International Conference on Complex, Intelligent and Software Intensive Systems

A Java-based Wrapper for Wireless Communications

A. Genco, S. Sorce, C. Ferrarotto, R. Gallea, A. Gentile, S. Impastato, M. Morana
DINFO – Dipartimento di Ingegneria Informatica
Viale delle Scienze, edificio 6 - 90128 Palermo, Italy
{genco,sorce}@unipa.it; {gallea,morana}@csai.unipa.it

Abstract

The increasing number of new applications for mobile devices in pervasive environments, do not cope with changes in the wireless communications. Developers of such applications have to deal with problems arising from the available wireless connections in the given environment. A middleware is a solution that allows to overcome some of these problems. It provides to the applications a set of functions that facilitate their development. In this paper we present a Java-based communication wrapper, called SmartTraffic, which allows programmers to seamlessly use TCP or UDP protocols over Bluetooth or any IP-based wireless network. Developers can use SmartTraffic within their Java applications, thus focusing on the application goals, and leaving out details about how it should interact with the available wireless connection.

1. Introduction

The main goal of pervasive systems is to spread a real-life environment with a large variety of smart devices, in order to supply people with some kind of service, thus becoming an augmented environment. Pervasive systems improve environment capabilities by the use of a variety of interaction devices such as sensors, actuators, remote displays, wearable computers and so on. These devices are capable of making our natural reality an augmented reality where all things and beings can be enriched by virtual contents.

The exponential diffusion of such devices, third-generation wireless and Bluetooth communication devices, as well as location technologies, have led to a growing interest towards the development of pervasively-accessible and context-aware services. Context factors, such as who, when, where, are used to provide people with useful information in relation with the user profile and with the environment within the user is in.

Personal mobile devices, such as PDAs or Smartphones, are currently successfully exploited in the human-environment interaction, where they could be made suitable to operate as remote controllers, or personal I/O interfaces, for some application remotely running. There is a large variety of application fields where services can be pervasively accessed by mobile devices, such as multimodal context-aware information provision within university campuses [1], interactive user profile-based guides in cultural heritage sites [2], augmented reality objects assembly in mobility [3], healthcare systems [4], personal communications systems [5].

The interest towards the use of mobile devices in pervasive environments, led to the proliferation of specific programming platforms and toolkits, such as the J2ME from Sun, thus allowing the programmers to develop applications for mobile devices.

Despite these existing useful software development instruments, programmers of mobile device have still to deal with device-specific issues, and in particular with the use of the available wireless connection. As a matter of fact, programmers of a given application to be executed within a mobile device, have to know what kind of wireless connection is available both on the mobile device side and on the environment side (e.g. IP-based or Bluetooth networks), in order to use the correct functions to access the pervasive system.

In this paper we present a Java-based communication wrapper, called SmartTraffic, which allows programmers to seamlessly use both Bluetooth-to-Bluetooth connections, and TCP or UDP protocols over IP-to-IP or Bluetooth-to-IP connections from their applications running on devices, either mobile or not. Developers can use SmartTraffic within their J2SE or J2ME applications (MIDlets), thus focusing on the application goals, and leaving out details about how it should interact with the available wireless connection.

2. Related Works

There are several research groups that are currently dealing with middleware-related problems within pervasive applications. Among them, Bisignano et al.

0-7695-3109-1/08 $25.00 © 2008 IEEE
DOI 10.1109/CISIS.2008.129

[6] presented a middleware for mobile peer-to-peer computing on handheld devices. Its key features include the interoperability with other well-known P2P frameworks, and the ability to work even in ad-hoc configurations, where no connection to a fixed network exists.

Authors of [7] discusses a middleware that takes care of all resource discovery issues using a cluster based dynamic hash algorithm, lying on a object request broker. They also addressed knowledge usability and self-healing properties of pervasive computing, and incorporated them in the proposed middleware as services.

The research work in [8] proposes a dependable device discovery mechanism for the middleware of the applications consisting of rapidly reconfiguring mobile devices. The approach claims to offer a comprehensive solution to potential problems that can arise in highly adaptive mobile ad-hoc networks of pervasive computing environments, by means of three algorithms. The discussed algorithms operate at middleware level, instead of other previously developed ones that operate at MAC level.

