An Empirical Evaluation of the Performance of Mobile Network Connections

Markus Bylund

Swedish Institute of Computer Science
Box 12 63, 164 29 Kista, Sweden
markus.bylund@sics.se

Abstract. We present the results of an empirical evaluation of the performance of some network connections available for mobile terminals. General Packet Radio Service (GPRS) is compared with Bluetooth and a USB wired connection from a Sony Ericsson P800 smart phone.

1 Introduction

Many, if not most, discussions about mobility have so far been about network issues, ranging from how to provide support for mobility in network protocols (for example Mobile IP [1]) to discussions about different solutions for wireless networking. The main reason for this is that different technical constraints influence what can be done in mobile settings. Bandwidth for example, can easily vary with a factor of 1000 depending on network connection (from GPRS with less than 100 kb/s to wired networks with 100 Mb/s or more). Variations in latency[1] are almost as dramatic – a factor of 100 can easily be found (from several seconds with GPRS down to milliseconds with wired networks).

In this paper, we present the results from an empirical evaluation of different network connections for mobile terminals. In contrast to other evaluations of (generally mobile) network connections [2, 3], ours is made from an application level in real user settings. This makes the evaluation neither exhaustive, nor unbiased; there are simply too many unknown variables. A more thorough evaluation of the performance of GPRS has been made by Chakravorty et. al [4].

2 Evaluation Setup

A simple Java application running on a Sony Ericsson P800 smart phone made a series of HTTP requests with the phone connected to the Internet via GPRS as well as Bluetooth and USB cable via a laptop computer (which was connected to a high speed wired LAN). The response time between the network connections turned out to be equal for USB and Bluetooth but many times as high for GPRS (see Table 1). A single HTTP HEAD request, which included less than 1 kB of content, required several seconds to complete when operating via GPRS. When the request was changed to HTTP GET, which in addition downloaded a 55 kB large entity, response times were raised to 15 s on average. A complete listing of the test data, including HTTP headers used and a compilation of all data is given in Appendix I. A complete listing of the Java source code used to run the test is given in Appendix II.

[1] Throughout this discussion, we define latency as the roundtrip time of a small (<512 bytes) message.

2003-08-05 ISSN 1100-3154
SICS Technical Report T2003:06 ISRN:SICS-T--2003/06-SE

		HEAD		GET		
		Latency	Std. Dev.	Latency	Std. Dev.	Bandwidth
Sony Ericsson P800	USB	220 ms	7 ms	2,200 ms	290 ms	200 kb/s
	BT	290 ms	14 ms	2,400 ms	710 ms	180 kb/s
	GPRS	2,600 ms	250 ms	15,000 ms	3,700 ms	29 kb/s
iPaq 3870	WLAN	140 ms	4 ms	260 ms	4 ms	2 Mb/s
Laptop PIII, 600MHz, Win 2k	LAN	<10 ms	5 ms	10 ms	5 ms	32 Mb/s
	BT/GPRS	1,600 ms	150 ms	14,000 ms	2,100 ms	32 kb/s

Table 1. A comparison between different means for accessing Internet based services from different mobile devices. The HEAD request type gives a practical estimate of the minimal latency of the connection while the GET request type also takes the factor of bandwidth into account. Each number is an average of 25 repetitions of each request. The bandwidth is estimated based on the GET response time.

For comparison, a Compaq iPaq 3870 PDA with a WLAN card was also tested. The PDA displayed a latency that was only 60% of the phone with its fastest connection (USB) and a ten times as high bandwidth. The PDA differ from the phone in a number of ways (processor, OS, and Java VM), but the differences are not big enough to explain the huge difference in network performance that the test reveals. It is therefore likely that the phone would display similar performance (as the iPaq) if it could be equipped with a WLAN network connection.

