T-110.5140 Network Application Frameworks and XML

Middleware

02.03.2009

Prof. Sasu Tarkoma

Contents

- Middleware
  - Motivation
  - Examples
- Summary

Middleware

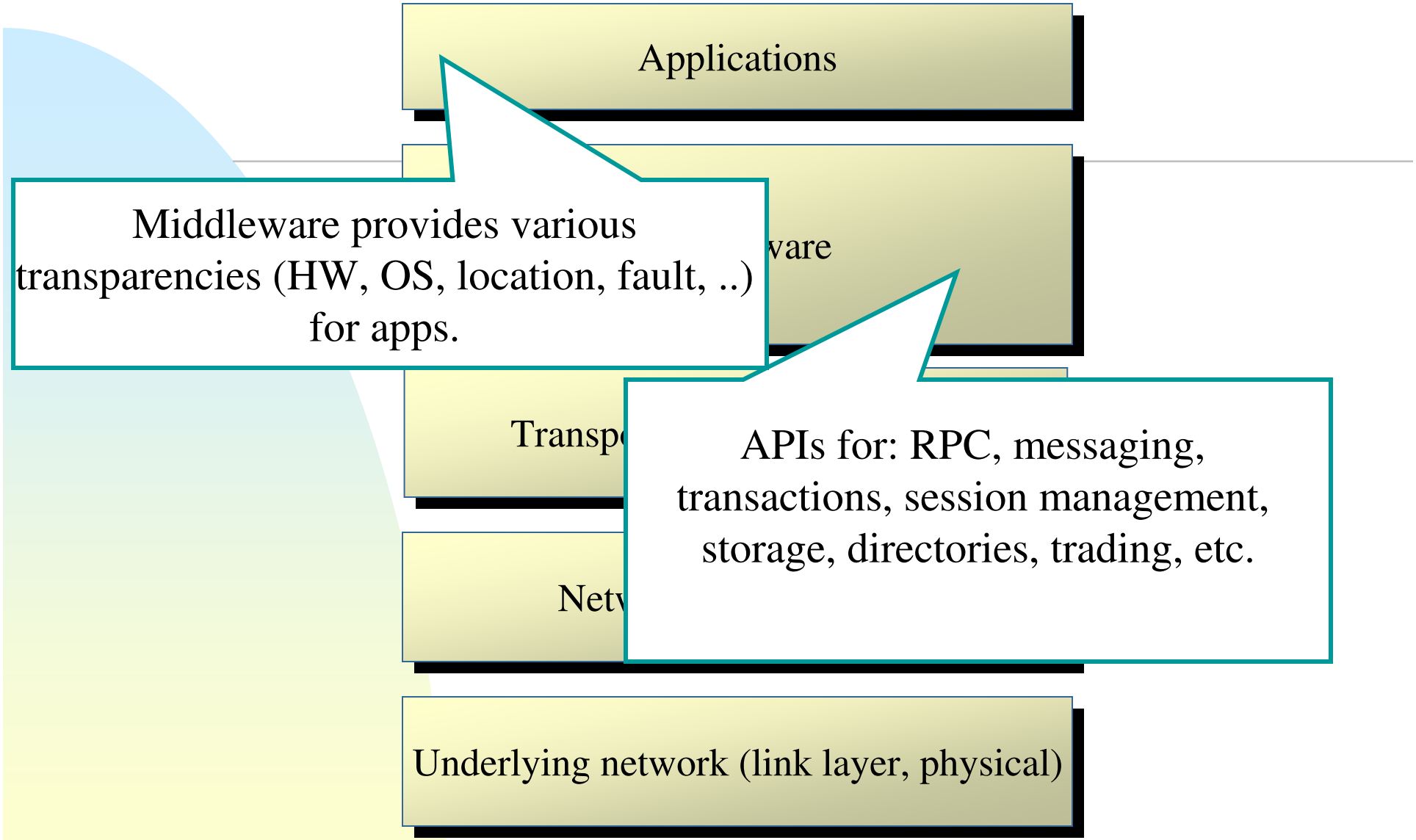
- Widely used and popular term
- Fuzzy term
- One definition
  - “A set of service elements above the operating system and the communications stack”
- Second definition
  - “Software that provides a programming model above the basic building blocks of processes and message passing” (Colouris, Dollimore, Kindberg, 2001)

Why Middleware?

- Application development is complex and time-consuming
  - Should every developer code their own protocols for directories, transactions, ..?
  - How to cope with heterogeneous environments?
    - Networks, operating systems, hardware, programming languages
- Middleware is needed
  - To cut down development time
    - Rapid application development
  - Simplify the development of applications
  - Support heterogeneous environments and mask differences in OS/languages/hardware

Middleware cont.

- Middleware services include
  - directory, trading, brokering
  - remote invocation (RPC) facilities
  - transactions
  - persistent repositories
  - location and failure transparency
  - messaging
  - Security
- Network stack (transport and below) is not part of middleware


Applications
Middleware provides various transparencies (HW, OS, location, fault, ..) for apps.
APIs for: RPC, messaging, transactions, session management, storage, directories, trading, etc.
Underlying network (link layer, physical)

