Last time

- RMI
  - Stubs
  - Remote objects and interfaces
  - RMIregistry
    - for bootstrapping (Naming.lookup(), Naming.bind())
  - Centralized chat server example
    - Code is available on Lectures web page

Is distributing computing different?

- What kinds of distributed computing environments exist?
- Ways in which distributed computing is different
  - Addressing objects
  - Latency
  - Partial failure
  - Concurrency

Distributed computing environments

- Usually refers to multiple CPU’s
- Interprocessor communication via shared address space or message passing?
- On same chip, in same room, or across the Internet?
  - latency, failure modes

Existing environments

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

Administrivia

- Project 4
  - due next Thursday, April 4
  - extra 24 hours if Maryland beats Duke
  - another 24 hours if they win the final
- Commentary for Project 2 due today, 6PM
- Commentary for Project 3 due Tuesday, April 10
  - programs sent out today
- Read CORBA section in Eckel, pp. 980-989 and distributed computing paper (link on Readings web page)
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 into account
- Failure and performance issues don’t belong in the design
- 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 failure
  - 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 pointers to them
  - then it’s not transparent

Partial failure

- Computers fail
- OS’s crash
- Networks fail
- PDA’s get turned off or taken out of the room
- 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) {} 
}

Partial failure

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

Concurrency

- Distributed computations mandates concurrency
  - 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 scheduler, etc.)

Latency

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

CORBA

- CORBA provides language-independent remote procedure call (RPC) capability
  - e.g., client/server implemented in Java/C++
- Not a language feature
  - an integration technology
  - software systems can be implemented to be CORBA-compliant
- Very complex spec
  - current architecture and specification document is 712 pages - see www.omg.org
Fundamentals

- Object Management Architecture (OMA)
  - spec for object interoperability
  - Core Object Model
    - what is an object, an interface, an operation, etc.
  - OMA Reference Architecture
    - underlying infrastructure to allow objects to interoperate
    - includes Object Request Broker (ORB), Object Services

ORB

- Communication process through which objects request services from other objects
  - provides location independence
  - naming service
  - everything needed to set up and then perform an RPC
    - to connect the client to the server

Interface Definition Language

- IDL is a language-independent way to specify data types, attributes, operations (methods), interfaces, etc.
  - syntax similar to C++ or Java
  - includes inheritance (with :, like C++)
  - IDL compiled by language-specific compiler into stubs and skeletons for that language
    - to marshal/unmarshal arguments and return values, and implement the remote call

IDL (cont.)

<table>
<thead>
<tr>
<th>CORBA IDL</th>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Package</td>
<td>Namespace</td>
</tr>
<tr>
<td>Interface</td>
<td>Interface</td>
<td>Pure abstract class</td>
</tr>
<tr>
<td>Method</td>
<td>Method</td>
<td>Member function</td>
</tr>
</tbody>
</table>

Naming service

- Like the registry in RMI
  - a component of the OMA
- CORBA objects accessed through references, just as in Java
  - Naming service provides a way to map a string to an object reference (and vice versa)
    - like java.rmi.Naming.lookup()
    - runs as a separate process, like RMIClient
- CORBA example in Eckel

CORBA vs. RMI

- Both support RPC
- But CORBA makes possible RPC between objects implemented in any language
  - that has an IDL compiler implemented
- An alternative to CORBA is to wrap a Java object around the non-Java code, then use RMI
  - requires wrapper to be able to connect to the non-Java code, for example via JNI (Java Native Interface)
  - then don’t need an ORB