The application was also tested on a laptop with two different network connections in order to find out the maximum impact of other factors. The laptop test with the LAN connection concluded that the impact of delays caused by the Web server that was targeted throughout all tests was less than 10 ms, which makes this delay negligible. Tests on the laptop with the BT/GPRS (Bluetooth connection to P800 phone, GPRS connection to the Internet) connection revealed that the overhead due to (unknown) issues with the P800 phone was at most 1 s per request (independently of request type). The latter test turned out to be irrelevant since the P800 phone was provably capable of executing a request in less than 300 ms, which is far less than the 1 s difference between phone and laptop.

3 Discussion

For quite a few classes of applications, the performance of GPRS is too low – for example, highly interactive applications such as games, where the latency is the most limiting factor. This is however also the case for as simple applications as web browsers. A web page is typically made out of several dozens of data entities which all need to be fetched with HTTP requests. For example, it typically requires several minutes to load the first page of the Swedish newspaper Dagens Nyheter (a total of about 300 kB and several dozens of entities to download) to a Sony Ericsson P800 smart phone over a GPRS connection. Loading the same page on a desktop computer with a high speed wired connection requires less than a second. By introducing a proxy close to the wired-wireless border, the performance of GPRS can improved [3], but the gap to 4G technologies and wired networks is still huge.

The small difference between the USB and Bluetooth connections came as a surprise. With a theoretical maximum bandwidth of 2 Mb/s for the USB connection, and 723 kb/s for Bluetooth, we expected the USB connection to be two to three times as fast as the Bluetooth connection, at least in the case of the GET request. One possible reason for this not being the case is that the Sony Ericsson Phone Connection PPC Suite, which provides network pass trough from laptop to phone, is implemented in a way that limits the performance of the USB connection.

It was also interesting to notice that the variation in response time for the slow GPRS connections was exceptionally high. These variations could be explained by variations in GPRS channel availability, but also TCP retransmissions due to the high latency of the connection [3].

References

[1] C. Perkins, "Mobile IP," in *IEEE Comm.*, vol. 35, 1997, pp. 84-99.

[2] S. Porcarelli, F. D. Giandomenico, and A. Bondavalli, "Analyzing Quality of Service of GPRS Network Systems from a User's Perspective," presented at The Seventh International Symposium on Computers and Communication (ISCC'02), 2002.

[3] R. Chakravorty and I. Pratt, "Performance Issues with General Packet Radio Service," *Journal of Communications and Networks*, vol. 4, pp. 266-281, 2002.

[4] R. Chakravorty, J. Cartwright, and I. Pratt, "Practical Experience with TCP over GPRS," presented at the IEEE Global Communications Conference (IEEE GLOBECOM 2002), Taipei, Taiwan, 2002.

Appendix I

HTTP Request Header

HEAD http://193.10.66.94/~bylund/images/Student97Brothers(480x480).jpg HTTP/1.1
User-Agent: Java3.0.2
Host: localhost
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP Reply Header (typical)

HTTP/1.1 200 OK
Date: Wed, 09 Apr 2003 09:45:42 GMT
Server: Apache/1.3.22 (Unix) (Red-Hat/Linux) mod_ssl/2.8.5 OpenSSL/0.9.6
DAV/1.0.2 PHP/4.1.2 mod_perl/1.24
Last-Modified: Tue, 03 Mar 1998 09:52:50 GMT
ETag: "15f8bc6-d411-34fbd2f2"
Accept-Ranges: bytes
Content-Length: 54289
Connection: close
Content-Type: image/jpeg

Network Types

GPRS: A GPRS network connection to Telia Mobile
BT: A serial Bluetooth connection to the Sony Ericsson Phone Connection PC Suite Internet passthrough
USB: A USB cable connection to the Sony Ericsson Phone Connection PC Suite Internet passthrough
BT/GPRS: Reference machine connected via Bluetooth and GPRS
LAN: Reference machine connected via LAN
WAN: Reference machine connected via WAN