A context-aware and service-oriented middleware architecture is proposed in [9], in order to provide satisfactory services to end users of mobile nodes within pervasive environments. This work shows how a suitable middleware layer is needed in such environments, due to the mobile nature of involved devices, their limited available resources and their reliability, in terms of energy, bandwidth, visualization capabilities.

The Mobile Service Manager middleware architecture [10] enables multiple interaction models suitable for mobile devices, and discuss its components. The proposed architecture dynamically composes the services bouquet for each user, thus avoiding a pervasive service to be discovered from users who cannot actually interact with such a service, possibly because their mobile device is not capable of running the service.

Another middleware for mobile computing environment is discussed in [11], where authors addressed the problems due to great heterogeneity of devices in such environment. To this end, they present a middleware for multi-client and multi-server mobile applications, taking into account the resource restrictions of mobile devices. The proposed middleware offers a transparent communication API and allows applications to divide their work amongst server and client sides.

This short review of recent research works, show the need to provide managers of pervasive systems with middleware instruments in order to take into account the heavy heterogeneity of such systems, both from the fixed part and from the mobile part. In this paper we mainly address issues related to problems programmers have to deal with when they have to take into account different wireless connections.

3. The SmartTraffic Wrapper

Our proposed Java-based communication wrapper, called SmartTraffic, allows programmers to develop applications providing them with a suitable tool to connect a mobile device with a fixed device, such as a PC, over a wireless connection. SmartTraffic also allows programmers to share the PC internet connection from mobile devices, thus transforming the PC in a proxy server. This feature can be useful to access a Bluetooth-based pervasive framework, where a mobile device "enters" the system through a PC near to it via the Bluetooth connection. The SmartTraffic comes with a number of Java packages including classes for different types of connection, and some utility class. The main packages and their use within SmartTraffic are:

- *SmartTraffic.microedition*, contains several classes which allow mobile devices to be connected with PCs using both TCP or UDP protocols over IP-to-IP or Bluetooth-to-IP connections.
- *SmartTraffic.standardedition*, contains classes for Bluetooth-to-IP connection bridging from the PC-side, carried on by means of Bluetooth USB dongles;
- *BTOperations*, contains utility classes used to search for Bluetooth devices in the neighborhood, and to start and manage the connection between mobile and fixed device over Bluetooth;

The proposed wrapper thus provide MIDlets programmers with three network access modes:

- *IP-to-IP*, by means of the Java standard socket classes. The mobile device can use GPRS or Wi-Fi connections, and both TCP and UDP protocols are implemented (Fig.1). In this case, only the *SmartTraffic.microedition* package is involved on the mobile device side;
- *Bluetooth-to-Bluetooth,* by means of classes belonging to *SmartTraffic.microedition* packages on the mobile device side (Fig. 2). Both data packet and data stream connections are implemented;

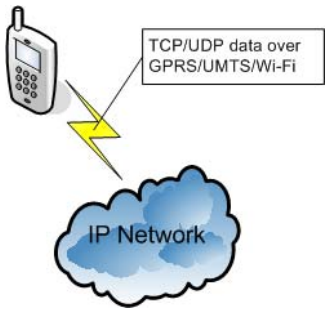


Figure 1. IP-to-IP network access mode

- *Bluetooth-to-IP*, by means of a proxy "BT2IP" server, which is equipped with both network interfaces (Fig. 3). In this case, all the three packages above described are used. In more detail, the classes of the *BTOperations* package on the mobile side allow the MIDlet to perform first a neighborhood scan for Bluetooth device and service discovery. Data to be sent is then wrapped by means of appropriate classes of the *SmartTraffic.microedition* package, thus sending them over the Bluetooth channel. Once wrapped data arrive on the server side, classes of the *SmartTraffic.standardedition* provide instruments to extract the TCP/UDP packets and to forward them on the IP channel to the rest of the network. Of course, data can follow the reverse path, that is from the BT2IP server to the mobile device, with the same operations flow.

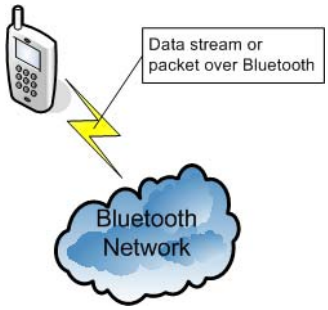


Figure 2. Bluetooth-to-Bluetooth network access mode

4. Midlet Programming with SmartTraffic

People who want to use the SmartTraffic wrapper within their applications, first have to import the appropriate package according to where applications are planned to be run.