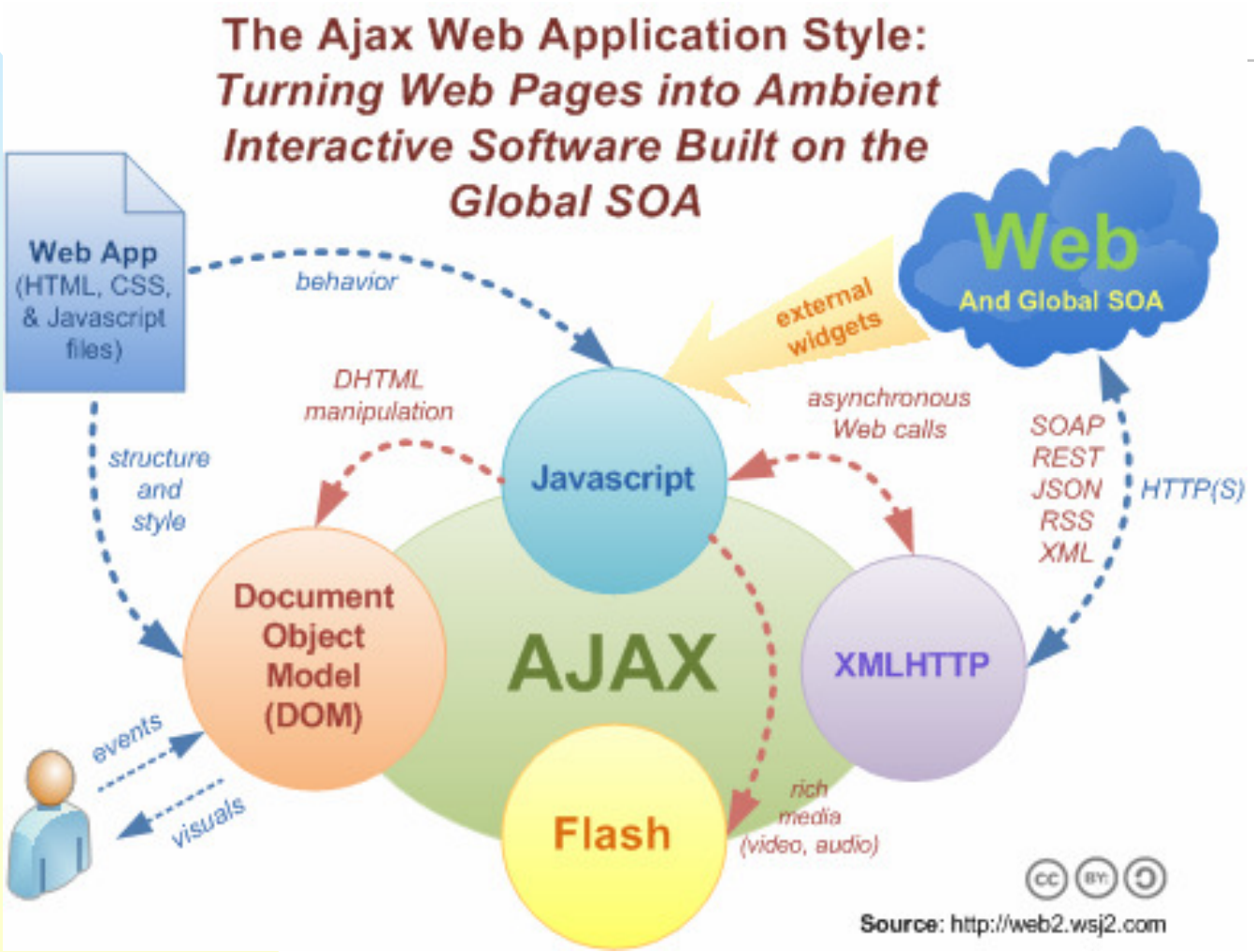
Examples

- Remote Procedure Call (RPC)
  - call of a remote procedure as it were local
  - marshalling / unmarshalling
- Remote Method Invocation (RMI)
  - call of a remote method as it were local
  - marshalling / unmarshalling
- Event-based computing
  - entities receive asynchronous notifications
  - a notification causes a state change
- Overlays and P2P content delivery

Web applications

- Recent trend has been to develop web applications
  - Traditional applications on Internet (office suites,..)
  - Search (Google, Yahoo, ..)
  - Instant communications and presence
  - Social collaboration and networking sites
  - Data sharing sites and video sharing
  - Data storage services
  - Blogging
- Another recent trend is to simplify signing to services
  - Single Sign-On, federated identity, OpenID
- And creating mashups
  - Combining services in new ways
  - Custom experience and personalization

AJAX


The Ajax Web Application Style:
Turning Web Pages into Ambient Interactive Software Built on the Global SOA
Web App (HTML, CSS, & Javascript files)
behavior
Web
And Global SOA
external widgets
DHTML manipulation
asynchronous Web calls
SOAP
REST
JSON
RSS
XML
HTTP(S)
structure and style
Javascript
Document Object Model (DOM)
AJAX
XMLHTTP
events
visuals
Flash
rich media (video, audio)
Source: http://web2.wsj2.com

Web Service Architecture

- Motivation
  - Machine readable content on the Web
  - Programming API for the Web
  - Access independent of the computing environment
- The three major roles in web services
  - Service provider
    - Provider of the WS
  - Service Requestor
    - Any consumer / client
  - Service Registry
    - logically centralized directory of services
- A protocol stack is needed to support these roles

