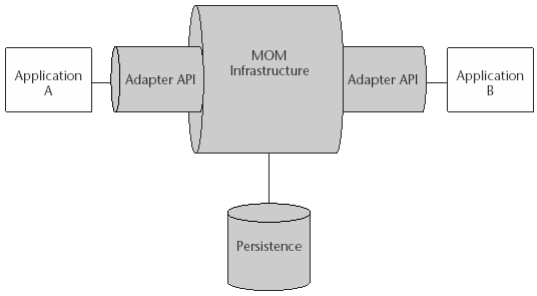
- Although CORBA, RMI, and DCOM differ in their basic architecture and approach, they adopted a tightly coupled mechanism of a ***synchronous*** communication model (request/response). All these technologies are based upon ***binary communication protocols*** and adopt tight integration across their logical tiers, which is susceptible to scalability issues.
- Message-Oriented Middleware (MOM) is based upon a ***loosely coupled asynchronous communication*** model where the application client does not need to know its application recipients or its method arguments. MOM enables applications to communicate indirectly using a messaging provider queue.

17

seed
beyond the obvious

Distributed Computing- Message Oriented Middleware

- Some of the widely known MOM-based technologies are SunONE Message Queue, IBM MQSeries, TIBCO, SonicMQ, and Microsoft Messaging Queue (MSMQ).
- The Java Platform provides a Java-based messaging API (JMS-Java Message Service)



18

seed
beyond the obvious

Distributed Computing- Message Oriented Middleware

- JMS provides **Point-to-Point** and **Publish/Subscribe** messaging models with the following features:
  - Complete transactional capabilities
  - Reliable message delivery
  - Security

Distributed Computing- Message Oriented Middleware

- Some of the common **challenges** while implementing a MOM-based application environment have been the following:
  - Most of the standard MOM implementations have provided native APIs for communication with their core infrastructure. This has affected the portability of applications across such implementations and has led to a specific vendor lock-in.
  - The MOM messages used for integrating applications are usually based upon a proprietary message format without any standard compliance.

Challenges in Distributed Computing

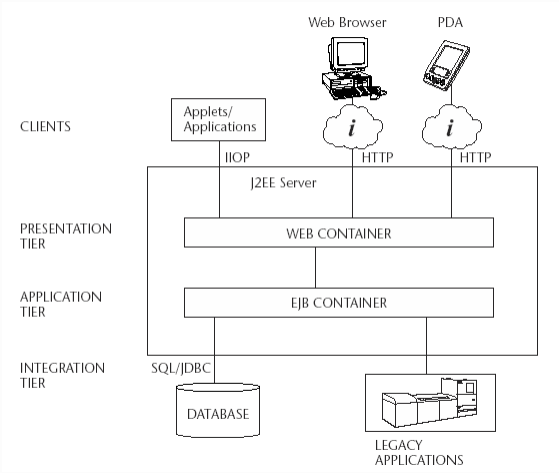
- Some of the **challenges** in the CORBA-, RMI-, and DCOM-based distributed computing solutions:
  - ***Maintenance*** of various versions of stubs/skeletons in the client and server environments is extremely complex in a heterogeneous network environment.
  - ***Quality of Service*** (QoS) goals like Scalability, Performance, and Availability in a distributed environment consume a major portion of the application's development time.
  - ***Interoperability*** of applications implementing different protocols on heterogeneous platforms almost becomes impossible. For example, a DCOM client communicating to an RMI server or an RMI client communicating to a DCOM server.
  - Most of these protocols are designed to work well within local networks. They are not very firewall friendly or able to be accessed over the Internet.

The Role of J2EE and XML in Distributed Computing

- With many organizations moving their businesses to the Internet, a whole new category of business models like ***business-to-business*** (B2B) and ***business-to-consumer*** (B2C) came into existence.
- This evolution lead to the specification of ***J2EE architecture***, which promoted a much more efficient platform for hosting Web-based applications.
- J2EE provides a programming model based upon ***Web and business components*** that are managed by the J2EE application server.
- The application server consists of many APIs and ***low-level services*** available to the components. These low-level services provide security, transactions, connections and instance pooling, and concurrency services, which enable a J2EE developer to focus primarily on business logic rather than plumbing.

The Role of J2EE and XML in Distributed Computing

- Figure shows various components of the J2EE architecture.



23

seed
beyond the obvious

The Role of J2EE and XML in Distributed Computing

- The emergence of the Extensible Markup Language (XML) for defining portable data in a structured and self-describing format is embraced by the industry as a communication medium for electronic data exchange.
- Using XML as a data exchange mechanism between applications promotes interoperability between applications and also enhances the scalability of the underlying applications.
- Combining the potential of a J2EE platform and XML offers a standard framework for B2B and inter-application communication across networks.

24

seed
beyond the obvious

The Role of J2EE and XML in Distributed Computing

- With J2EE enabling enterprise applications to the Internet and XML acting as a "glue" bridges these discrete J2EE-based applications by facilitating them to interoperate with each other.
- XML, with its incredible flexibility, also has been widely adopted and accepted as a standard by major vendors in the IT industry, including Sun, IBM, Microsoft, Oracle, and HP.
- The combination of these technologies promotes a new form of the distributed computing technology solution referred to as ***Web services***.

25

seed
beyond the obvious

The Emergence of Web Services

- The increasing demands of the industry for enabling B2B, application-to-application (A2A), communication has led to a growing requirement for ***service-oriented architectures***.
- Enabling service-oriented applications facilitates *the exposure of business applications as service components enable business applications from other organizations to link with these services for application interaction and data sharing without human intervention.*
- By adopting Web technologies, the service-oriented architecture model facilitates the delivery of services over the Internet by leveraging standard technologies such as XML.
- It uses platform-neutral standards by exposing the underlying application components and making them available to any application, any platform, or any device, and at any location.
- Today, this phenomenon is well adopted for implementation and is commonly referred to as ***Web services***.

26

seed
beyond the obvious

Quick Recap . . .

- In this session we have seen
  - The importance of distributed computing
  - Different technologies targeting distributed programming
  - Limitations of these technologies
  - Emergence of Web Services