Hardware Types

HW1: Sony Ericsson P800 with Symbian v. 7, virtual machine: Personal Java v. 1.1.1 (Symbian)
HW2: Intel PIII-M with Windows 2000, virtual machine: Java 1.4.1 (SUN)
HW3: Compaq iPaq 5370 with MS PocketPC 2002, virtual machine: Personal Java v.1.2 (Joede)

Compiled Results for Suite 1

Place: SICS, Isafjordsgatan 22, Kista, Sweden
Date: April 8, 2003

Network type	**USB**				**Bluetooth**			
Network name	SICS				SICS			
Hardware	HW-2				HW-2			
Time (Start/End)	16:24:44		16:26:02		16:29:32		16:31:06	
	Avg.	Std.dev.	Max	Min	Avg.	Std.dev.	Max	Min
HEAD	219 ms	7 ms	235 ms	203 ms	286 ms	14 ms	329 ms	265 ms
GET	2 151 ms	287 ms	2 718 ms	1 734 ms	2 386 ms	707 ms	3 860 ms	1 718 ms
Bandwidth	204 kbit/s	- - -	161 kbit/s	253 kbit/s	184 kbit/s	- - -	114 kbit/s	255 kbit/s

Network type	**GPRS**				**WLAN**			
Network name	TELIA-MOBILE-SE				SICS			
Hardware	HW-1				HW-3			
Time (Start/End)	15:53:44		16:01:25		16:13:09		16:13:22	
	Avg.	Std.dev.	Max	Min	Avg.	Std.dev.	Max	Min
HEAD	2 608 ms	255 ms	3 172 ms	1 922 ms	138 ms	4 ms	144 ms	128 ms
GET	15 258 ms	3 666 ms	26 407 ms	13 297 ms	263 ms	4 ms	272 ms	255 ms
Bandwidth	29 kbit/s	- - -	17 kbit/s	33 kbit/s	2 Mbit/s	- - -	2 Mbit/s	2 Mbit/s

Network type	**LAN**				**BT/GPRS**			
Network name	SICS				TELIA-MOBILE-SE			
Hardware	HW-2				HW-2			
Time (Start/End)	16:13:09		16:13:22		16:15:17		16:22:03	
	Avg.	Std.dev.	Max	Min	Avg.	Std.dev.	Max	Min
HEAD	6 ms	5 ms	11 ms	0 ms	1 654 ms	153 ms	2 304 ms	1 552 ms
GET	14 ms	5 ms	21 ms	10 ms	13 813 ms	2 092 ms	23 604 ms	12 839 ms
Bandwidth	32 Mbit/s	- - -	21 Mbit/s	44 Mbit/s	32 kbit/s	- - -	19 kbit/s	34 kbit/s

Compiled Results for Suite 2

Place: SICS, Lägerhyddsvägen 5, Uppsala, Sweden
Date: April 9, 2003

Network type	**USB**				**Bluetooth**			
Network name	SE-SUNET-193-10				SE-SUNET-193-10			
Hardware	HW-2				HW-2			
Time (Start/End)	09:52:46		09:54:13		10:08:02		10:09:09	
	Avg.	Std.dev.	Max	Min	Avg.	Std.dev.	Max	Min
HEAD	227 ms	8 ms	235 ms	218 ms	286 ms	10 ms	313 ms	265 ms
GET	2 179 ms	289 ms	2 750 ms	1 718 ms	1 736 ms	21 ms	1 766 ms	1 703 ms
Bandwidth	201 kbit/s	- - -	159 kbit/s	255 kbit/s	252 kbit/s	- - -	248 kbit/s	257 kbit/s

Network type	**GPRS**				**WLAN**			
Network name	TELIA-MOBILE-SE				SE-SUNET-193-10			
Hardware	HW-1				HW-3			
Time (Start/End)	10:22:41		10:47:52		09:49:13		09:49:28	
	Avg.	Std.dev.	Max	Min	Avg.	Std.dev.	Max	Min
HEAD	8 264 ms	9 703 ms	51 062 ms	2 015 ms	144 ms	3 ms	151 ms	140 ms
GET	16 806 ms	4 119 ms	30 844 ms	13 266 ms	284 ms	47 ms	489 ms	264 ms
Bandwidth	26 kbit/s	- - -	14 kbit/s	33 kbit/s	2 Mbit/s	- - -	1 Mbit/s	2 Mbit/s

