

Middleware

- "Glue" code between network operating system and customer applications
- In support of distributed heterogeneous computation
- Similar in concept to architectural connectors

Context

- Growth of networked machines
 - Increasing customer base
- Specialization of hardware resources
- Increasingly powerful applications
 - Integration of existing components
 - Negotiation

Needs

- Abstractions
 - Interfaces (APIs)
 - Standards
- Notations
- Tools
 - Design
 - Code generation

Characteristic Issues

1. Network communication
2. Coordination
3. Reliability
4. Scalability
5. Heterogeneity

1. Network Communication

- Data transport (marshalling)
- Transactions
 - *ACID*
 - Atomtic (all or none)
 - Consistency-preserving (database integrity)
 - Isolated (other transactions can't see intermediate states)
 - Durable (committed and persistent)
- Error handling / reliable delivery

Data Transportability

marshalling, serialization, pickling

- Bit order
- Byte order
- Character sets
- Alignment
- Word length
- Organization (of constituent pieces)
- Self definition (identity)

2. Coordination

- Synchronization
 - *Synchronous*: client blocked while waiting for service
 - *Asynchronous*: server initiated
- Multicast (push)
- Hardware stop/restart
- Activation/deactivation (persistent state)
- Server concurrency

Concurrency Control

- Session layer responsible for synchronization
- Blocking (synchronous)
- Non-blocking
 - Client (polling); deferred synchronization; pull
 - Server; asynchronous; push

3. Reliability

- Assurance policies (communication)
 - Best-effort
 - At-most-once
 - At-least-once
 - Exactly-once
- Reliability/performance tradeoff
- Replication

4. Scalability

- Change allocation of components to hosts without changing architecture or components
- *Transparency*
 - Whether or not a particular aspect of a system's architecture is visible externally

Kinds of Transparency

- Access transparency
 - Local / remote
- Location transparency
 - Physical
- Migration transparency
 - Load balancing
- Replication transparency

5. Heterogeneity

- Hardware
 - Including embedded devices
- Operating systems
- Programming languages
- Standards / protocols / APIs
- Access mechanism
 - Web browser
 - Device
- Versioning

Implications of Heterogeneity

- Standard APIs
 - W3C, OMG, ANSI, ISO
- Normative architectures
 - MDA (OMG: Platform independent and dependent models; code generation)
- Vendor comprehensive solutions
 - SUN (J(2)EE), Microsoft (.NET), IBM (WebSphere)

Other Non-Functional Issues

- Fault tolerance
- Flexibility
- Reuse / productivity
- Complexity
- Quality of service (e.g. tradeoff between performance and reliability)

Kinds of Middleware

(Based on interaction mechanism)

- Distributed transactions
 - Transactional middleware
- Message passing
 - Message-oriented middleware
- Remote procedure call
 - Procedural middleware
- Remote object requests
 - Object/component middleware

Transactional Middleware

- Two-phased commit
- Products
 - CICS (IBM mainframe)
 - Tuxedo (UNIX-based)
 - Encina (HP)
- Distributed client/server; location transparency

Message-Oriented Middleware (MOM)

- Asynchronous
- Products
 - IBM's MQSeries
 - SUN's Java Message Queue
- Distributed event notification; publish/subscribe
- Fault tolerant because of queues
- Not very transparent
 - Clients must implement coordination

Procedural Middleware

- Remote procedure call (RPC)
- SUN (Open Network Computing)
- NDR (network data representation)
- Synchronous
- Operating system dependent
 - Open standard

Object and Component Middleware

- Extension of RPC
 - Object identity
 - Inheritance
- Products/approaches
 - CORBA - Mainframes
 - COM - PCs
 - SUN - Java RMI
- Marshalling (e.g. *serializable*)
- Synchronous / asynchronous
- Exception handling

Software Engineering Issues

- Requirements engineering
 - Elicitation of non-functional requirements; QOS; quantification
- Software architecture
 - Derive architectures that will address non-functional requirements
 - Relationship of connectors to middleware
- Design issues
 - Network latency for distributed interactions
 - Activation, statefulness, persistence
 - Concurrency
 - Synchronization, deadlock avoidance, liveness

Research Questions

- Naming (white pages) vs. trading (yellow pages)
- Use of reflection, meta-object protocols
- Object-specific replication strategies to improve scalability
- Scheduling & prioritization
- Embedded systems / memory limitations
 - Power consumption; clock cycle
- Real time
- Mobility (unreachability)

