Distributed Systems

Prof. Steve Wilbur
Department of Computer Science
University College London

1

Distributed Systems

... "use of more than one computer connected by communications links to carry out a computational task"

... includes "parallel" computing

... most contemporary professional systems

2

Course Objectives

- *to identify special characteristics of distributed systems*
- *to identify ways user/programmer view can be simplified*
- *to identify protocols, techniques and algorithms which address DS issues*

Why Distribute?

- *resource sharing*
- *physical separation*
- *robustness*
- *performance*
- *cost-effective system evolution*

Examples

- *file server, print server, etc.*
- *networked management*
- *directory and naming services*
- *desktop (multimedia) conferences*
- *large-scale computation*
- *etc.*

Why are DSs Different?

- *non-zero (and variable) message transmission time*
- *probability of partial failure of collaborating components*
- *large scale*
- *environment is not totally secure*

Message Transmission Time

- *processors executing in parallel*
- *multi-path links between them*
- *same message may arrive at different times at different processors*
- *possibility of "race" situations, i.e. non-determinism in algorithm execution (or bugs)*
- *consistency of data at different sites*

Partial Failures - 1

- *not a concept found in single processor programs*
- *recovery from failures*
  - *use of alternative processor*
  - *migration of services*
  - *update and propagation of naming/routing information*

Partial Failures - 2

- ***Fault tolerant algorithms***
  - ***leadership elections***
  - ***distributed synchronisation***

Scaling

- ***Abstraction***
- ***Inheritance/object orientation***

Security

- *computer hosts may be protectable via architecture and OS etc.*
- *... but link is very vulnerable to*
  - *tapping/message reply*
  - *hosts faking addresses*
- *... and hosts vulnerable to*
  - *deliberate message overload*
  - *faked services*

Distributed Programming

- *Can we arrange for programmer's view of DS to be identical to that for single-processor system?*
- *Simple answer is "no", but we can do quite a lot*

Remote Procedure Call - 1

- *common approach*
- *client-server model*
- *caller's arguments are marshalled by "stubs", put in a packet and sent to remote proc.*
- *problems with memory addresses - need args to be sent by "value"*

Remote Procedure Call - 2

- *execution semantics when failure*
  - *at-least-once (cheap and easy but ...)*
  - *at-most-once (relatively easy)*
  - *exactly once (expensive protocol)*
- *how is server located?*
  - *name server based on type matching*
  - *trader includes other attributes e.g. location of device, options, etc.*
  - *version management too*

Remote Procedure Call - 3

- *parameter security*
  - *may need to pass encryption keys or other security tokens as parameters*
- *Interface Definition Language (IDL)*
  - *indicates type and order of parameters*
  - *signature for matching in name server*
  - *possibly security indication for stubs*

Real World

- *often termed "Middleware"*
- *support by environments such as DCE, CORBA etc.*

Transparency

- *... is desirable, but may not be achievable:*
  - *distribution - effects of distribution (delays etc.) should be invisible to user*
  - *location - location of components should be irrelevant*
  - *migration - remote objects may relocate during use*
  - *failure - recovery from faults*

17

Security

Use of cryptographic techniques to deal with:

- *secrecy*
- *authentication of individuals and messages*
- *replay and faking attacks*
- *can also provide access control to object methods*

18