Network type	**LAN**				**BT/GPRS**			
Network name	SE-SUNET-193-10				TELIA-MOBILE-SE			
Hardware	HW-2				HW-2			
Time (Start/End)	09:49:13		09:49:28		10:53:04		11:03:33	
	Avg.	Std.dev.	Max	Min	Avg.	Std.dev.	Max	Min
HEAD	12 ms	4 ms	20 ms	10 ms	3 141 ms	1 389 ms	5 668 ms	1 713 ms
GET	40 ms	5 ms	60 ms	30 ms	16 210 ms	4 333 ms	30 293 ms	13 399 ms
Bandwidth	11 Mbit/s	- - -	7 Mbit/s	15 Mbit/s	27 kbit/s	- - -	14 kbit/s	33 kbit/s

Raw Data for Suite 1

All numbers are in milliseconds.

USB		Bluetooth		GPRS		WLAN		LAN		BT/GPRS	
HEAD	GET	HEAD	GET	HEAD	GET	HEAD	GET	HEAD	GET	HEAD	GET
234	1 797	297	1 781	2 188	13 297	142	263	10	10	1 852	12 858
203	1 750	329	1 969	1 953	13 750	139	268	10	10	1 682	12 839
218	1 984	281	2 172	2 657	14 047	137	264	10	20	2 304	13 339
218	2 031	297	2 594	3 172	23 703	142	267	10	10	1 692	13 569
219	2 718	281	3 203	1 922	13 985	138	269	0	10	1 692	23 604
203	2 031	297	1 796	2 672	14 187	141	261	10	10	1 693	15 071
219	2 094	281	1 765	2 625	26 407	140	263	10	10	1 602	13 088
219	2 079	297	1 718	2 828	13 407	139	265	0	10	1 582	13 249
219	2 469	297	1 734	2 765	14 265	136	265	10	10	1 612	13 930
219	2 312	281	1 765	2 641	13 765	144	262	0	10	1 602	13 049
219	1 969	281	1 813	2 625	13 938	137	258	10	20	1 552	13 359
234	2 484	281	1 734	2 656	13 500	134	261	0	20	1 622	13 089
219	2 515	281	1 766	2 609	13 656	142	262	0	10	1 572	13 660
219	1 734	265	1 875	2 782	24 469	141	272	10	20	1 622	13 339
219	2 125	282	1 984	2 593	13 453	138	267	10	20	1 773	13 460
219	2 172	281	1 922	2 625	14 343	143	263	0	20	1 562	13 670
219	2 015	282	2 156	2 688	13 859	128	261	11	10	1 612	13 289
218	2 297	282	3 282	2 594	13 672	138	265	0	10	1 592	13 580
219	2 297	265	2 813	2 594	13 907	136	259	10	10	1 592	13 359
219	1 735	265	3 187	2 687	14 110	141	260	0	20	1 603	12 999
219	2 344	312	3 860	2 625	13 938	132	265	10	20	1 572	13 740
218	1 750	297	3 719	2 641	13 843	137	255	10	10	1 583	13 690
235	2 688	281	3 281	2 594	13 937	130	264	10	21	1 552	13 670
218	2 172	282	2 797	2 656	14 610	133	260	0	10	1 593	12 979
219	2 203	281	2 953	2 797	15 390	140	259	10	10	1 642	12 839

Raw Data for Suite 2

All numbers are in milliseconds.

