

Slide 1

Administrivia

- Reminder: Homework 7 code due today. Homework 8 design nominally due today, but accepted through Thursday at 11:59pm without penalty. Homework 8 due the day of the final. Extensions past then, or past Thursday for other homeworks, only under very unusual circumstances.
- Review sheet for final on Web.
- Questions about grading? (Grades and comments on homework coming by e-mail soon.)
- Office hours this week announced by e-mail yesterday.

Slide 2

Networking Basics

- Inter-computer communication based on layered approach and “protocols”:
 - Application level — HTTP, FTP, telnet, SMTP, POP, IMAP, NTP, etc., etc.
 - Transport level — TCP (Transmission Control Protocol), UDP (User Datagram Protocol).
 - Network level — IP (Internet Protocol — addressing, routing of packets).
 - Link level — device drivers, etc.
- Messages are routed to
 - A machine (“host”), identified by IPA or name.
 - A process, identified by “port number” (16 bits). 0 — 1023 are “well-known ports”, others available for applications.

Networking Basics — TCP and UDP

- UDP — independent messages, no guarantees about reliability or message order — analogous to (snailmail) letter.
- TCP — point-to-point channel, guarantees reliability and message order — analogous to phone call. Endpoints called “sockets”.

Slide 3

Networking in Java

- Classes for communicating at application level — e.g., URL (“show URL” example).
- Classes for communicating at network level:
 - TCP — `Socket`, `ServerSocket`.
 - UDP — `Datagram*`.
- RMI (Remote Method Invocation).

Slide 4

Slide 5

Networking in Java — Sockets

- 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.
- Simple example in binary-I/O program from last week.

Slide 6

Networking in Java — 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.
- Dynamic code loading possible too.

Slide 7

Networking in Java — RMI, Quick How-To

- Define a class for remote objects:
 - Define interface that extends `Remote`
 - Define class that implements that interface, extends a Java "remote object" class. Can also include other methods, only available locally.
 - Write code using classes — if using as remote object, reference interface; otherwise can reference class.
- Compile and execute:
 - Compile as usual, *plus* run `rmi` to generate "stubs" to be used in communicating with remote objects as remote objects.
 - Make classes network-accessible.
 - Start `rmiregistry`.
 - Run server and clients as usual.

Slide 8

More Networking Examples

- Sockets versus RMI: Java master/worker example from parallel programming class.
- Chat program.

Slide 9

Course Recap — What Did We Do?

- Java basics.
- Object-oriented programming — polymorphism, inheritance, etc. Not stressed much in class, but game is a good example of a non-trivial o-o design.
- Basic ADTs — stacks, queues, trees (sorted and heaps); different implementations (arrays versus dynamic data structures using references).
- Recursion review.
- Tour of the Java libraries — GUIs, graphics, I/O; a very little about threads and networking.
- A fairly large programming project involving using someone else's code.
- To get a sense of what you learned — compare what you knew in August to what you know now.

Slide 10

Minute Essay

- How did the course compare to your expectations/goals? Did you learn what you hoped to learn?