Transparencies

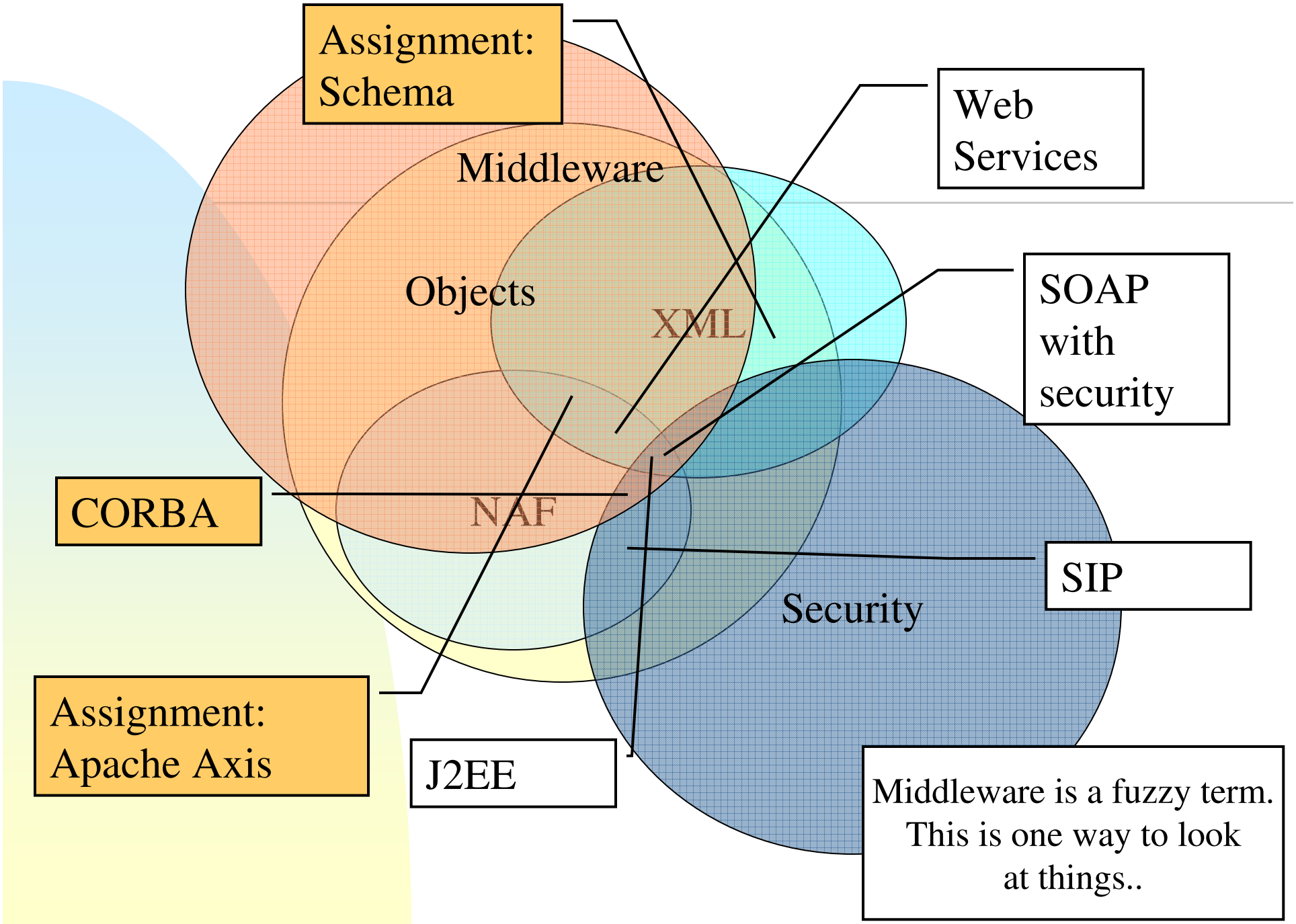
- Location transparency
  - RPC and RMI used without knowledge of the location of the invoked procedure / object
- transport protocol transparency
  - RPC may be implemented using any transport protocol
- transparency of OS and hardware
  - RPC/RMI uses external data representation
  - Presentation is important
  - XML is becoming increasingly important
- transparency of programming languages
  - language independent definition of procedures: CORBA IDL, WSDL

Network Application Framework

- Network Application Framework is middleware
- Contains services for distributed applications
- Middleware as a term is fuzzier and larger
- In this course, we focus on network application frameworks and XML
  - objects (discovery, representation)
  - directories (overlays,..)
  - network
  - security

Object Model

- Components: object references, interfaces, actions (methods), exceptions, garbage collection
- Distributed object model
  - client-server model
  - usually implemented using request-reply
  - marshalling, unmarshalling
  - Example: Java RMI
  - Invocation Semantics
    - Exactly-once cannot be guaranteed
    - Typical semantics: at-least-once, at-most-once
    - At-least-once is good for idempotent operations
    - CORBA and RMI use at-most-once


Assignment: Schema
Middleware
Web Services
Objects
XML
SOAP with security
CORBA
NAF
SIP
Security
Assignment: Apache Axis
J2EE
Middleware is a fuzzy term. This is one way to look at things..

Examples

- CORBA
- Message-oriented Middleware
- Event Systems & tuple spaces
- Java Message Service
- Java 2 Enterprise Edition (J2EE)
- .NET

CORBA I

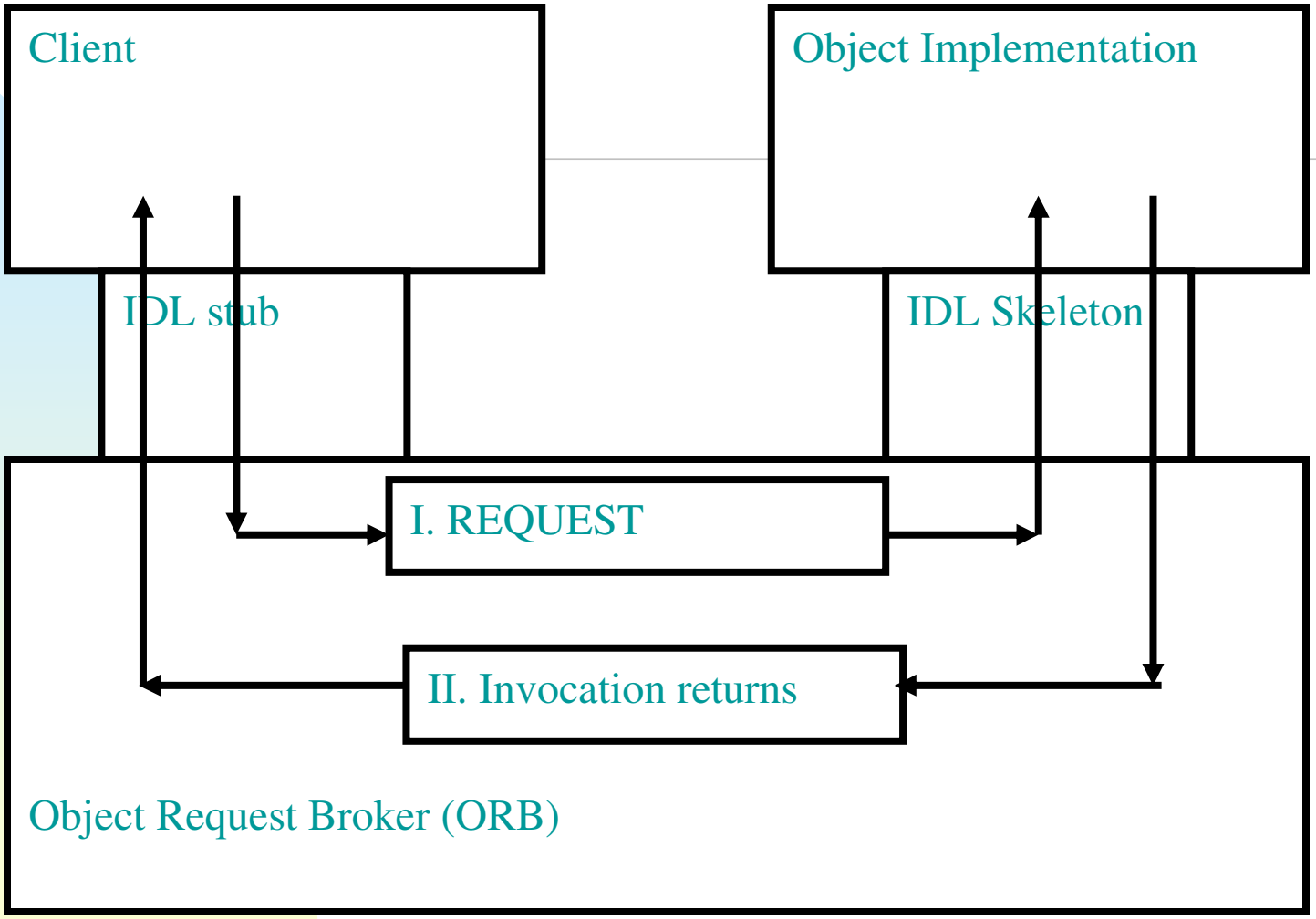
- Common Object Request Broker Architecture (CORBA)
  - Standardized by Object Management Group (OMG)
  - OMG est. 1989, currently over 800 members
- Distributed object-oriented middleware
  - Network abstraction of object location
  - Support for heterogeneous environments
    - hardware, networks, OS, languages