USB		Bluetooth		GPRS		WLAN		WAN		BT/GPRS	
HEAD	GET	HEAD	GET	HEAD	GET	HEAD	GET	HEAD	GET	HEAD	GET
234	2 062	297	1 750	8 922	17 437	145	306	20	60	2 784	30 293
218	2 360	281	1 703	2 015	14 312	149	277	10	40	3 084	13 790
219	2 016	297	1 735	6 797	13 953	141	267	10	30	2 214	17 845
235	2 140	296	1 703	6 781	16 078	142	274	10	40	2 173	14 612
234	2 406	281	1 750	5 594	23 750	149	270	10	40	4 977	14 891
219	2 000	265	1 703	9 031	14 469	151	363	10	41	3 064	13 700
218	2 359	281	1 765	9 016	30 844	141	267	20	40	5 408	13 910
219	1 750	281	1 734	2 687	13 516	142	272	10	40	1 713	14 711
219	2 109	281	1 766	2 782	14 282	141	268	10	41	5 628	13 520
218	1 718	297	1 765	2 782	13 938	145	489	10	40	2 844	13 790
234	2 656	313	1 766	14 875	20 422	145	282	10	40	5 668	13 399
235	2 078	281	1 766	11 391	14 937	146	272	10	41	2 243	16 954
219	2 140	281	1 750	3 282	16 578	144	270	10	40	2 173	13 940
235	2 188	282	1 719	6 203	14 094	144	273	20	40	2 183	14 622
234	2 328	281	1 735	5 937	14 640	145	271	10	40	2 284	14 230
234	2 125	282	1 750	7 250	14 922	147	271	10	40	5 528	16 283
235	2 546	297	1 735	51 062	17 641	141	272	10	40	2 223	13 910
234	1 781	282	1 750	5 687	23 188	144	268	10	30	3 284	18 727
234	2 281	297	1 703	2 062	14 375	143	272	20	40	2 333	14 470
235	2 485	281	1 734	5 907	14 547	142	272	10	40	1 843	19 678
234	2 344	282	1 718	2 906	20 359	142	267	11	40	5 488	14 321
219	2 359	281	1 735	15 515	14 438	143	267	10	40	2 153	13 460
219	2 750	297	1 718	3 359	13 266	140	270	10	40	1 892	17 305
218	1 750	281	1 719	11 750	16 797	143	265	10	40	3 094	28 471
219	1 750	281	1 735	3 016	17 375	141	264	10	40	2 244	14 421

