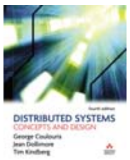
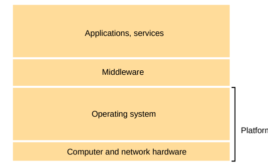
Slides for Chapter 2: Architectural Models



From Coulouris, Dollimore and Kindberg
Distributed Systems:
Concepts and Design
Edition 4, © Pearson Education 2005

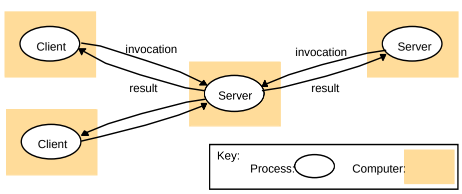
Software Layers

- Platform: hardware, OS
- Middleware
  - masking heterogeneity of underlying platform, hence providing a common "platform"
  - Examples:
    - Sun RPC, Java RMI
    - CORBA, Microsoft .NET, Java J2EE
  - reliable services in the middle layer, still need application-specific reliability
- Applications, services



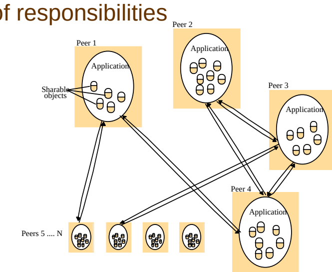
Client-Server

- Servers provide services
- Clients access services



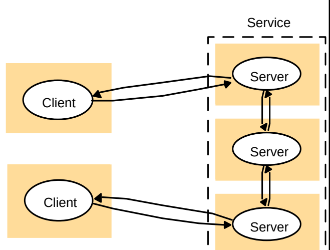
Peer to Peer

- Peers play similar roles
- No distinction of responsibilities



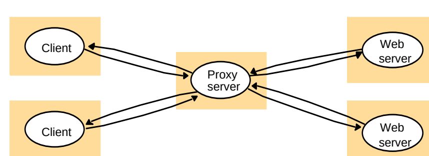
A service provided by multiple servers

- Objects are partitioned/replicated
  - Web: each server manages its objects
  - NIS: replicated login/password info
  - Cluster: closely coupled, scalable (search engines)



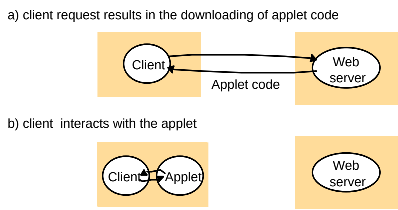
Web proxy servers and caches

- Cache: local copies of remote objects for faster access
- Browser cache
- Proxy server
  - Additional roles: filtering, firewall



Mobile code and web applets

- Running code locally vs. remotely
  - Network bandwidth
  - What examples have you seen?
  - Security issues?



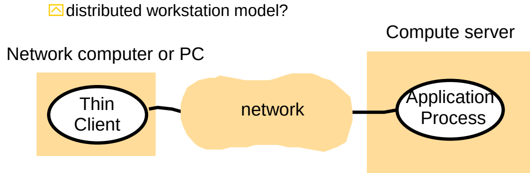
Mobile agents

- Running program that travels from one computer to another
  - Perform tasks on users' behalf
  - Security issues
- How does it compare to one client interacting with multiple servers?
  - What if the program needs to access a lot of remote data?
- How does it compare with applets?
- Agents can provide functionality a remote web site does not provide, but allows access to data.

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Thin clients and compute servers (network computers)

- Thin client
  - Basically a display with keyboard ("dumb terminal")
  - Remote computation, storage
- How does it compare with the
  - PC model or
  - distributed workstation model?



Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Mobile devices and spontaneous interoperation

- Laptops, PDA's, cell phones, wearable computers…
- Metropolitan (GSM, CDPD): hundreds of meters, 100s Kb/s
- Local Area (BlueTooth, infra-red): meters, 10 Mb/s
- Infrastructure vs ad-hoc networking
- Accessing services, variable bandwidth/connectivity, power supply, security,
- Spontaneous interoperation--easy connection and location of services
  - Service discovery
  - Context-aware

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Interfaces and objects

- At the programming level
  - Need standard interfaces
    - Access/provide services/objects
    - RPC/RMI

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Design requirements for distributed architectures

- Performance
  - Responsiveness, throughput, load balancing
- Quality of service (QoS)
  - time-critical applications (real-time apps)
  - guarantee certain level of quality delivered by the deadline
  - allocation of computation and communication resources
- Caching and replication
  - web caching: server provides expiration time
- Dependability
  - fault tolerance: redundancy, recovery
  - security

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Fundamental Models

- ⌘ Fundamental properties in processes and communication, shared among different architectures discussed previously

Interaction model (1)

- ⌘ sequential vs. distributed algorithms timing, distributed state
- ⌘ performance of communication channels
  - latency: transmission, access, os
  - bandwidth
  - jitter: variation among messages
- ⌘ clocks and timing events
  - clock drift
  - synchronization

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

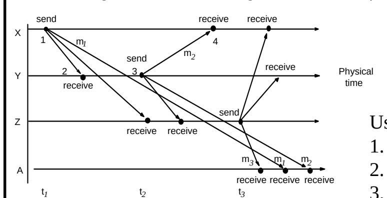
Interaction Model (2) [Synchronous vs Asynchronous]

- ⌘ synchronous distributed systems
  - lower and upper bounds for execution of a step
  - message transmission in bounded time
  - clock drift rate is bounded
  - failures can be detected when bounds are exceeded
  - accomplished by allocating sufficient resources
- ⌘ asynchronous distributed systems
  - no bounds on process speed, message delay, clock drift rate
  - failures are harder to detect
  - performance can't be guaranteed

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Interaction Model (3)

- ⌘ event ordering
  - Relative ordering might be more important than exact time
  - logical clocks--ordering events without physical clocks



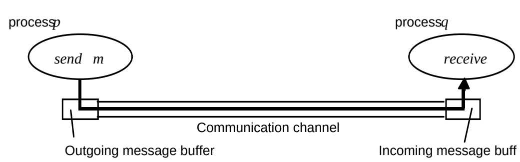
User A:

1. From Z: Re: Meeting
2. From X: Meeting
3. From Y: Re: Meeting

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Failure Model (1)

- ⌘ Failure of processes or communication channels



Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Failure Model (2): Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4
© Pearson Education 2005

Failure Model (3): Timing failures

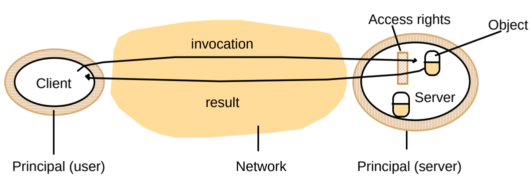
Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Failure Model (4)

- masking failures
  - hiding--use another server to respond
  - converting it into more acceptable--drop the packet if it is corrupted
- reliable one-to-one communication
  - validity: eventually delivered
  - integrity: content not corrupted or duplicated

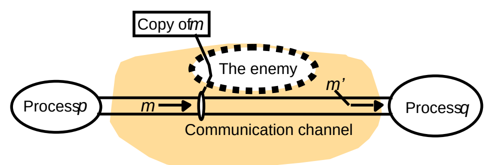
Security Model (1)

- Protecting objects:
  - authorization (access rights to principals)
  - authentication (identity of parties/principals)



Security Model (2)

- Threat to processes & communication channels



Security Model (3)

- Threat to processes & communication channels
- Denial of service
- Mobile code
- Cryptography: science of keeping messages secret
  - encryption: process of scrambling a message to hid its content
  - secret keys--large numbers that are difficult to guess
  - authentication--encrypt the identity, check the decrypted identity
  - secured channels--authentication, privacy/integrity, time stamp to prevent replaying and reordering