Interface Definition Language (IDL)

- IDL is language independent
- Used to define object interfaces
- Hides underlying object implementation
- Language mappings for C, Java, C++, Cobol, ..
- IDL compiler generates language specific stubs and skeletons from an IDL definition
- Stubs and skeletons marshal and unmarshal request/response data to packets

CORBA II

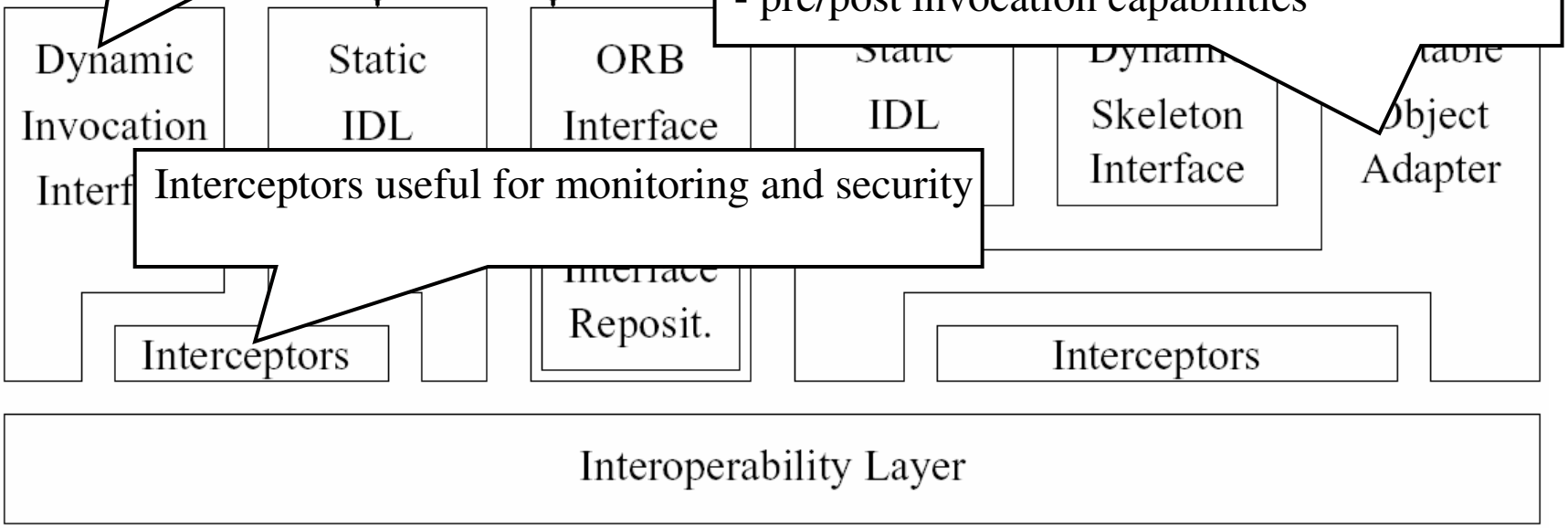
- Object Request Broker (ORB)
  - Broker pattern, transparent object invocation
    - object location, activation, communication
    - CORBA works for both OO and non-OO languages
- Interoperable Object Reference (IOR)
  - Uniquely identifies each object
  - Shareable reference
- Support for dynamic and static method invocation
- Many commercial and non-commercial implementations


Client
Object Implementation
IDL stub
IDL Skeleton
I. REQUEST
II. Invocation returns
Object Request Broker (ORB)

Implementation layer

- Dynamic Invocation Interface
- Alternative to static stub/skeleton calls
- Generic runtime invocation, generic inte
defined in IDL, first search and locate int
then do the invocation

- generate and interpret object references
- demultiplex requests
- Handle method invocations via skeletons
- activation policies, thread models
- object life cycle
- pre/post invocation capabilities