Appendix II

```
import java.awt.*;
import java.awt.event.*;
import java.awt.datatransfer.*;
import java.io.*;
import java.net.*;
import java.util.*;
import java.text.*;
import javax.swing.*;

public class NetTest {
  private NetTestWindow gw;

  public NetTest(NetTestWindow gw){
    this.gw = gw;
  }

  public void test(String target, boolean headOnly) {
    try {
      long average;
      URL url;
      String host, path, requestString, contentLengthHeaderString;
      byte[] request, contentLengthHeader, reply, entity;
      int contentLength, index, repeat;
      Date[] begin, end;
      Socket socket;
      InputStream is;
      OutputStream os;
      byte cs, lastChecksum;

      // initialize
      url = new URL(target);
      host = url.getHost();
      path = url.getFile();

      requestString = (headOnly ? "HEAD " : "GET ") + path + " HTTP/1.1\r\n" +
            "User-Agent: Java3.0.2\r\n" +
            "Host: localhost\r\n" +
            "Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2\r\n" +
            "Connection: keep-alive\r\n" +
            "\r\n";
      request = requestString.getBytes();

      contentLengthHeaderString = "Content-Length:";
      contentLengthHeader = contentLengthHeaderString.getBytes();

      contentLength = 0;
      index = 0;
      repeat = 25;
      lastChecksum = 0;
      average = 0;

      reply = new byte[1024];
      entity = new byte[65536];
      begin = new Date[repeat];
      end = new Date[repeat];

      // do the test 'repeat' times
      for (int i=0; i<repeat; i++) {
            // force a garbage collection before starting the timer
            System.gc();
```

```
            // start the timer
            begin[i] = new Date();

            // open the connection and send the request
            socket = new Socket(url.getHost(), 80);
            os = socket.getOutputStream();
            os.write(request);
            is = socket.getInputStream();

            // read headers
            index = 0;
            while (true) {
              int length = readLine(is, reply, index);
              if (equals(reply, index, contentLengthHeader)) contentLength =
                parseContentLength(reply, index); // parse content length
              index += length;
              reply[index++] = (byte) '\n';
              if (length==0) break;
            }

            // read entity (if any)
            if (!headOnly) {
              int nbBytesRead = 0;
              do {
                nbBytesRead += is.read(entity, nbBytesRead, contentLength-nbBytesRead);
              } while (nbBytesRead<contentLength);
            }

            // close connection
            os.close();
            is.close();

            // stop the timer
            end[i] = new Date();

            // check content
            cs = checksum(entity);
            if (i>0) {
              if (cs!=lastChecksum) {
                gw.error("Checksum inconsitency: " + cs + "!=" + lastChecksum);
                return;
              }
            }
            lastChecksum = cs;
    }

    // dump report
    average = (end[repeat-1].getTime()-begin[0].getTime())/repeat;
    gw.log("Made " + repeat + " HTTP " + (headOnly ? "HEAD " : "GET ") +
      " requests to " + target + "\n");
    gw.log("  date: " + begin[0] + " to " + end[24] + "\n");
    gw.log("  size: " + request.length + " bytes sent and " + index +
      ((headOnly) ? "" : (" + " + contentLength)) + " bytes received\n");
    if (!headOnly) gw.log("  checksum: " + checksum(entity) + "\n");
    gw.log("  average: " + average + " msec on average [");
    for (int i=0; i<(repeat-1); i++) gw.log((end[i].getTime()-begin[i].getTime()) + ", ");
    gw.log((end[repeat-1].getTime()-begin[repeat-1].getTime()) + "]\n\n");
    gw.log(new String(reply, 0, index) + "\n");
  } catch (Exception e) {
    e.printStackTrace();
    gw.error("Exception thrown: " + e.getMessage());
  }
}
```

```
  byte checksum(byte[] array) {
    byte b = 0;
    for(int i=1; i<array.length; i++) b ^= array[i];
    return b;
  }

  int readLine(InputStream is, byte[] line, int off)
    throws IOException {
    int i;
    for (i=off; i<1024; i++) {
      line[i] = (byte) is.read();
      if (line[i]=='\r') {
            line[i+1] = (byte) is.read();
            if (line[i+1]=='\n') break;
            else i++;
      }
    }
    return i-off;
  }

  boolean equals(byte[] a, int offset, byte[] a2) {
    int length = a2.length;
    for (int i=0; i<length; i++)
      if (a[i+offset]!=a2[i])
            return false;
    return true;
  }

  int parseContentLength(byte[] headers, int offset) {
    try {
