CS 580 Client-Server Programming
Fall Semester, 2002
Doc 10 Server Intro
Contents

References

Java Network Programming, Harold

VisualWorks Internet Client Developer's Guide

java.net.ServerSocket & Socket. See
http://java.sun.com/j2se/1.4.1/docs/api/

Reading Assignment

Java

Java Network Programming, Harold

Basic Network Concepts, Chapter 2
If you are not familiar with basic networking

Java I/O Chapter 4
If you are weak on Streams, Readers and Writers

Sockets for Servers Chapter 11

Smalltalk

Internet Client Developer's Guide, Socket Programming Chapter 2.

Internet Client Developer's Guide is doc/NetClientDevGuide.pdf in the VisualWorks 7 installation

What is a Server?

Server

Any program that waits for incoming communication requests from a client

Extracts requested information from data and return to client

Basic algorithm:

```
while (true) {
  Wait for an incoming request;
  Perform whatever actions are requested;
}
```

Example - Echo Server

```
| server |
server := SocketAccessor newTCPserverAtPort: 9009.
server listenFor: 5.

 [ | acceptedSocket |
   "wait for a new connection"
   acceptedSocket := server accept.

   "fork off processing of the new stream socket"
   [ | stream char |
      stream := acceptedSocket readAppendStream.
      stream lineEndTransparent.
      [ (char := stream next) isNil ] whileFalse: [
         stream nextPut: char; commit ].
      stream close.
   ] forkAt: Processor userSchedulingPriority -1.
] repeat.
```

Some Basic Server Issues

- How to wait for an incoming request?
- How to know when there is a request?
- What happens when there are multiple requests?
- How do clients know how to contact server?
- How to parse client request?
- How do we know when the server has the entire request?

Sockets
Streams verses Buffers

Both Java & Smalltalk provide access to socket data via

- Streams
- Buffers

Stream access is easier

Buffer access can be faster

We will cover Stream access first

Java TCP Sockets
Main Classes

ServerSocket
- Used by servers to listen for clients

Socket
- Used by clients to talk to servers
- Used by servers to talk to clients

ServerSocket basic methods

```
public ServerSocket(int port) //port = 0 gives random port
public ServerSocket(int port, int backlog)
public ServerSocket(int port, int backlog, InetAddress bindAddress)

public Socket accept() throws IOException
public void close() throws IOException
public int getLocalPort()
```

Socket basic methods

```
public InputStream getInputStream() throws IOException
public OutputStream getOutputStream() throws IOException
```

A Simple Date Server

```
import java.net.Socket;
import java.net.ServerSocket;
import java.io.*;
import java.util.Date;

class SimpleDateServer {

   public static void main(String[] args) throws IOException {
      ServerSocket acceptor = new ServerSocket(0);
      System.out.println("On port " + acceptor.getLocalPort());

      while (true) {
         Socket client = acceptor.accept();
         processRequest(
            client.getInputStream(),
            client.getOutputStream());
         client.close();
      }
   }
```

(Why "new ServerSocket(0)"?)

Processing Client Request

```
  static void processRequest(InputStream in,OutputStream out)
    throws IOException {

    BufferedReader parsedInput =
         new BufferedReader(new InputStreamReader(in));

    // the "true" is to get autoflushing:
    PrintWriter parsedOutput = new PrintWriter(out,true);

    String inputLine = parsedInput.readLine();

    if (inputLine.startsWith("date")) {
      Date now = new Date();
      parsedOutput.println(now.toString());
    }
  }
}
```

Note: This server is just a first example. It needs a lot of work. We will be working on improving it in later lectures.

Running the Server

Sample run of SimpleDateServer.
(I typed everything appearing in bold font here.)

rohan 16-> **java SimpleDateServer &**
[1] 16269
On port 62047

rohan 17-> **telnet rohan 62047**
Trying 130.191.3.100...
Connected to rohan.sdsu.edu.
Escape character is '^]'.
date today
Mon Sep 04 13:37:30 PDT 2000
Connection closed by foreign host.

rohan 18-> **telnet rohan 62047**
Trying 130.191.3.100...
Connected to rohan.sdsu.edu.
Escape character is '^]'.
time
Connection closed by foreign host.

In this class, shut things down:

rohan 19-> fg
java SimpleDateServer
^C

Warning About telnet Usage

Using telnet to interact with a server is

- Useful as development tool
- Useful in debugging
- Not for end-users

Don’t design server assuming it interacts directly with a human!