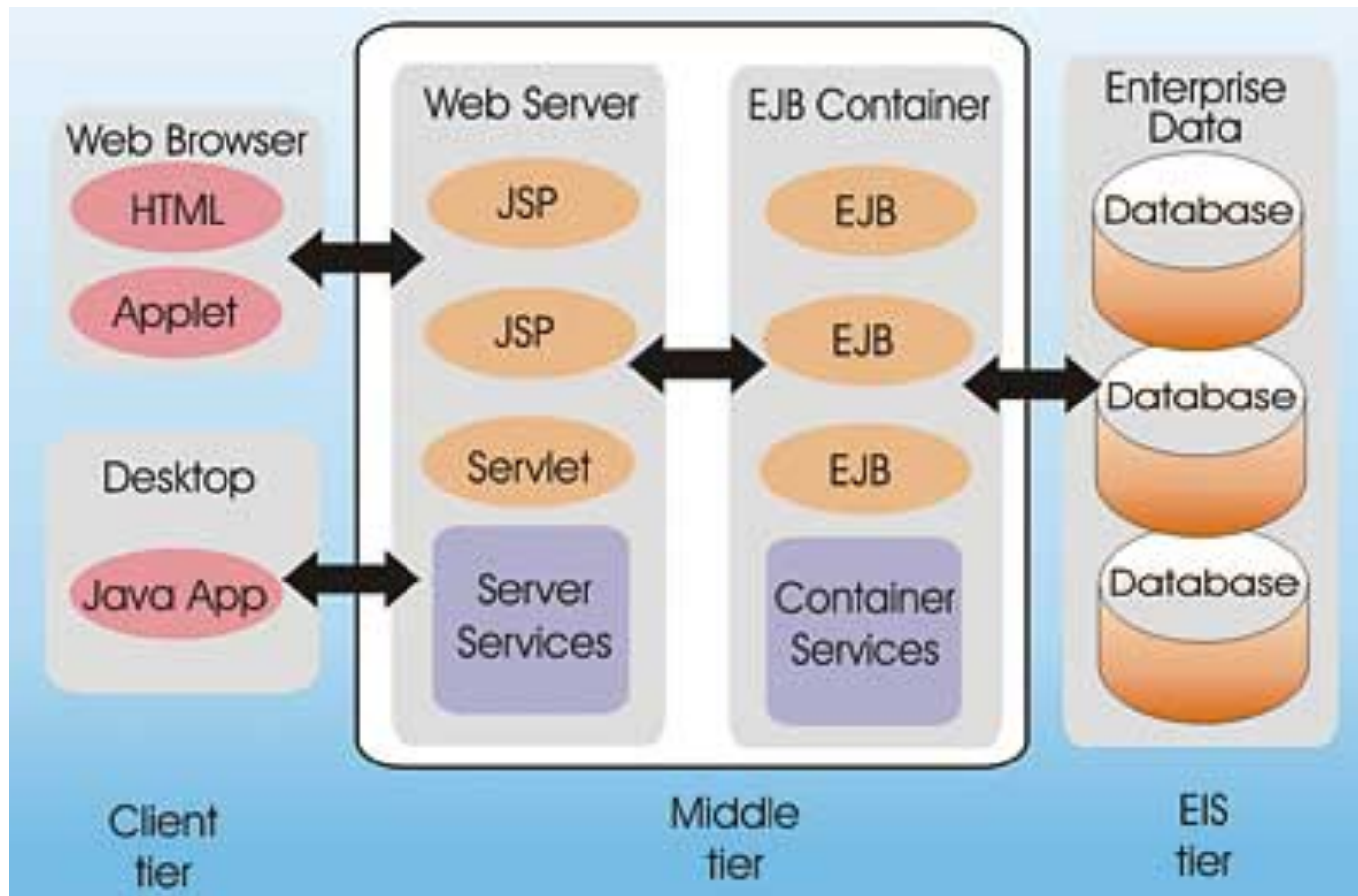
Web Services

- "A software system designed to support interoperable Machine to Machine interaction over a network" -- W3C
- Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services -- Wikipedia

Web Services Protocols

- XML (Extensible Markup Language) - self-defining data descriptions
 - SOAP (Simple Object Access Protocol)
 - RDF (Resource Description Framework)
 - OWL (Web Ontology Language)
- WSDL
 - Web Services Description Language
- UDDI - directory services; “Yellow Pages”
 - Universal Description, Discovery and Integration

J2EE System (SUN)



Service Oriented Architecture (SOA)

- *A computer systems architectural style for creating and using business processes, packaged as services throughout their lifecycle* -- Wikipedia

Services

A service, in the context of service oriented architecture, is a self-contained, self-defined, modular application: with a defined and self-documented use protocol; that is “useful” in that it directly relates to the job responsibilities or business needs of the service user; that can be comprised of a suite of sub-services; that is published, located, and dynamically invoked

Service Orientation

- Operations: transactions representing single logical units of work (from the point of view of the customer)
- Services: Logical groupings of operations
- Business processes: long-running set of activities for a specific business goal; ordered sequence of services (*service choreography*)

Characteristics of Services

- Flexibility; factoring
- Meaningful; self-contained; user oriented
- Stateless?
 - But can make use of a database
- Transparent with respect to middleware