CS 580 Client-Server Programming
Spring Semester, 2005
Doc 7 Socket Options

Socket Options 2
Timeouts 3
Buffer Size 4
Nagle's Algorithm 10
Linger on close 12
Keep Alive 13
Urgent (Out of Band) Data 14

References

Java On-line API http://java.sun.com/j2se/1.4.1/docs/api/

Unix Network Programming, Stevens, 1990, Berkeley Sockets chapter 6.

TCP/IP Illustrated Vol 1, Stevens, 1994, chapter 20.

Internetworking with TCP/IP, BSD Socket Version Vol. 3, Comer, Stevens, Prentice-Hall, 1993

Copyright ©, All rights reserved. 2005 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.

Socket Options

- Timeouts
- Buffer Size
- Multi-Homing
- No Delay for small data
- Linger on close
- Keep-Alive
- Urgent-Data

Timeouts

Socket will time out after specified time of inactivity

Java JDK 1.4 and later

Both Socket and ServerSocket class support:

```
void setSoTimeout(int timeoutInMilliseconds) throws SocketException
void getSoTimeout() throws SocketException
```

Must be sent before performing a read

Read throws SocketTimeoutException when socket times out

Not normally used on ServerSockets

Buffer Size

Each TCP socket has

- Receive buffer
- Send Buffer

Buffers are in the TCP stack space (not the VM)

Buffer size should:

- Be at least 16KB on Ethernet
- Applications that send lots of data use 48KB or 64KB

TCP does not allow the sender to overflow the receiver's buffer

So the receiver's receive buffer as large as the sender's send buffer

Buffers larger than 64KB require special set up

Default Buffer Size

Depends on platform

Has changed over time

OS	Receive buffer Size
Solaris	32KB
Mac OS 10	32KB+ (33304 bytes)

Setting the Buffer Size – Java

void setReceiveBufferSize(int size) throws SocketException
int getReceiveBufferSize() throws SocketException

void setSendBufferSize(int size) throws SocketException
int getSendBufferSize() throws SocketException

A Socket object has both a send & receive buffer

A ServerSocket only has a receive buffer

Java Example

In this example the default buffer size will be fine

Setting the buffer size just to show how to do it

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

public class ServerWithTimeout extends Thread {
   static final int CLIENT_TIMEOUT = 3 * 1000; // in milliseconds
   static final int BUFFER_SIZE = 16 * 1024;
   ServerSocket acceptor;

   public static void main(String[] args) throws IOException {
      int port = Integer.parseInt( args[1]);

      ServerWithTimeout server = new ServerWithTimeout( port );
      server.start();
   }

   public ServerWithTimeout(int port ) throws IOException {
      acceptor = new ServerSocket(port);
      acceptor.setReceiveBufferSize( BUFFER_SIZE );
   }
```

Java Example Continued

```
public void run()
   {
   while (true)
      {
      try
         {
         Socket client = acceptor.accept();
         processRequest( client );
         }
      catch (IOException acceptError)
         {
         // for a later lecture
         }
      }
   }

void processRequest( Socket  client) throws IOException
   {
   try
      {
      client.setReceiveBufferSize( BUFFER_SIZE);
      client.setSoTimeout( CLIENT_TIMEOUT);
      processRequest(
         client.getInputStream(),
         client.getOutputStream());
      }
   finally
      {
      client.close();
      }
   }
```

Java Example Continued

```
void processRequest(InputStream in,OutputStream out)
   throws IOException
   {
   BufferedReader parsedInput = null;
   PrintWriter parsedOutput = null;
   try
      {
      parsedInput = new BufferedReader(new InputStreamReader(in));
      parsedOutput = new PrintWriter(out,true);

      String inputLine = parsedInput.readLine();

      if (inputLine.startsWith("date"))
         {
         Date now = new Date();
         parsedOutput.println(now.toString());
         }
      }
   catch (SocketTimeoutException clientTooSlow)
      {
      parsedOutput.println("Connection timed out");
      }
   }
}
```

Nagle's Algorithm

Delays transmission of new TCP packets while any data remains unacknowledged

Allows TCP to merge data into larger packets before sending

Introduced to avoid lots of small packets across a WAN

Delay is on be default

Java

```
class Socket
   {
   void setTcpNoDelay(Boolean noDelay) throws SocketException
   void getTcpNoDelay() throws SocketException
   }
```

Set noDelay to true to turn delay off

Linger on close

Determines what happens when a socket is closed

How long does the socket remain after close to

- Acknowledge packets
- Retransmit lost packets

Default is to

- Allow the application to continue
- TCP handles sending unsent data & rejecting new requests

Keep Alive

Send packet on inactive connection to prevent timeouts

At least 2 hour delay between sending keep alive packets

Long delay limits it usefulness

Urgent (Out of Band) Data

Urgent data can be read out of order

- Read before data that was sent before it

Java 1.4

- Supports sending of urgent data
- Does not promote urgent data in the input stream