System Models

Topics for this lecture

- Architectural models
  - Layers
  - System Architectures
  - Interfaces and objects
  - Design requirements
- Fundamental models
  - Interaction
  - Failure
  - Security

System models

- Systems need to be designed
- Models describe common properties and design issues for distributed systems
- Models can take two forms
  - Architectural
  - Fundamental

System models

- **Architectural models**
  - The placement of parts in the system, i.e. Client-server, peer-peer
- **Fundamental models**
  - No global time
  - Communication achieved by message passing
    - Message delays, communication failures, security
  - Three types of fundamental model
    - **Interaction**
      - Performance and timing
    - **Failure model**
      - Specify failures and defines correct process
    - **Security model**
      - Securing communication and processes

Architectural models

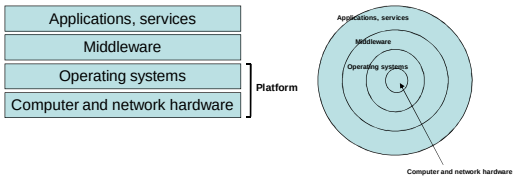
- Major concerns:
  - Manageable, reliable, adaptable and cost-effective
  - One aim is to simplify design
  - Another is the placement of components in a useful manner
  - While considering the interrelationship between components

Software layers

- The structuring of software into modules
- Deals with complexity – divide and conquer
- Delegates responsibility for specific behaviour to particular layers
- A layer should be cohesive
- Allows flexibility as a layer can be replaced by other similar layers

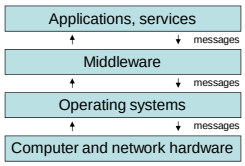
Software layers

Two, abstract ideas of layers that imply the same meaning



Software layers and messages

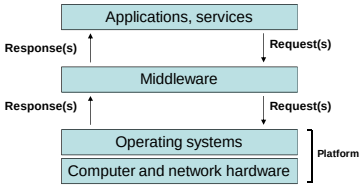
Messaging is required between the layers



Typical software layers

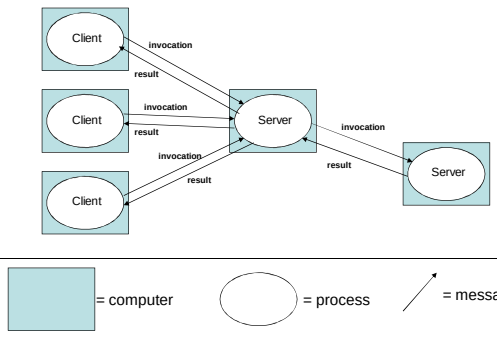
- Platform
  - Low level software & the hardware
    - Operating systems, networks, computer chips, circuits, routers, printers, etc.
- Middleware *(Software whose purpose is to mask heterogeneity)*
  - Sits between the applications, service layer & the platform
  - It abstracts and masks the heterogeneity
  - It acts as a mediator between the application(s) and the platform(s)
  - This helps to reduce dependency as applications depend on middleware not platforms

Middleware layers

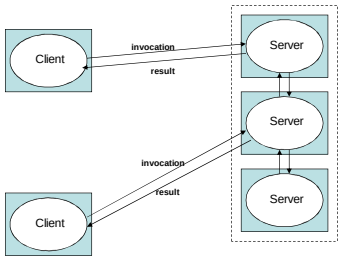


The Middleware layer acts as a mediator that brokers the request from applications, services

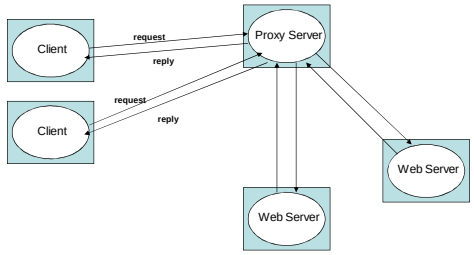
System Architectures



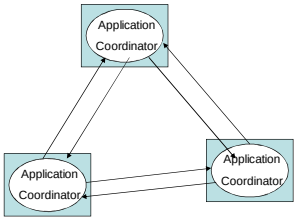
Proxy server



Proxy server



Peer processes



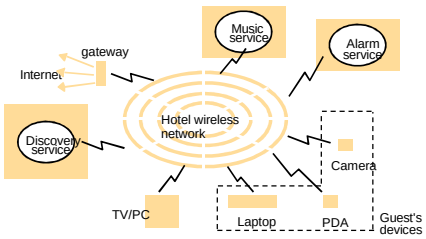
Other client server models

- Thick client
  - If the client performs some of the application logic in addition to presentation
- Thin client
  - The server provides nearly all of the application logic
  - The client merely deals with presentation issues