CORBA communications

- GIOP (General Inter-ORB Protocol)
  - Abstract protocol for ORBs
  - Common Data Representation
    - On-the-wire presentation of OMG IDL data types
  - Interoperable Object Reference (IOR(
    - Format for describing remote reference
    - Protocol, server address, object key
  - The defined message formats
    - Request,reply,fragment, ..
- IIOP (Internet Inter-ORB Protocol)
  - GIOP implementation for TCP/IP

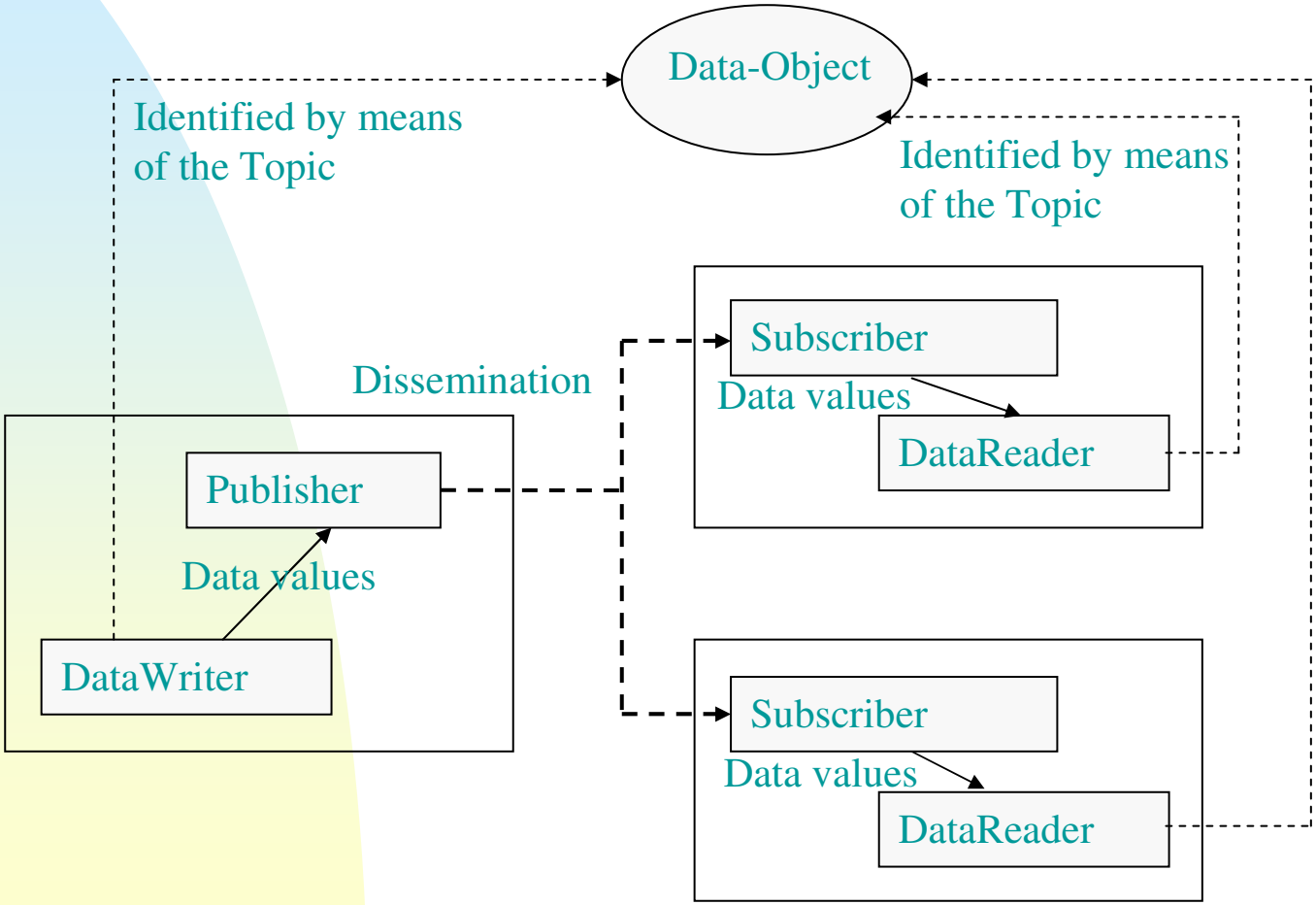
CORBA Services

- Services specified by OMG to help using distributed objects
- Naming Service
- Event and Notification Service
- Security Service
  - authentication, access control, non-repudiation
- Persistent Object Service
  - persistent objects (activation / deactivation)
- Trading Service
  - directory service, objects are identified by attributes
- Transaction and Concurrency Control Service
  - database transactions

OMG Distributed Data Service I

- The Data Distribution Service for Real-Time Systems (DDS)
- The specification defines an API for data-centric publish/subscribe communication for distributed real-time systems.
- DDS is a middleware service that provides a global data space that is accessible to all interested applications.
- DDS uses the combination of a Topic object and a key to uniquely identify instances of data-objects.
- Content filtering and QoS negotiation are supported
- DDS is suitable for signal, data, and event propagation.

DDS II


Data-Object
Identified by means of the Topic
Identified by means of the Topic
Subscriber
Data values
DataReader
Dissemination
Publisher
Data values
DataWriter
Subscriber
Data values
DataReader

Message-oriented Middleware

- Transfers messages between applications
  - Does not consider the content of messages
- Asynchronous communication
- Direct or queued
  - Queued (buffered) communication supports wireless clients
- Examples
  - Sun Microsystems JMS
  - Microsoft: MSMQ
  - IBM: Websphere MQ

Event Systems I

- Traditional MoM systems are message queue based (one-to-one)
- Event systems and publish/subscribe are one-to-many or many-to-many
  - One object monitors another object
  - Reacts to changes in the object
  - Multiple objects can be notified about changes
- Events address problems with synchronous operation and polling
- In distributed environments a logically centralized services mediates events
  - anonymous communication
  - expressive semantics using filtering

