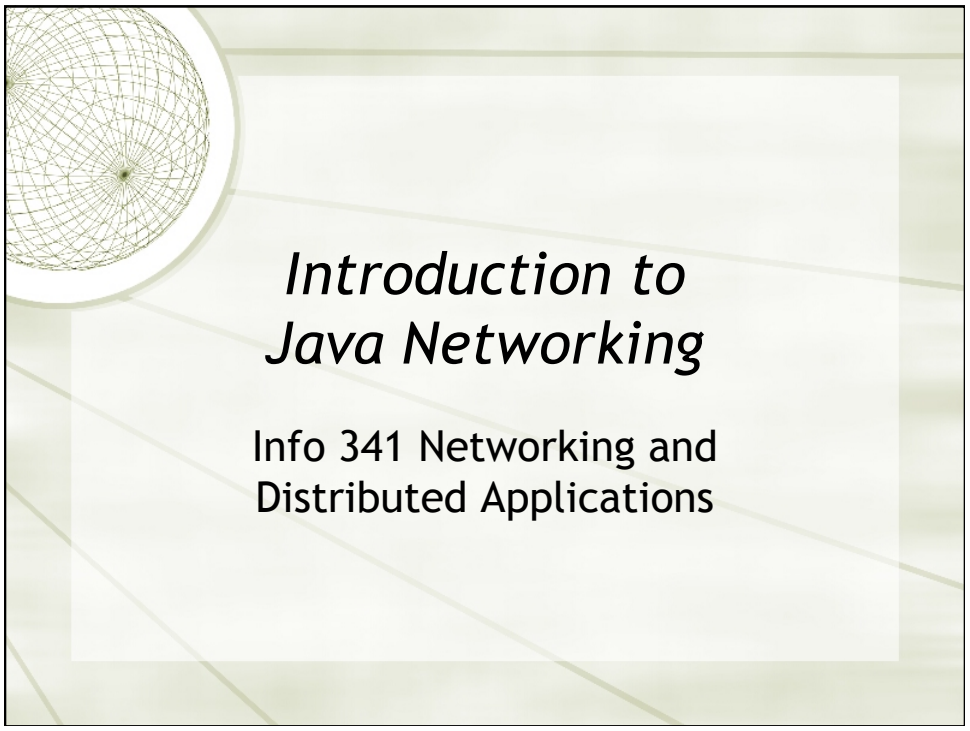
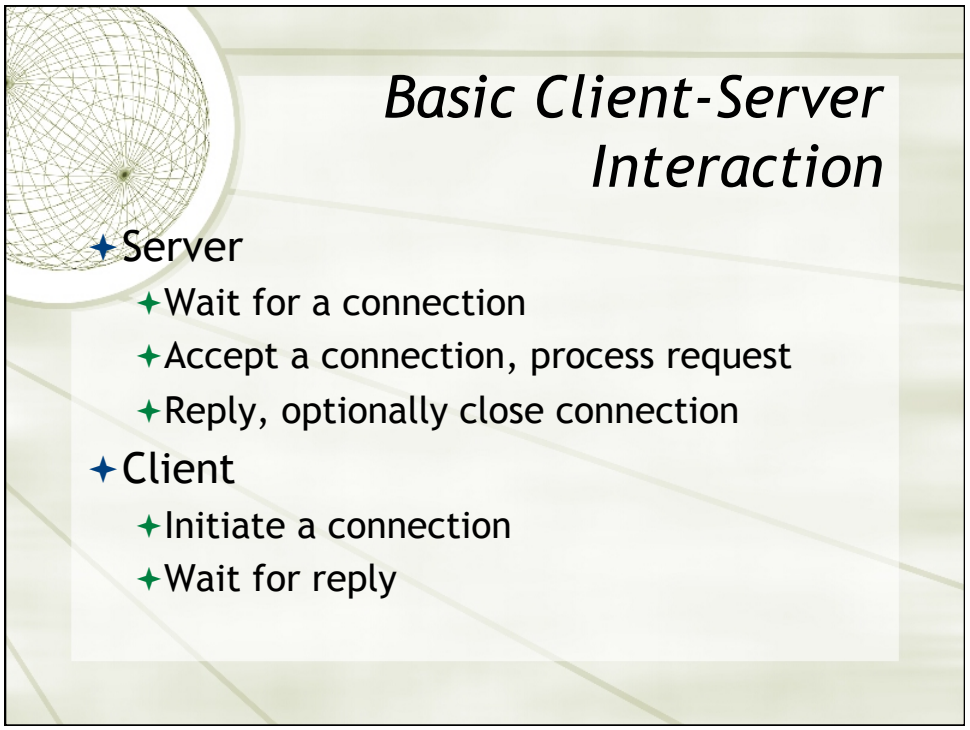