In the following we will give some more detail about the programming using SmartTraffic, and in particular about the main classes to be used in mobile devices applications and in PCs applications.

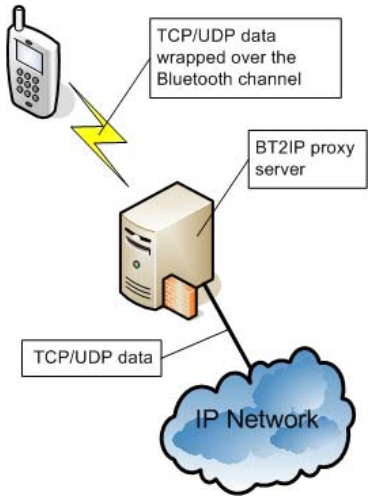


Figure 3. Bluetooth-to-IP network access mode

4.1. SmartTraffic.microedition classes

There are two main class groups: the *IP group*, and the *Bluetooth group*.

The IP group contains classes for:

- TCP over IP connections;
- TCP over Bluetooth connections;
- Combined TCP over IP and Bluetooth connections;
- UDP over IP connections;
- UDP over Bluetooth connections;
- Combined UDP over IP and Bluetooth connections.

Among this group, the classes for TCP connections are *STTCPSocket*, *STTCPOverBTSocket* and *STTCPSocketSelector*. The first one can be used to make connections over a IP connection only (GPRS/UMTS, Wi-Fi); the second can be used to make connections by means of the Bluetooth proxy server; the third class encapsulates both previous ones, thus allowing programmers to use both kind of connections. All these classes present the same software interface, which is represented by the following methods:

```
public STTCPSocket(MIDlet parent, String
  host, int port) throws IOException
public STTCPOverBTSocket(MIDlet parent,
  String host, int port) throws
  IOException
public STTCPSocketSelector(MIDlet
  parent, String host, int port) throws
  IOException
```

The parameters to be passed are: the reference to the parent application the object belongs to; the IP address or the host name of the remote machine; the port to be connected.

In a similar way, the classes of the IP group for UDP connections are *STUDPSocket*, *STUDPOverBTSocket* and *STUDPSocketSelector*. Like

above described, the first two classes have to be used to use an IP or Bluetooth only connections respectively, the third one encapsulates both previous classes. All the UDP classes present the same software interface, which is represented by the following methods:

```
public STUDPSocket(MIDlet parent, String
  host, int port) throws IOException
public STUDPOverBTSocket(MIDlet parent,
  String host, int port) throws
  IOException
public STUDPSocketSelector(MIDlet
  parent, String host, int port) throws
  IOException
```

The parameters to be passed are again: the reference to the parent application the object belongs to; the IP address or the host name of the remote machine; the port to be connected.

All the IP group classes have methods to send and receive data, and to close the connection:

```
public void send(byte[] data) throws
  IOException
public byte[] receive() throws
  IOException
public void close() throws IOException
```

4.2. The Bluetooth classes group

The two classes of the Bluetooth group have to be used to make Bluetooth connections, and they allow programmers to use both data stream and data packet communications.

The first one, *STBTStreamSocket*, is devoted to the stream communications, the second one, *STBTDatagramSocket*, allow data packet communications. Constructors of both classes are respectively:

```
public STBTStreamSocket(MIDlet parent,
  UUID uuid) throws IOException
public STBTDatagramSocket(MIDlet parent,
  UUID uuid) throws IOException
```

The parameters to be passed are the reference to the parent application the socket belongs to, and the UUID (Universal Unique IDentifier) of the needed service. The constructor start the connection process, and a list of available devices with the needed service is presented, after a Bluetooth neighborhood inquiry. At last, the user will have to choice which device has to be connected.

Of course, the Bluetooth group classes have methods to send and receive data, and to close the connection:

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

5. Examples of Midlets with SmartTraffic

In order to test the SmartTraffic operations, we created two simple trial MIDlets. The first one connects a SMTP server to send an e-mail message, the second one connects a RSS (Really Simple Syndication) server to read the corresponding XML feed.