Event Systems II

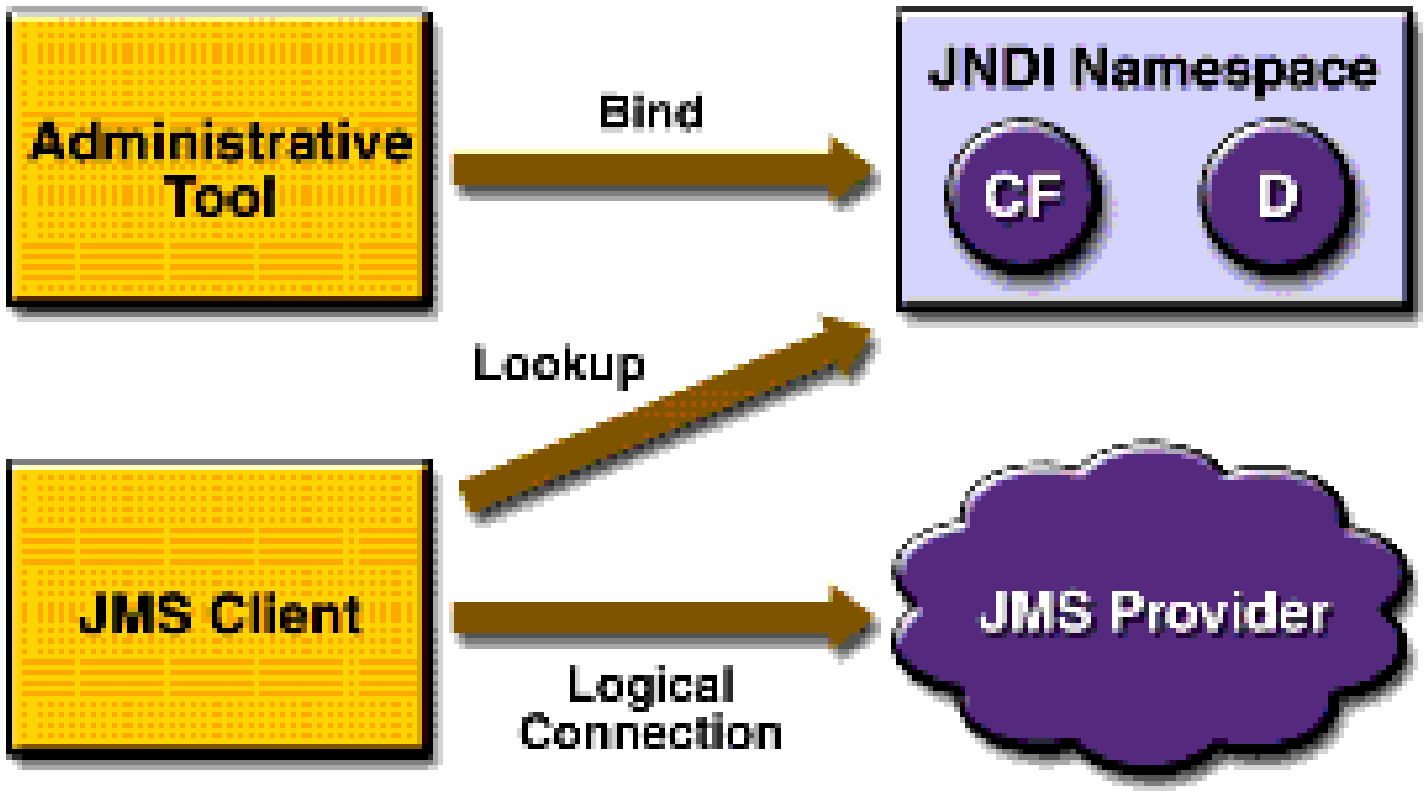
- Push versus Pull
- May be implemented using RPC, unicast, multicast, broadcast,..
- Three main patterns
  - Observer design pattern
    - Used in Java / Jini
  - Notifier architectural pattern
    - Used by many research systems
  - Event channel
    - Used in CORBA Event/Notification Service
- Filtering improves scalability / accuracy
  - Research topic: content-based routing

Tuple Spaces

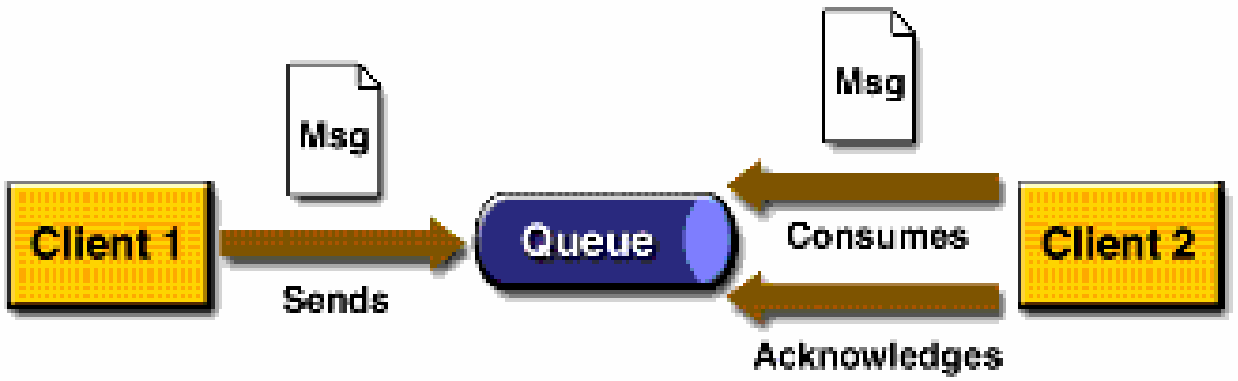
- Tuple-based model of coordination
- The shared tuple space is global and persistent
- Communication is
  - decoupled in space and time
  - implicit and content-based
- Primitives
  - **In,** atomically read and removes a tuple
  - **Rd**, non-destructive read
  - **Out**, produce a tuple
  - **Eval**, creates a process to evaluate tuples
- Examples: Linda, Lime, JavaSpaces, TSpaces