Introduction to
Java Networking
Info 341 Networking and
Distributed Applications


Basic Client-Server
Interaction
Server
Wait for a connection
Accept a connection, process request
Reply, optionally close connection
Client
Initiate a connection
Wait for reply

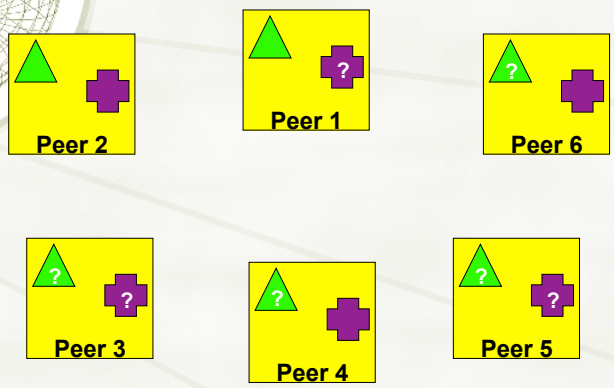
Client-Server

- Most of the Internet is client-server
  - Web
  - Email
  - File services
- An alternative is peer-to-peer

Peer-to-Peer Model

- In the basic peer-to-peer model every machine is both a client and a server
- Each peer can both produce and consume a service
- But this is not without problems …