We also created a Java 2 Micro Edition (J2ME) class, called Matlet, to remotely use Matlab and its features from mobile devices, in order to test the SmartTraffic effectiveness, even in atypical application fields for a mobile devices, such as image enhancement and complex functions analysis and plot. To this end, we also developed two MIDlet applications, both including the Matlet class. The first one is called MatImager, and the second one is called MatFunction.

5.1. MailSender

The MailSender MIDlet allows its users to connect a SMTP server and to send e-mail messages. In this case, we first need to declare a member of the STTCPSocketSelector class:

```
private STTCPSocketSelector myClient =
  null;
```

Once the object is initialized, we obtain the handles to the input and output streams:

```
myClient = new
  STTCPSocketSelector(parent,host.getStr
  ing(),
  Integer.parseInt(port.getString()));
in = myClient.getInputStream();
out = myClient.getOutputStream();
```

At this point, programming can go on in the same way as we are developing a typical network Java application. SmartTraffic will automatically route the traffic over the available connection.

Once the SMTP server has been connected, user of the MIDlet fills in the text fields with recipient's address, subject, and message body, and at last the message is forwarded. Fig. 4a and 4b show the screenshots of the MailSender MIDlet running on the J2ME Wireless Toolkit (J2WTK) mobile device emulator.

Figures 4a and 4b. The MailSender MIDlet

5.2. RSS Reader

The RSS Reader MIDlet allows its users to read XML feeds. In a similar way as above described, a STTCPSocketSelector object is first declared thus obtaining the input and output streams. Then, the RSS request string is composed and forwarded over the created output stream:

```
STTCPSocketSelector socket = new
  STTCPSocketSelector(instance,
  host,80);
InputStream = socket.getInputStream();
String request = "GET " + requestURI + "
  HTTP/1.1\r\nHost: " + host +
  "\r\n\r\n";
socket.getOutputStream().write(request.g
  etBytes());
```

The received XML document is parsed and finally the corresponding feeds are presented on the mobile device display (fig. 5a and 5b).

Figures 5a and 5b. The RSS Reader MIDlet

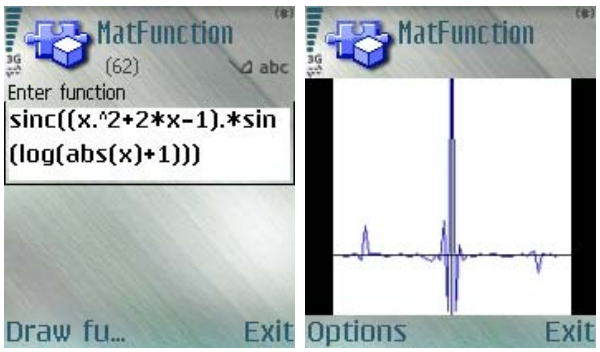
5.3. MatFunction with Matlet

MatFunction is a MIDlet application which runs on Smartphones and allows users to plot functions, being them calculated on a remote Matlab server. MatImager includes the Matlet class for remote use of Matlab features.

Once a wireless connection has been established, users write the function to be plotted in a text field. The function is then sent to the Matlab server, which parses the text and computes the values corresponding to a given range. Results are sent back to the MIDlet which only use them to plot corresponding points in a Canvas object (Fig. 6a and 6b).

It has to be noticed that no parsing or computing are carried on the Smartphone side, which are hard tasks for a reduced-resource device. As a consequence, the time taken to plot a function is mainly due to the complexity of the function itself in relation with the computing capabilities of the machine on which computations take physically place.

The overhead due to the use of a mobile device is given by the data transfer time from and to the mobile device.



Figures 6a and 6b: a sample plot with MatFunction on a Nokia N70 Smartphone

5.4. MatImager with Matlet

MatImager is a MIDlet which runs on Smartphones and allows users to do operations on images, provided that there is a corresponding script on the Matlab server side. MatImager includes the Matlet class for remote invocation of Matlab scripts (Fig. 7a and 7b). The steps leading to the processed image are:

1. send the image to be processed to the Matlab server;
2. remotely invoke a script on the image;
3. download the processed image.

