



**Distributed Software Architecture
Using Middleware**

Mitul Patel

 **UCL ComputerScience** 1

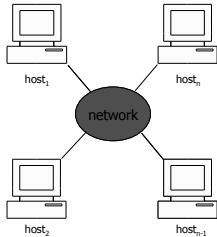
Overview


- Distributed Systems
- Middleware
 - What is it?
 - Why do we need it?
- Types of Middleware
- Example
- Summary

 **UCL ComputerScience** 2

Distributed Systems



- Components of a system are not always held on the same host
- Hosts are connected via a network
- Appears to users as a single integrated computing facility



 **UCL ComputerScience** 3

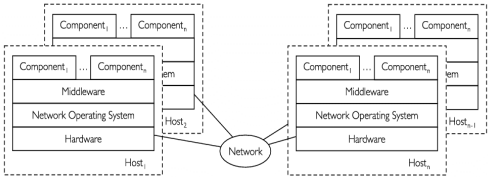
Distributed Systems



- **Positives**
 - Increased resource sharing
 - Better integration of existing system components
 - Increased performance & reliability
 - Cheaper than a centralised system (sometimes)
- **Negatives**
 - Difficult construction
 - Complicated communication through a network
 - Security issues
 - Differing platforms and languages in use



4

Middleware



- Middleware is a term that refers to a set of software services that reside between the application and the OS and aims to facilitate the development of distributed applications by abstracting over the complexity and heterogeneity of the underlying environment.





5


Middleware

- Provides high-level primitives that simplify distributed system construction
- Common environment across platforms and languages
- Consistent Application Programming Interface (API)
- Supports interoperability across OS, hardware, networks, and programming languages
- An abstraction for application designers/programmers
 - Resolves heterogeneity
 - Enables expert design
 - Simplified interface
 - Provides transparency
 - Reusable
- Concentrates development on requirements and not underlying network protocols, concurrency issues, etc



6

Middleware Requirements


- Quality of Service (QoS)
- Scalability
- Load balancing
- Reliability (fault-tolerance)
- Predictability
- Transparency
- Group Requests
- Security
- Marshalling

 **UCL ComputerScience** 7

Middleware


- However
 - Limits interaction with the operating system
 - Potential performance hit
 - Can sometimes add to complexity and cost

→ May not be appropriate in all circumstances

 **UCL ComputerScience** 8



Types of Middleware

- Transactional Middleware
 - Supports transactions
- Message-Oriented Middleware
 - Supports message exchange
- Procedural Middleware
 - Supports remote procedure calls (RPCs)
- Object/Component Middleware
 - OO version of procedural middleware

 **UCL ComputerScience** 9



Transactional Middleware

- Supports transactions involving components on distributed relational databases
- Assumes servers use 2 phase commit protocol (2PC)
 - Helps keep the system in a consistent state
 - Ensures operations occur on either all or no hosts
 - Reliable
- Communication can be synchronous or asynchronous
- An unnecessary overhead exists if transactions are not needed
- Programmers have to deal with marshalling
- e.g. IBM's Customer Information Control System (CICS)



10



Message-Orientated Middleware

- Supports messaging & notifications between distributed components
- Point-to-Multipoint support
 - Great for publisher-subscriber based systems
 - And for group communication
- Communication via asynchronous message exchange
- Good fault tolerance using persistent message queues
- Programmers have to deal with marshalling
- Limited support for scalability & heterogeneity
- e.g. Sun's Java Message Service (JMS)



11



Procedural Middleware

- Supports remote procedure calls
 - Developed by Sun Microsystems during the 80's
- Uses Interface Definition Language (IDL)
- Synchronous communication
 - Between one client and one server only
- Middleware deals with marshalling and un-marshalling
- Bindings for many different programming languages
- Not very fault tolerant or scaleable
- No single standard
- e.g. Sun Remote Procedure Call (RPC)



12



Object/Component Middleware

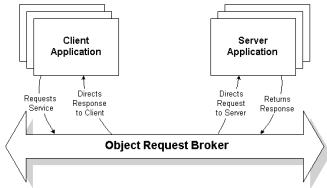
- OO evolution of procedural middleware
 - Adds inheritance, references, exceptions etc
- Uses an Object Request Broker (ORB)
- Communication is synchronous or asynchronous
 - Supports multicasting
- Integrates most of the capabilities of transactional middleware, message_oriented middleware and procedural middleware
- Very powerful, but scalability is still rather limited
- e.g. Common Object Request Broker Architecture (CORBA) (open standard)
Distributed Common Object Model (DCOM) (proprietary)





13

Example Middleware

- *Common Object Request Broker Architecture (CORBA)*, is a distributed object architecture that allows objects to interoperate across networks regardless of the language in which they were written or the platform on which they are deployed
- CORBA allows developers to write applications that are more flexible and future-proof, to wrap legacy systems, and to code in the language they know best
- The *Object Request Broker (ORB)* is the middleware that handles the communication details between the objects. The CORBA 2.0 standard, adopted in December of 1994, defines true interoperability by specifying how ORBs from different vendors can communicate using a common protocol
- In the CORBA model, a client can request a service without knowing what servers are attached to the network. The various ORBs receive the requests, forward them to the appropriate servers, and then hand the results back to the client.



14





15

Summary

- Distributed systems are **great** but there are drawbacks
 - Complex construction
 - Security issues
 - etc
 - Middleware hides many of the complications that arise with building & maintaining distributed systems
 - Resolves heterogeneity
 - Provides transparency
 - etc
- Middleware is **great!**



Further Reading

- Book:
IT Architectures and Middleware:
Strategies for Building Large, Integrated Systems
 - Chris Britton
 - ISBN: 0321246942
- IEEE Computer Society:
 - <http://dsonline.computer.org/middleware>
- Papers:
Software Engineering and Middleware: A Roadmap
 - Wolfgang Emmerich
 - <http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/ICSE2000/SOTAR>
Distributed Component Technologies & their Software Engineering Implications
 - Wolfgang Emmerich
 - <http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/ICSE2002/SOA/>