CS 683 Emerging Technologies: Embracing Change
Spring Semester, 2001
Doc 25 Squeak Client-Server
Contents

Squeak Client-Server 2

References

Squeak source code

Comanche 4.7

Copyright ©, All rights reserved. 2001 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.

Squeak Client-Server

When trying to develop client-server applications like Gnutella in Squeak one is led to the Socket class. In the examples in that class one finds sample client code like:

```
Socket class>>remoteTestClientTCP
    | socket bytesToSend sendBuf receiveBuf done bytesSent bytesReceived packetsSent packetsReceived
t serverName |
    Transcript show: 'starting client/server TCP test'; cr.
    Transcript show: 'initializing network ... '.
    Socket initializeNetworkIfFail: [^Transcript show:'failed'].
    socket := Socket newTCP.
    serverName := FillInTheBlank
        request: 'What is your remote Test Server?'
        initialAnswer: ''.
    socket connectTo: (NetNameResolver addressFromString: serverName) port: 54321.
    socket waitForConnectionUntil: Socket standardDeadline.
    bytesToSend := 1000000.
    sendBuf := String new: 4000 withAll: $x.
    receiveBuf := String new: 50000.
    done := false.
    bytesSent := bytesReceived := packetsSent := packetsReceived := 0.
    t := Time millisecondsToRun: [
        [done] whileFalse:
            [(socket sendDone and: [bytesSent < bytesToSend]) ifTrue:
                [packetsSent := packetsSent + 1.
                bytesSent := bytesSent + (socket sendData: sendBuf)].
            socket dataAvailable ifTrue:
                [packetsReceived := packetsReceived + 1.
                bytesReceived := bytesReceived + (socket receiveDataInto: receiveBuf)].
            done := (bytesSent >= bytesToSend)].
        [bytesReceived < bytesToSend] whileTrue:
            [socket dataAvailable ifTrue:
                [packetsReceived := packetsReceived + 1.
                bytesReceived := bytesReceived + (socket receiveDataInto: receiveBuf)]]].
    socket closeAndDestroy.
    Transcript show: 'remoteClient TCP test done; time = ', t printString; cr.
    Transcript show: packetsSent printString, ' packets, ',
                    bytesSent printString, ' bytes sent (',
                    (bytesSent * 1000 // t) printString, ' bytes/sec)'; cr.
    Transcript show: packetsReceived printString, ' packets, ',
                    bytesReceived printString, ' bytes received (',
                    (bytesReceived * 1000 // t) printString, ' bytes/sec)'; cr.
```

Problems

The above code contains some low-level work with buffers. When developing client-server applications such work has to be done. However, it is a good idea to do this low-level work once and encapsulate it in a class.

Another issue is the disposing of sockets when one is done with them. If they are not disposed of they can interfere with the next socket on that port. This can make debugging very difficult.

SocketStream

SocketStream here is a modified version of the class in Comanche 4.7. The modifications remove dependencies on the http protocol. SocketStream provides stream access to sockets so one does not have to deal with buffers.

You also need BufferStream, which is also modified from Comanche 4.7.

Some Important Methods

binary
: Set the SocketStream to read and write binary data from the socket. Data is read/written as bytes. ByteArray is used to handle collection of bytes

text
: Set the SocketStream to read and write ascii data from the socket. Data is read/written as characters. String is used to handle collection of characters

autoFlushOff, autoFlushOn
: Turn autFlush on/off

Writing data to other end of the connection

flush
: Write all buffered data to socket

cr, lf, crlf
: Write a cr (ascii 13), lf (ascii 10), or both to the socket. Data is sent to the other end of the socket connection. If autoFlush is on data is sent immediately, if autoFlush is off data is buffered until flush is called.

nextPut: charOrByte
nextPutAll: stringOrByteArray
: Write the characters/bytes to the socket. If autoFlush is on data is sent immediately, if autoFlush is off data is buffered until flush is called.

Reading data sent to us

These methods read from the data sent from the other end of the connection

next
Return the next char/byte from the socket.

next: integer
Return the next integer characters/bytes from the socket.

nextChunk
Return all data from other end that is currently available.

putDataBack: stringOrByteArray
Put the data back. It will be in front of the read queue

upTo: charOrByte
Return all data up to charOrByte. Skip over charOrByte.

upToAll: stringOrByteArray
Return all data up to stringOrByteArray. Skip over stringOrByteArray.

upToEnd
Read all data sent until other end closes connection

Client Example

The following example reads a web page and displays it on the Transcript. Not a complex client but it shows how to use SocketStream.

```
| socket clientStream crlfcrlf |
crlfcrlf := String crlf , String crlf.
Socket initializeNetworkIfFail: [^Transcript show:'failed'].
[socket := Socket newTCP.
socket
   connectTo:
      (NetNameResolver addressForName: 'www.eli.sdsu.edu')
   port: 80.
socket waitForConnectionUntil: Socket standardDeadline.
clientStream := SocketStream on: socket.
clientStream
   autoFlushOff;
   nextPutAll: 'GET / HTTP/1.0';
   crlf;
   crlf;
   flush.
Transcript
   show: 'headers';
   cr;
   show: (clientStream upToAll: crlfcrlf);
   cr;
   show: (clientStream upToEnd);
   cr.
clientStream close.]
ensure:
   [socket closeAndDestroy]
```

Server Example

The following is an example of a server using SocketStream. It implements an echo server for telnet. Start the server, then connect to the server (running on port 54320) with a telnet client. Each line you type is echoed back to you. Closing the connection allows the server to service a second connection. There is more work needed here, but enough for now.

```
| server socketStream socket |
Socket initializeNetworkIfFail: [^Transcript show:'failed'].
[server := Socket newTCP.
server listenOn: 54320 backlogSize: 4.
2 timesRepeat:
  [
  socket := server waitForAcceptUntil: (Socket deadlineSecs: 60).
  socketStream := SocketStream on: socket.
  [socket notNil and: [socketStream isConnected]] whileTrue:
    [data := socketStream upToAll: String crlf.
    Transcript
      show: data;
      cr.
    socketStream
      autoFlushOff;
      nextPutAll: data;
      crlf;
      flush].
  socket notNil
    ifTrue:[socket closeAndDestroy]]]
ensure:
  [server closeAndDestroy].
```