This way, algorithms do not have to be rewritten in Java for execution on mobile devices, since they are executed on the server side as Matlab scripts. Furthermore, users can apply even complex algorithms on their images, since they do not require any computing power from mobile devices. Actually, mobile devices can be used for other tasks waiting for the server to end the requested image processing.

Figures 7a and 7b: MatImager on a Nokia N70 Smartphone for image thresholding

6. Conclusions and Future Work

A communication wrapper to help programmers in developing Java applications for both personal mobile devices and PCs in order to use different wireless connections available in a pervasive environment has been presented, along with some program example by using the proposed solution. The sample MIDlets we implemented for our tests show that SmartTraffic make the programmers work easier, due to the very small programming overhead. The implementation work showed the actual advantage of the wrapper, both for its technical soundness and for its effective management of resources.

SmartTraffic allow pervasive service providers to use more the Bluetooth wireless technology for the interaction with their users, taking even more advantage of its large diffusion in personal mobile devices, of its low power consumption, and of its increasing bit rate.

Future work will investigate the development of an integrated middleware for pervasive environment composition, in order to provide programmers with further instruments to be included in their applications. The main goal is to allow them to focus its attention on service provision related problems, with no care about the actual communication and execution available frameworks.

References

[1] S. Sorce, A. Augello, A. Santangelo, G. Pilato, A. Gentile, A. Genco, S. Gaglio, "A Multimodal Guide for the Augmented Campus", in proc. of the ACM-SIGUCCS 2007 Fall Conference, 7-10 oct. 2007, Orlando, FL (USA), pp. 325-331

[2] Raptis D., Tselios N., Avouris N., "Context-based design of mobile applications for museums: a survey of existing practices", Proc. of the 7th ACM International Conference on Human-Computer Interaction with Mobile Devices & Services, Salzburg, Austria 2005, pp: 153-160

[3] Henrysson A., M. Ollila, M. Billinghurst, "Mobile phone based AR scene assembly", Proc. of the 4th International Conference on Mobile and Ubiquitous Multimedia, Christchurch, New Zealand 2005, pp. 95-102

[4] Price, S.; Summers, R., "Mobile Healthcare in the Home Environment", Proc. of 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society EMBS '06, New York, NY, Aug. 2006, pp. 6446-6448

[5] S. Sorce, F. Cinquegrani, S. Anzalone, D. Caccìa, A. Gentile and A. Genco, "A Dynamic System for Personal Communications: the Opportunistic Chat", in proc. of the IEEE International Conference on Intelligent Pervasive Computing (IPC-07), 11-13 oct 2007, Jeju Island, Korea, pp. 307-312, DOI: 10.1109/IPC.2007.9.

[6] Bisignano, M.; Di Modica, G.; Tomarchio, O.; "JMobiPeer: a middleware for mobile peer-to-peer computing in MANETs", 25th IEEE International Conference on Distributed Computing Systems Workshops, 6-10 June 2005 Page(s):785 – 791

[7] Sharmin, M.; Ahmed, S.; Ahamed, S.I.; "MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing) for Mobile Devices of Pervasive Computing Environments", Third International Conference on Information Technology: New Generations, ITNG 2006, 10-12 April 2006, Page(s):306 - 313

[8] Ahamed, S.I.; Zulkernine, M.; Anamanamuri, S.; "A dependable device discovery approach for pervasive computing middleware", First International Conference on Availability, Reliability and Security, ARES 2006, 20-22 April 2006, DOI: 10.1109/ARES.2006.5

[9] Huigui Su; Xiufen Fu; Zhiqing Li; Qunsheng Yang; Shaohua Teng; "A Service-oriented Middleware for Pervasive Computing Environments", 1st International Symposium on Pervasive Computing and Applications, 3-5 Aug. 2006, Page(s):36 – 41

[10] Mahmoud, Q.H.; Al-Masri, E.; "MSM: A Middleware Architecture for Enhancing Interaction with Mobile Services", 2nd International Symposium on Wireless Pervasive Computing, ISWPC '07, 5-7 Feb. 2007.

[11] Rocha, B.P.S.; Rezende, C.G.; Loureiro, A.A.R.; "Middleware for multi-client and multi-server mobile applications", 2nd International Symposium on Wireless Pervasive Computing, ISWPC '07, 5-7 Feb. 2007.