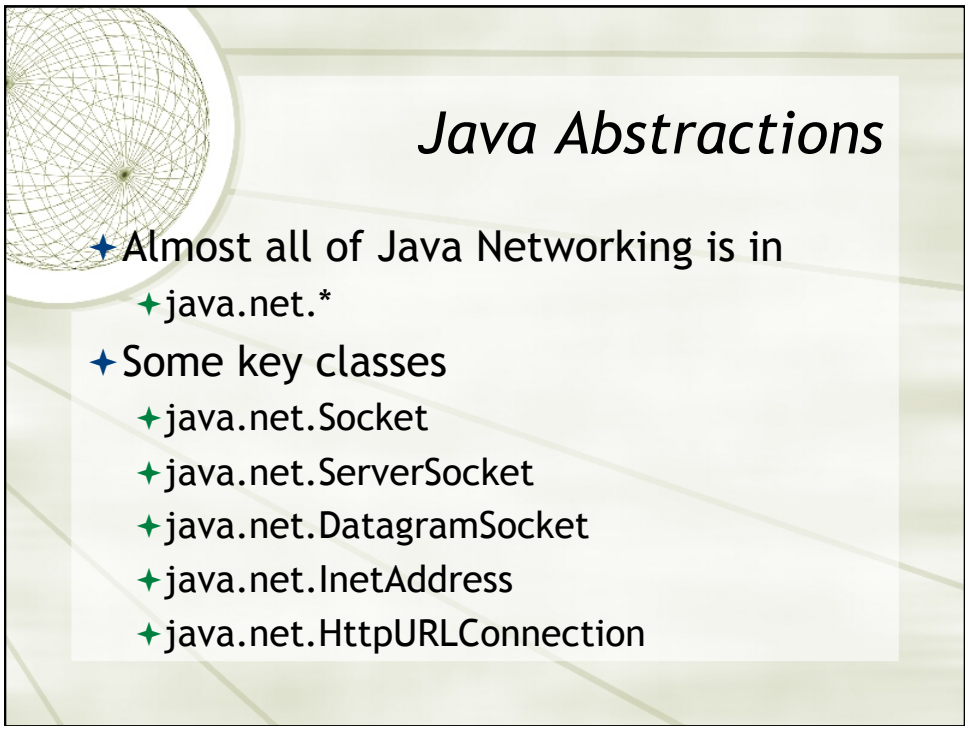
Peer-to-Peer Services

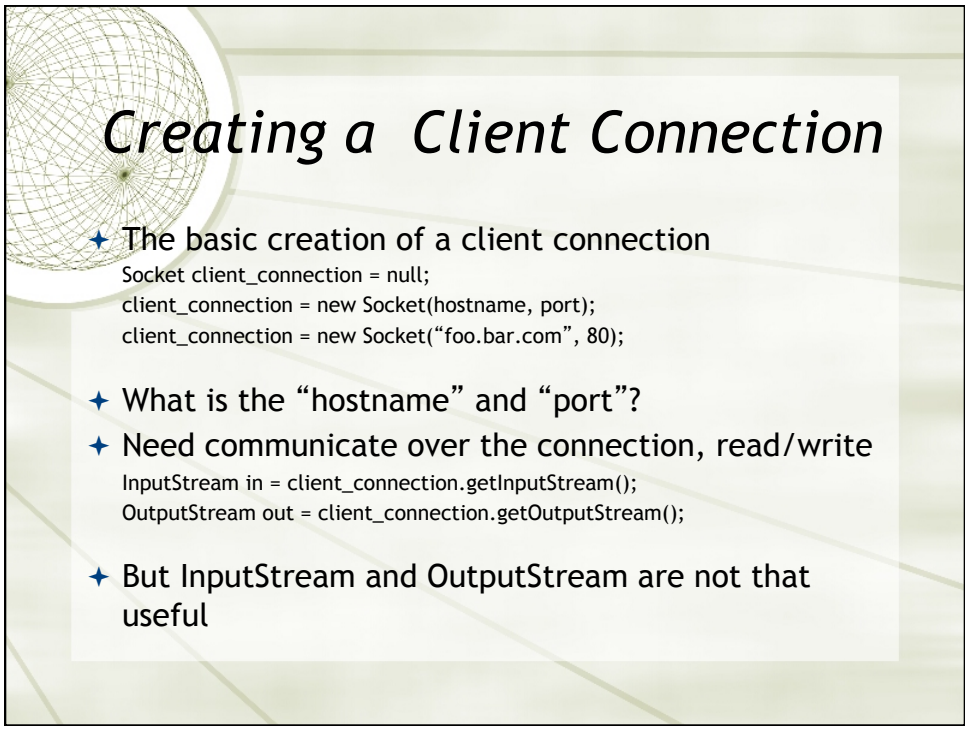


- How do peers find the services?
- How do we make sure that one peer is not overwhelmed?

Connections

- Socket
  - A common term for the network connection
  - Used by clients and servers
  - Sockets are generally "stream" connections
    - What kind of connection does "stream"
    - What is the alternative?
  - There are no "plugs" (sorry)


Java Abstractions
Almost all of Java Networking is in
java.net.*
Some key classes
java.net.Socket
java.net.ServerSocket
java.net.DatagramSocket
java.net.InetAddress
java.net.HttpURLConnection


Creating a Client Connection
The basic creation of a client connection
Socket client_connection = null;
client_connection = new Socket(hostname, port);
client_connection = new Socket("foo.bar.com", 80);
What is the "hostname" and "port"?
Need communicate over the connection, read/write
InputStream in = client_connection.getInputStream();
OutputStream out = client_connection.getOutputStream();
But InputStream and OutputStream are not that useful

Client Input/Output Streams

- Reading and writing on the socket

```
BufferedReader input_stream = null;
input_stream = new BufferedReader(
                    new InputStreamReader(
                    client_connection.getInputStream()));

PrintStream output_stream = null;
output_stream = new PrintStream(client_connection.getOutputStream());
```

- Almost any form of reader/writer can be created
- Reading from input_stream will "block"
  - What does that mean?

Handling Blocking I/O

- Need a separate Thread
  - What is a "thread"?

```
public class Connection extends Thread {
        …
    public void run() {
        String mesg;
        try {
            while(true) {
                mesg = input_stream.readLine();
                << call some other object.method passing "mesg" >>
            }
        } catch( IOException e ) {
            System.err.println("Exception: "+e.toString());
        }
    }
}
```

General Connection

- Connection needs a few methods to work …

```
public class Connection extends Thread {
        …
        public void run() {
            …
        }
        public void send(String mesg) {
            …
        }
        public void close() {
            …
        }
}
```

Server Sockets

- What is the general model of the server?
  - Wait for incoming connection
- Create a special kind of socket

```
ServerSocket server_socket = null;
server_socket = new ServerSocket(port);
```

  - What is “port” here?
- Accept incoming connections (socket requests)

```
Socket client_socket = null;
client_socket = server_socket.accept();
```

  - accept() blocks – what does that mean?

Server Connections

- Server objects need to be a Thread
  - Create a connection and hand it off

```
public class Server extends Thread {
   …
   public void run() {
       try {
           while(true) {
               Socket client_sock = server_sock.accept();
               Connection client_connection = new Connection(client_sock);
               << hand off the client_connection to another object >>
           }
        } catch( IOException e ) {
            System.out.Println("Exception: ”+e.toString());
        }
    }
}
```

Tid bits to remember

- Threads are covered in
  - java.lang.Thread
  - Must always have a public void run() method
  - Must call <object>.start() to begin the execution of the thread
- Socket Objects
  - Some cool methods, getInetAddress(), getLocalAddress()
- Protocol Ports
  - Port numbers 1-1024 are reserved for OS related services
  - Generally, use numbers above 20,000 (can go as high as 65,000)
  - The server must use the same port number as the client
  - Biggest issue – firewalls prevent some incoming/outgoing connections – You will need to approve/allow the connections when you run your code