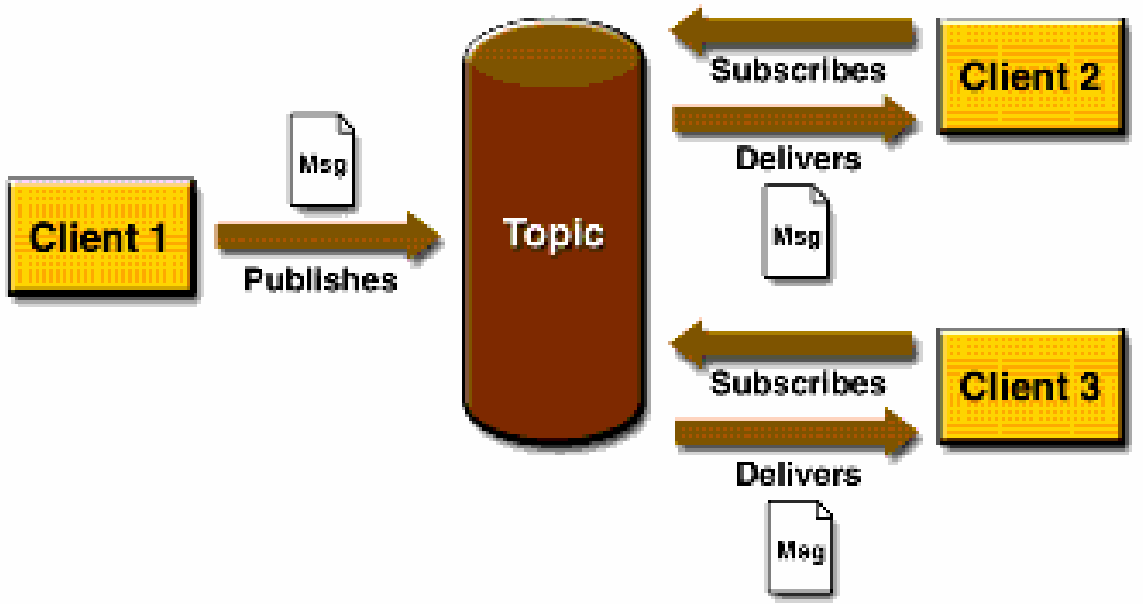
Java Message Service (JMS)

- Asynchronous messaging support for Java
- Point-to-point messaging
  - One-to-one
- Topic-based publish/subscribe
  - SQL for filtering messages at the topic event queue
  - One-to-many
- Message types:
  - Map, Object, Stream, Text, and Bytes
- Durable subscribers
  - Event stored at server if not deliverable
- Transactions with rollback



Source: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html





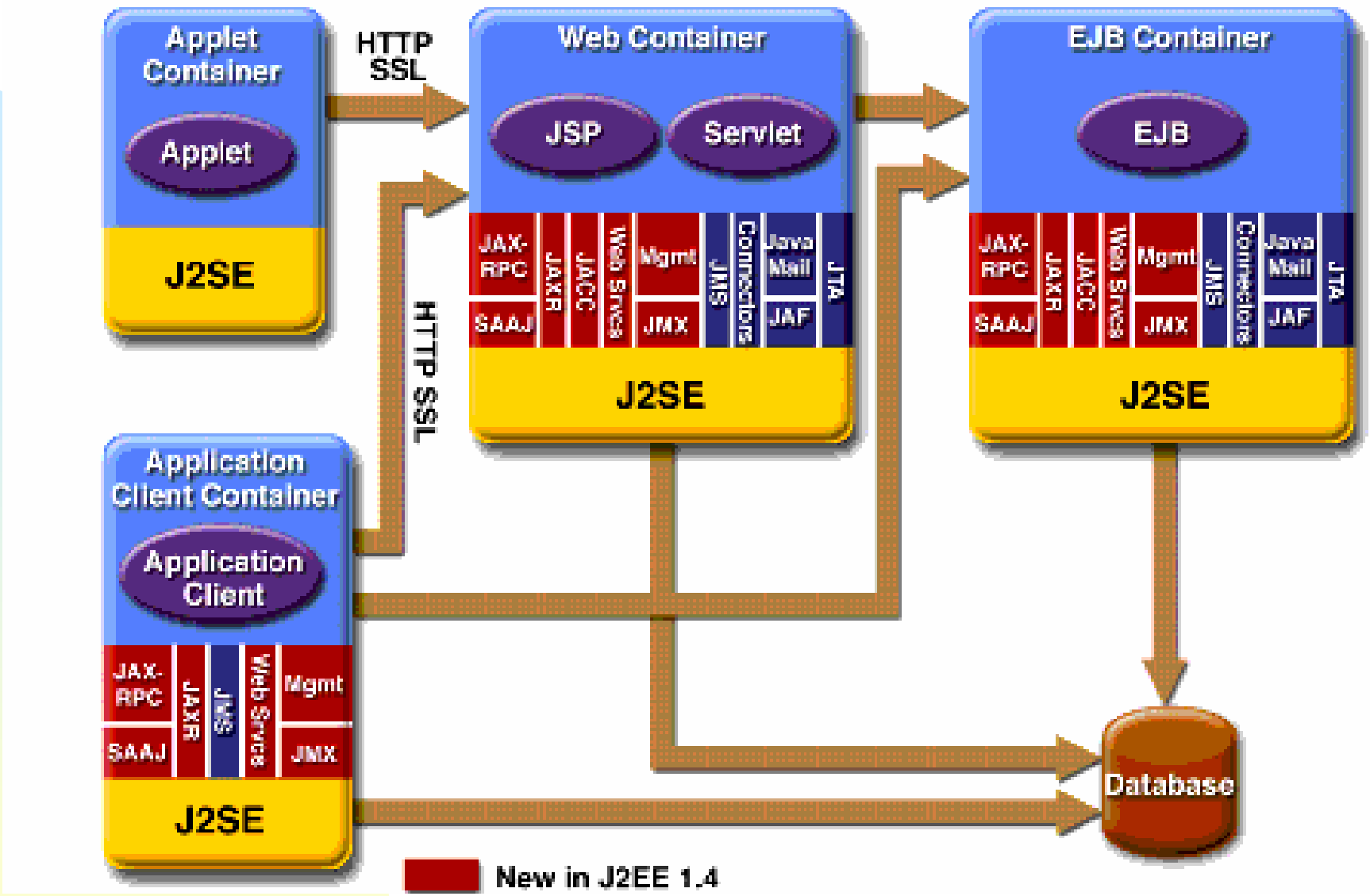
Source: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

JMS messaging

- JMS messaging proceeds in the following fashion:
  - Client obtains a Connection from a ConnectionFactory
  - Client uses the Connection to create a Session object
  - The Session is used to create MessageProducer and MessageConsumer objects, which are based on Destinations.
  - MessageProducers are used to produce messages that are delivered to destinations.
  - MessageConsumers are used to either poll or asynchronously consume (using MessageListeners) messages from producers.

