Slide 1

Administrivia

- Homework 4 due Wednesday.
- One more thing to do — small-scale project. Info on the Web tomorrow.

Slide 2

Distributed-Memory Programming in Java Using Sockets

- Based on client/server model.
- Server sets up “server socket” specifying port number, then waits to accept connections. Connection generates socket.
- Client connects to server by giving name/IPA and port number — generates a socket.
- On each side, get input/output streams for socket. Program must define protocol for the two sides to communicate.

Slide 3

Distributed-Memory Programming in Java Using RMI

- Motivation — for client/server applications, can be annoying to have to design your own protocol.
- Instead, idea is to define "remote objects" that can be treated (at program level) like any other objects — invoke methods.
- Typical use in client/server program:
  - Server creates some remote objects and "registers" them.
  - Clients look up server's remote objects and invoke their methods.
  - Both sides can pass around references to other remote objects.

Slide 4

Distributed-Memory Programming in Java — Example

- Example — simplified generic master/worker program, similar to the versions in OpenMP and MPI. (Look at those, briefly.)
- Version using sockets is relatively straightforward — server creates a new thread for each client, only tricky bits are in making sure things are shut down properly. Notice use of `synchronized` in code to ensure thread-safe access to shared variables.
- Version using RMI is also straightforward, again except for code to shut down properly. Notice use of `synchronized` in code to ensure thread-safe access to shared variables; experiment suggests that RMI may use multiple threads to process concurrent requests.
- (Caveat: These programs were developed under Java 1.5 so do not necessarily reflect best practice for later releases.)

Slide 5

Java RMI — A Short How-To

- Define a class for remote objects:
  - Define interface that extends `Remote`
  - Define class that implements that interface, instances of which can be remote objects. (So, either have the class extend a remote-object class or use a static method of a remote-object class.) Notice that the class can also include other methods, only available locally.
  - Write code using classes — if using as remote object, reference interface; otherwise can reference class.
- (Continued . . .)

Slide 6

Java RMI — A Short How-To, Continued

- Compile as usual.
- Make `.class` files network-accessible. (There are other options, but this is simplest.)
- Start `rmiregistry`.
- Run server and clients as usual.

Slide 7

Distributed-Memory Java and *Implementation Mechanisms*

- Very similar to MPI, really — UE management is outside the scope of the libraries, synchronization is implicit. For sockets, communication is explicit; for RMI, implicit.

Slide 8

Minute Essay

- None — sign in.