Characterisation of a DIS

Topics for this lecture

- Understand the key characteristics of distributed systems
- Be able to explain terms such as Openness, Reliability, Scalability etc.
- Understand the concept of Transparency
- Explain the importance of Transparency as a measure of the quality of distributed systems

What is a DS?

- Definition of a Distributed System

"One in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages."

[Coulouris et al, 2001]

Examples of a DS

- Networks are everywhere
  - Internet
  - Mobile phone networks
  - Corporate factory networks
  - Campus networks
  - Home networks
- As such distributed systems are everywhere

Why use a DIS?

- Simple -- to share resources
  - *"okay.. So what is a resource?"*
  - A resource is something that can useful be shared over a computer network
- A resources could take the form of:
  - Hardware components
    - Disks, printers, processing, etc.
  - Software entities:
    - Files, databases, objects, etc.

Major issues with DS

- Concurrency
  - Work is done at the same time on many computers
  - This is the notion of concurrency.
  - How do we deal with this?
- No global clock
  - Cooperation between computers occurs through messages
  - Close coordination depends on timing
  - However there are limitations on clock synchronization over networks
  - How do we deal with this?
- Failures
  - Systems / computers can fail, networks can fail and run slow
  - How do we cope with failure?
  - How do we minimise its effects?

Examples of DS

- The WWW (a huge DS)
  - Essentially its a series of interconnected computers of many different types
  - They can operate together through message passing
  - The nature of the WWW is that it has an open structure
    - Can add additional resources easily
  - Intranets are connected together via backbone
    - A high transmission link, i.e. satellite, optic fibre, etc.

Examples of DS

- Intranets
  - Typically a portion of the internet administered separately and with a boundary
  - Intranets are composed of several LANs linked by backbone connections
    - Could be a single site or many sites
    - Could be a single company or many companies
  - Intranets can be connected to the internet
    - But are not always, military, police, etc.
    - When connected firewalls can protect against unauthorised access

Examples of DS

- Mobile and ubiquitous computing
  - Laptops, handheld devices (PDAs), mobile phones, Cars, hi-fi, etc.
  - Mobile, e.g. computing whilst moving.
    - Devices that access networks (internet, intranet, etc.)
  - Ubiquitous, e.g. computers everywhere
    - Small computing devices that exist in all places, home, work, computers are getting everywhere all the time
    - Useful when they are tied together through networks (wired and wireless)
  - Mobile too can in principle use ubiquitous computing

Some key terms

- Jargon: A Service

"A distinct part of a computer system that manages a collection of related resources and presents their functionality to users and applications"

[Coulouris et al, 2001]

  - Examples,
    - Access shared files through a *'file service'*
    - Send files go to a printer through a *'print service'*
    - Buy good through a *'payment service'*
  - In each case we access the service through its exported set of operations, i.e. its interface

Some key terms

- Jargon: a process

"A running program (a process) on a networked computer that accepts requests from programs running on other computers to perform a service and responds appropriately"

[Coulouris et al, 2001]

Some key terms

- Jargon: client server
  - Typically a Request messages come from a client to a server
    - The client is said to *invoke* an operation on the server
    - A complete interaction between the client and the server is known as a *remote invocation*
  - A server might make a request of other servers
    - Thus a server could also be a client of another server
  - The term Client or Server reflects the role during transactions that occur as part of running processes

Key concept: Heterogeneity

- Heterogeneity refers to variety and difference
  - *Networks*
  - *Computer hardware*
  - *Operating systems*
  - *Programming languages*
  - *Implementations by developers*
- Communication between heterogeneous systems occurs through agreed common standards that each communicator must adhere to
- This is commonly referred to as **MIDDLEWARE**

Key concept: Middleware

- So what is middleware?
  - Middleware is software layer of abstraction
  - Middleware masks heterogeneity
  - Using it you can make remote calls to distributed resources without worrying where they are physically located
  - Systems talk to the middleware rather than other systems
    - Examples, CORBA, Java RMI
  - Most middleware runs over the internet, why?