Java 2 Platform Enterprise Edition (J2EE)

- Specifications and practices for developing, deploying, and managing multi-tier server-centric applications
  - Builds on J2SE
  - Web Services support
- Containers - separation of business logic from resource and lifecycle management
  - Enterprise JavaBeans (EJB)
  - Servlets
- Java Message Service (JMS)
  - async. communication supports decoupling



Source: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

J2EE Technologies

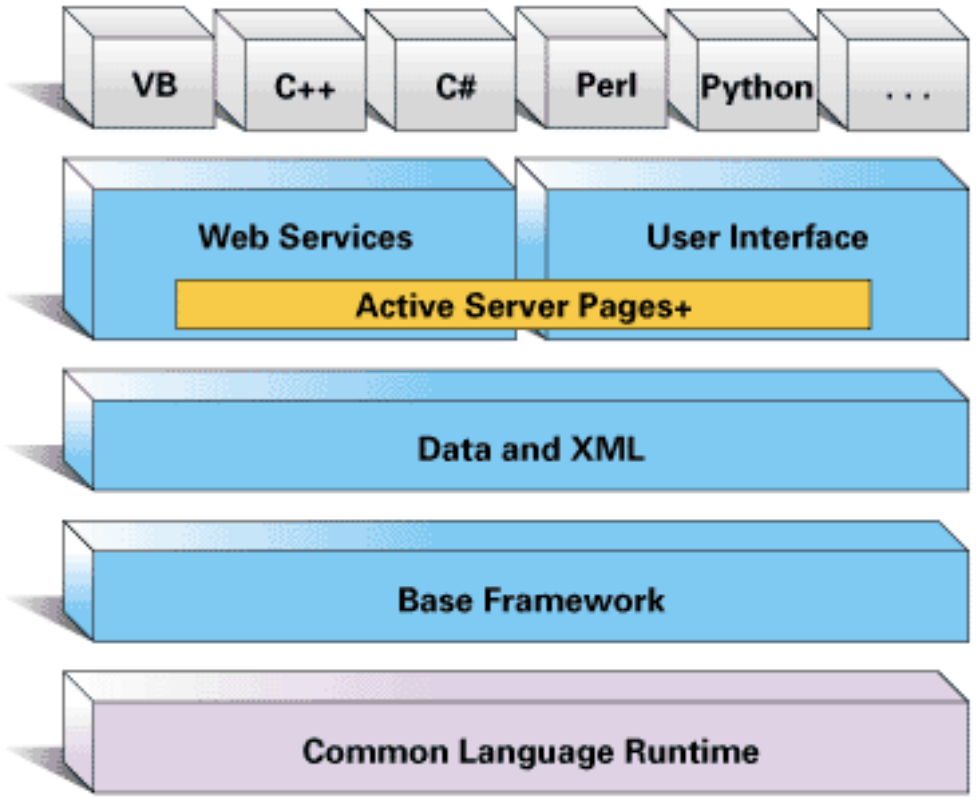
Java Management Extensions (JMX)
J2EE Authorization Contract for Containers
Java API for XML Registries (JAXR)
Java Message Service (JMS)
Java Naming and Directory Interface (JNDI)
Java Transaction API (JTA)
CORBA
JDBC data access API.

Java API for XML-Based RPC (JAX-RPC)
JavaServer Pages
Java Servlets
Enterprise JavaBeans components
J2EE Connector Architecture
J2EE Management Model
J2EE Deployment API

.NET

- The .NET Framework is Microsoft's the next generation application platform
  - applications, services, web services, ..
  - Protocol stack and computing model for TCP/IP based distributed computing
- Based on the CLR (Common Language Runtime)
  - JIT compiles and executes .NET code
- Components
  - .NET architecture, .NET Integrated Programming, Common Language Runtime (CLR), .NET System Class Libraries, Data and XML, Web Services / ASP+

.NET Architecture



Secure, integrated class libraries

- Unified programming models across languages
- Enables cross-language integration
- Factored for extensibility
- Designed for tools

New

Active Server Pages+

High-productivity environment for building and running Web services

Common Language Runtime

Executes code, maintains security, handles component "plumbing," and dependencies

Source: MSDN

Windows Communication Foundation

- Single technology platform that unifies a number of different techniques
  - ASP.NET Web Services (ASMX), Web Service Enhancements (WSE) extensions, the Microsoft Message Queue (MSMQ), Enterprise Services/COM+ runtime environment, .NET Remoting
- **Address, Binding, Contract**
- Service oriented programming model with a single API for comms
  - Unifies Web services, .NET remoting, Distributed Transactions, Message Queues
- Based on SOAP and XML

WCF Service

- Three parts
  - Service class
    - Service contract
      - Data contract
  - Host environment
  - Endpoints
    - Contract for accessing endpoints
  - Endpoints use proxy objects to communicate RPC style (abstract service as an object)

Summary

- Middleware
  - for application development and deployment
  - for supporting heterogeneous environments
  - Main communication paradigms: RPC/RMI, asynchronous events (publish/subscribe)
  - Standardization needed
  - J2EE, CORBA, ..
  - J2EE/JMS Java specific
- Current trends
  - Flexibility, decoupled nature
  - Convergence / unification