      int begin, end;
      begin = offset+15; // get rid of "Content-Length:"
      while (headers[begin]<48 || headers[begin]>57) begin++; // skip WS before the actual number
      end = begin+1;
      while (headers[end]>=48 && headers[end]<=57) end++; // get the index of the end of the number
      return Integer.parseInt(new String(headers, begin, end-begin));
    } catch (Exception e) {
      e.printStackTrace();
      return 0;
    }
  }

  static public void main(String[] args) {
    new NetTestWindow();
  }
}

class NetTestWindow implements ActionListener {
  JButton testButton;
  TextField textField;
  JFrame frame;
  JCheckBox checkBox;
  JPanel panel, panel2, panel3;
  JTextArea textArea;
  JMenu fileMenu, editMenu;
  JMenuBar menuBar;
  JMenuItem quitMenuItem, cutMenuItem, copyMenuItem, pasteMenuItem, selectAllMenuItem;

  NetTestWindow() { //
    quitMenuItem = new JMenuItem("Quit");
    quitMenuItem.setActionCommand("quit");
    quitMenuItem.addActionListener(this);
    fileMenu = new JMenu("File");
```

```
  fileMenu.add(quitMenuItem);
  cutMenuItem = new JMenuItem("Cut");
  cutMenuItem.setActionCommand("cut");
  cutMenuItem.addActionListener(this);
  copyMenuItem = new JMenuItem("Copy");
  copyMenuItem.setActionCommand("copy");
  copyMenuItem.addActionListener(this);
  pasteMenuItem = new JMenuItem("Paste");
  pasteMenuItem.setActionCommand("paste");
  pasteMenuItem.addActionListener(this);
  selectAllMenuItem = new JMenuItem("Select All");
  selectAllMenuItem.setActionCommand("selectAll");
  selectAllMenuItem.addActionListener(this);
  editMenu = new JMenu("Edit");
  editMenu.add(cutMenuItem);
  editMenu.add(copyMenuItem);
  editMenu.add(pasteMenuItem);
  editMenu.add(selectAllMenuItem);
  menuBar = new JMenuBar();
  menuBar.add(fileMenu);
  menuBar.add(editMenu);
  textArea = new JTextArea();
  textArea.setEditable(false);
  textField = new TextField(20);
  textField.setText("http://193.10.66.94/~bylund/images/Student97Brothers(480x480).jpg");
  textField.setEditable(true);
  checkBox = new JCheckBox("HTTP HEAD", true);
  testButton = new JButton("Test");
  testButton.setActionCommand("test");
  testButton.addActionListener(this);
  panel = new JPanel(new FlowLayout());
  panel.add(checkBox);
  panel.add(testButton);
  panel2 = new JPanel(new BorderLayout());
  panel2.add("North", textField);
  panel2.add("South", panel);
  panel3 = new JPanel(new BorderLayout());
  panel3.add("Center", new JScrollPane(textArea));
  panel3.add("South", panel2);
  frame = new JFrame();
  frame.setTitle("NetTest");
  frame.setSize(208, 276);
  frame.setJMenuBar(menuBar);
  frame.getContentPane().setLayout(new BorderLayout());
  frame.getContentPane().add(panel3);
  frame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
  frame.setVisible(true);
}

public void actionPerformed(ActionEvent evt) {
  if (evt.getActionCommand().equals("test")) {
    new NetTest(this).test(textField.getText().trim(), checkBox.isSelected());
  } else if (evt.getActionCommand().equals("quit")) {
    System.exit(0);
  } else if (evt.getActionCommand().equals("cut")) {
    Clipboard cb = Toolkit.getDefaultToolkit().getSystemClipboard();
    String s = textArea.getSelectedText();
    StringSelection contents = new StringSelection(s);
    cb.setContents(contents, null);
    textArea.replaceSelection("");
  } else if (evt.getActionCommand().equals("copy")) {
    Clipboard cb = Toolkit.getDefaultToolkit().getSystemClipboard();
    String s = textArea.getSelectedText();
    StringSelection contents = new StringSelection(s);
```

```
      cb.setContents(contents, null);
    } else if (evt.getActionCommand().equals("paste")) {
      Clipboard cb = Toolkit.getDefaultToolkit().getSystemClipboard();
      Transferable content = cb.getContents(this);
      try {
              String s = (String) content.getTransferData(DataFlavor.stringFlavor);
              if (textArea.getSelectedText()==null) textArea.insert(s, textArea.getCaretPosition());
              else textArea.replaceSelection(s);
      } catch (Exception e) {
              System.err.println(e);
      }
    } else if (evt.getActionCommand().equals("selectAll")) {
      textArea.selectAll();
    }
  }

  public void error(final String msg) {
    SwingUtilities.invokeLater(new Runnable() {
              public void run() {
                textArea.append("### ERROR ###\n");
                textArea.append(msg + "\n");
              }
      });
  }

  public void log(final String msg) {
    SwingUtilities.invokeLater(new Runnable() {
              public void run() {
                textArea.append(msg);
              }
      });
  }
}
```