CMSC433, Fall 2002 Programming Language Technology and Paradigms Distributed Computing

Adam Porter

March 14, 2002

Is distributing computing different?

- What kinds of distributed computing environments exist?

  Ways in which distributed computing is

  – Addressing objects
  – Latency
  – Partial failure
  – Concurrency

Distributed computing environments

- Usually refers to multiple CPU's

  communication via shared address space or message passing?

  On same chip, in same room, or across the

  – latency, failure modes

Existing environments

- Seti @ Home, Entropia
- Server for search engine
- My laptop, PDA, cell phone, MP3 player, digital camera, …

Types of failure

- Machine sleeps
  – wakes up, recovers state
- Machine crash or failure
  – machine may reboot and rejoin
- Network partition
  – network may heal

Uniform view of distributed objects

- Some objects are remote, some are local
  – Doesn't really matter to user of object

  Objects might transparently migrate
- Design doesn't have to take object distribution

  Failure and performance issues don't belong in the

  The interface doesn't change if an object is remote

Uniform view

- Not appropriate for
  - wide area networks,
  - consumer electronics,
  - portable devices
- Appropriate for some local area networks
  - but robust distributed applications plan for

    even if all objects local
    means in interfaces, not just implementation

Memory access

- Can we make the fact that an object is remote transparent?

  Perhaps for objects
  - What about int's ?
  - What about char *'s?
- If you can't directly access fields and create

  - then it's not transparent

Partial failure

- Computers fail
- OS's crash
- Networks fail
- PDA's get turned off or taken out of the

  Often no warning, and each hardware/software component can fail independently of all others
  - and no other component may be able to tell exactly what happened

Queue example

- Want to add x to remote queue q
  - q.enqueue(x)
- Operation could fail
- Want to reliably enqueue x

Queue example

```
while (true) {
    try {
        q.enqueue(x);
        break;
        }
    catch (RemoteException e) {}
}
```


- Object was enqueued, but failure occurred during return message
- Could enqueue x multiple times
- Solution?
  - Need a request tag so that duplicate requests can be detected
- Real question here is how much does it cost to make the remote call look the same as a

Concurrency

- Distributed computations mandates
  - objects at different locations can't be stopped from executing concurrently

    even less control than with multi-threading
    - no single point of control (hardware, OS, thread

Latency

- Making a call to an object on a remote machine is much more expensive than a
  - several orders of magnitude
- Leading to overall system performance problems if there are "too many" remote