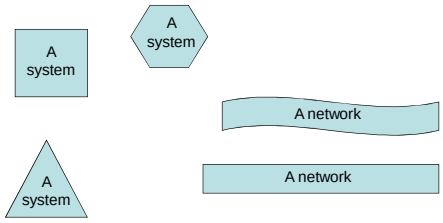
Key Concept: Openness

- What is it and why have it?
  - Openness is the characteristic that determines whether the system can be extended and re-implemented in various ways
  - Openness requires published interfaces (API)
  - Uses a standard interface, agreed, used by all
  - The interface is the means of linking the different systems together

Key concept: Interfaces

- Interfaces

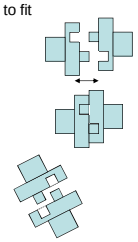
Systems (software and hardware) can be different so how do they fit together



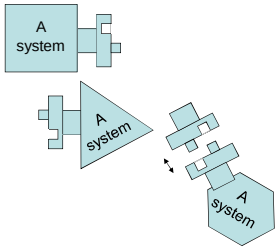
Key concept: Interfaces

Interfaces: are based on agreed standards

An interface that plugs into other similar interfaces needs to fit

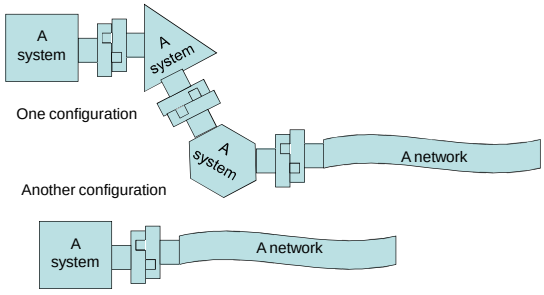


Develop systems based on the interfaces



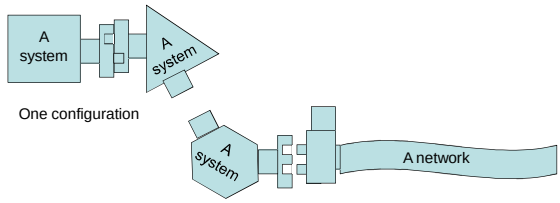
Key concept: Interfaces

Then we can join systems (hardware or software) more easily



Key concept: Interfaces

!!IMPORTANT!! And the interfaces need not all be the same!!



They just need to match with who they are going to connect to

Key Concept: Openness

- Examples:
  - The internet and the Web?
    - Published Internet communication protocols, see **www.ietf.org**
    - So developers can use the published protocols to write software and develop hardware
- An Open System
  - Open systems are characterised by their **key interfaces being published**
  - Open DS are based on a **uniform communication** mechanism
  - Open DS can be constructed from heterogeneous hardware, software, etc. However, **conformance to published standards** must be tested and verified

Key Concept: Security

- We have three types of security to consider:
  - **Confidentiality** (protection against disclosure to unauthorized individuals)
  - **Availability** (protecting against interference with the means to access the resource)
  - **Integrity** (protection against alteration or corruption)
- Firewalls can help to secure intranets
- But…
  - How do we secure within the intranet?
  - How do we also secure accessing resources over the internet some of which might not be protected by firewalls?

Key Concept: Security

- Consider, a communication from a client to a server
- What do we need to secure?
  - We not only need to secure the details of the message itself from eavesdroppers
    - We also need to consider senders and receivers are valid
    - We also need to ensure the contents remain the same
    - We also need to ensure that messages are not replayed
    - And more..
- Encryption techniques can provide this capability in modern systems.

Key Concept: Scalability

- A scalable system remains effective once it increases in terms of resources and user
- The internet has increased in size dramatically recently, see below:

Computers in the Internet

	Computers	Web servers
1979, Dec	188	0
1989, Dec	130,000	0
1999, Dec	56,218,000	5,560,866
2003, Jan	171,638,297	35,424,956

[Coulouris et al, 2005]

Key Concept: Scalability

- Key challenges
  - Controlling the cost of physical resources
  - Controlling the performance loss
  - Preventing software resources running out
  - Avoiding performance bottlenecks

Key Concept: Scalability

- Key challenges:
  - Controlling the cost of physical resources
    - Does adding more hardware increase available resources,
    - Does… $2\ server_{performance} = 2^*\ 1\ server_{performance}$
  - Controlling the performance loss
    - Does increases in size lead affect performance linearly or non-linearly
    - Ideally performance should be no-worse than $O(\log n)$