Smalltalk TCP Sockets
Main Classes

IPSocketAddress
  Represents an IP address for a machine

SocketAccessor
  Socket connection for server & client

SocketAccessor creation methods

newTCP "selects random port"
newTCPserverAtPort: portNumber

SocketAccessor instance methods

accept
  Wait for a client connection and return it

acceptNonBlock
  return any waiting client connection,
  return nil if no waiting client connections

readAppendStream
readStream
writeStream
  Return a stream of the given type on the conection

A Simple Date Server

```
Smalltalk defineClass: #SimpleDateServer
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'serverSocket '
   classInstanceVariableNames: ''
   imports: ''
   category: 'SimpleServer'

SimpleDateServer class methodsFor: 'instance creation'

port: anInteger
   ^super new setPort: anInteger
```

SimpleDateServer instance methods

```
setPort: anInteger
   serverSocket := SocketAccessor newTCPserverAtPort: anInteger.
   serverSocket
      listenFor: 4;
      soReuseaddr: true

run
   | childSocket clientConnection clientIOStream |
   [childSocket := serverSocket accept.
   clientIOStream := childSocket readAppendStream.
   clientIOStream lineEndTransparent.
   self processRequestOn: clientIOStream.] repeat

processRequestOn: anReadAppendStream
   | clientRequest |
   clientRequest := anReadAppendStream through: Character cr.
   (clientRequest startsWith: 'date')
      ifTrue: [anReadAppendStream nextPutAll:
                  Time dateAndTimeNow printString].
   anReadAppendStream close
```

Running the Server

```
server := SimpleDateServer port: 5556.
serverProcess := [server run] fork
```

Simple Server Issues

- How do we test for our server?

- Request processing blocks any other connections.

Using our SimpleDateServer

> Client A builds connection to server,
> Client A goes to lunch
> Client B builds connection to server and ... :-(

Solution:
Multiple connections need to be accepted concurrently.

Backlog

TCP accepts connections before the server is ready

TCP keeps a backlog queue of connections server has not accepted

Java ServerSocket constructor
- Allows you to request a maximum size of the backlog queue
- The OS will not exceed its maximum TCP backlog queue size
- The OS silently reduces your request to it maximum

SocketAccessor>>listenFor: aNumber
- Requests the backlog queue in Smalltalk

There is no reasonable way to find out:
- OS's maximum TCP backlog queue size

Backlog Experiment

Start the SimpleDateServer

Connect to the server using telnet

While the server is waiting for you to type something

Connect to the server with a second telnet session

In the second session type “date” and return

What happens?

Now type “date” and return in the first session

What happens?

Multi-homed Machines

Some machines have two or more physical network interface

- Rohan has two ethernet cards
- Some laptops have ethernet & 802.11a

Each network interface has its own IP address

public ServerSocket(int port)

Listens to the give port on all IP addresses for the machine

public ServerSocket(int port, int backlog, InetAddress bindAddress)

Listens to the give port only on the one IP address

Reusing a Port

Closing TCP connections can remain for several minutes

TCP may block use of the port until the connection is gone

This can be annoying in development

SocketAccessor>> soReuseaddr: true

ServerSocket method setReuseAddress(boolean on)

Allows the port to be resued

End of Line

Platform	End of Line Convention
Unix	Line Feed (LF)
Macintosh	Carriage return (CR)
Windows	CR-LF

LF is ASCII character 10

CR is ASCII character 13

A server should not make assumptions about a client's platform

A client should not make assumptions about a server's platform

Client-server protocol should specify which characters are used

End of Line & Smalltalk

Java and Smalltalk programs run on all major platforms

Smalltalk assumes files use platform's end of line convention

Smalltalk input streams convert platforms end of line to CR

Smalltalk output streams convert CR to platform's end of line

This makes writing cross platform programs easier

Don't want this to happen socket streams

BufferedExternalStream>> lineEndTransparent

Turns off converstion between CR & end of line

How does Java handle this?

End of File

On a stream connected to a socket

End of file indicates that the connection has been closed!

Don’t use end of file to determine when other end is done talking!

End of Message

How do we know when we are at the end of a message?

```
BufferedReader parsedInput =
      new BufferedReader(new InputStreamReader(in));

char[] message = new char[500];
int sizeRead = parsedInput.read(message, 0, 500);
```

If

- client's message is less than 500 characters and
- Above read returns

We still may not have the entire message!

Why?

End of Message

A good client-server protocol specifies

- How to determine the end of a message

Main methods used:

- Include the length of the message
- Specify end of message character sequence

Buffers

Java & Smalltalk streams are buffered

TCP buffers output before sending

A server cannot read bytes left in a client's buffer

ExternalStream>>commit

Sends data in stream to OS

PrintWriter flush();