Other client server models

- Networked computers
  - Download OS and Applications from a server
  - Use low spec clients
  - Increase network traffic
- Mobile code
  - Download from a sever code which is run by clients
  - This code might then require server communication
  - A code way of distributing applications

Spontaneous networks



Interfaces and Objects

- Interfaces define methods that can be used by clients
- Traditional method
  - Clearly defined server processes make available a set of operations (their interface)
- OO method
  - Individual server processes can be invoked
  - This is not static and could be generated at run-time

Design requirements

- Performance requirements
  - Responsiveness
    - We want responsive systems
    - It is affected by network latency, traffic, server queues, the number of software layers
  - Throughput
    - Determines the capacity of the DIS?
  - Load balancing
    - Can we share work to improve performance
    - i.e. the use of replicated services

Design requirements

- Quality of service requirements
  - Requires a reasonable or defined level of performance levels
    - Security and reliability
  - Are services available?
  - Can they deliver the services they claim to?
  - Is the system secure?

Design requirements

- Dependability

Design requirements

- Caching

Fundamental models

- Interaction model
- Failure model
- Security model

Interaction model

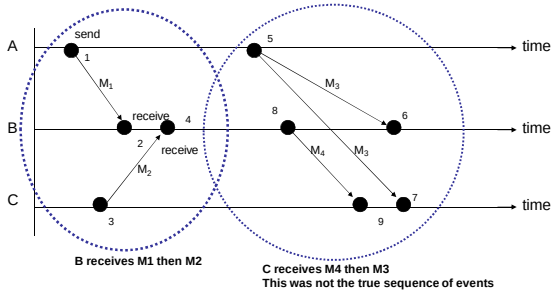
- Aspects of interaction to consider:
  - Many servers interacting, peer processes interacting
  - Clocks and timing
  - Synchronous DS
  - Asynchronous DS
  - Event ordering

Interaction model

- Timing of events
  - Clocks are used to:
    - Define an ordering of events
    - Co-ordinate events at roughly the same real time
    - Permit reasoning about the global state of the system based only on local information.
  - Clocks run at different speeds (clock drift)
  - In a DIS ordering events is more important than synchronising to any real-world clock

Interaction model

- Message order



Interaction model

- Ordering is important
  - M3 = Balance+10
  - M4 = Balance*0.1
  - M3 then M4 = (Balance+10)*0.1
  - M4 then M3 = (Balance*0.1)+10
- Solution is to timestamp
  - But times can be inaccurate
- Solution is to use logic
  - If time is not essential

Interaction model

- Synchronous
  - Attempts to guarantee synchronization
  - Time bounded
    - Processes and communications
    - Clock drift is known and time bounded
- Asynchronous
  - Makes no guarantees re synchronization
  - Not time bounded
    - No bounded communication or processing
    - Clock drift is variable

Failure model

- Aspects of failure to consider:
  - Omission failures
    - Messages/processes were omitted
  - Timing failures
    - Messages / processes did not meet time constraints
  - Reliability
    - Ensuring validity and integrity

Failure model

- Aspects of failure to consider:
  - Arbitrary failures
    - Unknown failures that occur seemingly at random
  - Failure transparency
    - Failures can be hidden from the user

Security model

- Aspects of security to consider:
  - Protecting data
    - Access rights, access levels, authorisation
  - Enemies and threats
    - Mainly re messaging
    - Disclosure, manipulation, denial of service, etc.
  - Security mechanisms
    - Cryptography, certificates, secure channels, authentication servers, etc.

Summary

- Architectural models
  - Layers
  - System Architectures
  - Interfaces and objects
  - Design requirements
- Fundamental models
  - Interaction
  - Failure
  - Security

Referenced material

- Distributed Systems: Concepts and Deisgn, *George Colouris, Jean Dollimore, Tim Kindberg, Addison-wesley, Forth Edition, 2005, ISBN*
  - *Power point slides, text quotations, examples and diagrams*
- Distributed Operating Systems, *A.S.Tanenbaum, Prentice Hall, 1995, 0-13-143934-0*
  - *Diagrams and examples*

END