Key Concept: Scalability

- Key challenges (cont..):
  - Example, the 32bit addresses used of the Internet will run out this decade, solution use 128bit addresses
  - Its better to plan ahead that have to make retrospective changes!
  - Avoiding performance bottlenecks
  - A bottleneck is a constriction that reduces the flow of traffic through a given system (fluid flows from the bottle as fast as the neck allows)
  - The Internet used a single file to store references before DNS (Domain Naming Services) were developed,
  - This was okay while small but as the number grew it became a bottleneck

Key Concept: Failure handling

- What is a failure?
  - A failure is something that can halt programs, stop operations and introduce errors
- In a DS a failure is **partial**
  - It affects some of the resources and part of the DS system
  - Some parts of the system will carry on functioning
  - This can create some problem as parts can become inconsistent

Key Concept: Failure handling

- **Detecting failures**
  - Some failures can be detected (checksums, etc.)
  - Some failures cannot be detected (internet server crashes)
  - Regardless we need to plan and manage all types of failures
- **Masking failures**
  - Some can be hidden (i.e. a message that does not reach its recipient can be resent, RAID arrays duplicate data, etc.)
  - Implications, speed, what happens if both disks fail?

Key Concept: Failure handling

- **Tolerating faults**
  - We can tolerate failures and inform processes and users
  - We need to be clear how we do this
  - How also can we reconstruct after failures?
- **Redundancy**
  - One method is through redundancy (redundant components)
  - Examples, *DNS replications, multiple routes, database replicata's*
  - But his raises issues concerning the consistency of these duplicated services
  - How to we ensure correctness?

Key Concept: Concurrency

- **Concurrency**
  - Access to the same resource by two or more processes at the same time
  - Sounds great! But how do we deal with it?

Key Concept: Concurrency

- **Concurrency**
  - Access to the same resource by two or more processes at the same time
  - Q .How might we deal with this?
    - We might take each process one at a time (serialize)
    - This limits throughput though
    - We need better ways of managing concurrency

Key Concept: Transparency

A definition of transparency

"The concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components"

[Coulouris et al 2005]

Key Concept: Transparency

- 8 forms of transparency:
  - Access transparency
  - Location transparency
  - Concurrency transparency
  - Replication transparency
  - Failure transparency
  - Mobility transparency
  - Performance transparency
  - Scaling transparency

Key Concept: Transparency

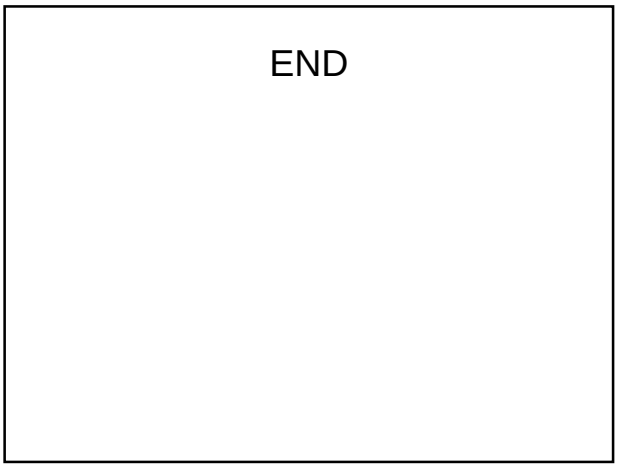
- Access transparency
  - Local and remote resources can access using the same identical operations
- Location transparency
  - Resources can be accessed without knowing their location
- Replication transparency
  - If multiple copies of resources are used replication transparency hides these details from the user so that users do not need to concern themselves with which replica is being accessed
- Concurrency transparency
  - Processes can access resources without worrying about other concurrent processes

Key Concept: Transparency

- Mobility transparency
  - Resources and clients can be moved without affecting the operations of other users and programs
- Failure transparency
  - Faults are concealed such that applications can continue without knowledge that a fault has occurred
- Performance transparency
  - The performance of systems should degrade gracefully as the load on the system increases Scaling transparency
- Scalability transparency
  - it should be possible to scale-up an application, service or system without changing the system structure or algorithms

Summary

- Understand the key characteristics of distributed systems
- Be able to explain terms such as Openness, Reliability, Scalability etc.
- Understand the concept of Transparency
- Explain the importance of Transparency as a measure of the quality of distributed systems


END