Agent Based Adaptive Management of Non-Homogeneous Connectivity Resources

Flavio Esposito, Francesco Chiti, Romano Fantacci
Università di Firenze
Dipartimento di Elettronica e Telecomunicazioni,
Via di S.Marta 3, I-50139 Firenze, Italy
Email: flavio.esposito@ee.oulu.fi
chiti@lenst.det.unifi.it
fantacci@lenst.det.unifi.it

Simo Hosio, Junzhao Sun
MediaTeam, University of Oulu
P. O. Box 4500 FIN-90014 University of Oulu,
Oulu, Finland
Email: simo.hosio@ee.oulu.fi
jzsun@ee.oulu.fi.

Abstract—In this paper, a middleware architecture that enables transparent inter-operability between two different wireless networks is presented. In particular, access to the communication channel is maintained by adaptively switching between GPRS (General Packet Radio Service) and Bluetooth interfaces, both established automatically with the assistance of an RFID tag. The access channels are continuously monitored, allowing the device to switch to the Bluetooth interface, whenever this lower-cost alternative is available. A preliminary field trial has been set up, which has demonstrated the effectiveness of the approach in supporting seamless handover procedures within a heterogeneous network as envisioned for future wireless communications systems.

I. INTRODUCTION

Recently, the vision of a wireless communication paradigm in which mobile phones can resort to multiple heterogeneous access networks has arisen. A major requirement to the realization of a pervasive communications system is to provide seamless interconnection amongst multiple access channels. To fulfill this task, particular care is needed in the design of resource management procedures. Consequently, there is a growing interest in the development of new middleware for future hybrid mobile communication systems.

Two main issues must be addressed when a mobile device equipped with multiple interfaces attempts to enter a heterogeneous network. The first is to establish a connection with the available access resources (devices), selecting the best radio interface at that moment. The second is to preserve the connection if conditions change, e.g., one of the non-homogeneous network elements becomes unavailable.

A great deal of research has been carried out on heterogeneous network integration. Stemm and Katz [1] introduced the concept of vertical handoff, in reference to handoffs across two cells belonging to different network segments, such as Wireless Local Area Network and General Packet Radio Service. In fact, the present proposal consider the vertical handoff approach as a starting point. A detailed overview on the most significant issues involved in the handoff mechanism has been considered by K. Pahlavan [2].

In particular, a vertical handover policy based on both the knowledge of the higher level parameters related to the transport and application layers and the knowledge of the available access networks’ characteristics is devised by Calvagna and Di Modica [3]. A policy enabled handoffs across heterogeneous wireless networks is presented by Wang [4] in order to minimize the user interaction. Instead, a general approach to the location management topic (update and delivery procedures) and handoff management is presented by Akyildiz [5].

All the previous handover architectures analyzed, are common based on a switch with an overhanging infrastructure. Therefore different resources of different technologies are dynamically connected but all of them are beneath the same main resource, for instance, internet. The architecture of this paper introduce the possibility to skip the infrastructure to access to the resource in the heterogenous network.

To achieve the seamless integration of the wireless non-homogeneous network in this new context we assure that in our application during the vertical handover there is no worthy waste of information. The software presented aims to provide the switch in a mobile phone environment, which has much less computational capacity than the other works analyzed. The present proposal is strictly related to more advanced release of the architecture of the CAPnet (Context-Aware and Pervasive networks) system, providing seamless services for mobile users [17], within which a novel middleware for mobile phone has been implemented.

The agents for connection management accept the packets coming from the two different channels and route the stream on the right one. The best connection where to route has chosen priorly.

This paper is organized as follows: Section II specifies the reference hybrid network model together with the context-aware middleware: in details, a scenario, the related architecture and the involved interfaces are presented. The design and the implementation issue are discussed in Section III. Section IV highlights the results and the performance measurement of an experimental validation endowed with a simple but significant application. Finally, the last section discusses the main contributions provided by the proposed approach.
II. OVERALL ARCHITECTURE

The optimal maintenance of the channel in a context-aware environment where a mobile phone is equipped with multiple wireless networks has been the issue of some of the previously proposed architectures [11,12]. The first paper mentioned imposes more attention to describe the connectivity management in a WLAN and 3G context. The second, more generally describes the requirement of an architecture for an heterogeneous network. In this section we look at the general problems of a non-homogeneous network from the application point of view clarifying the main problems and the software solution for each. Abreast has been suggested the solution for one specific scenario.

Before explaining in details the architecture, the terms channel and connection need to be explained [15]: A channel is intended as a logical link between physical application components (usually client and server) running on separate network devices. A connection is the physical correspondence of channel, that is an end-to-end link between two network interfaces of the two separate peer hosts.

Each channel uses a specific connection to transfer a specific data flow between specific network interfaces of local and remote hosts.

A. Problems for the Optimal Maintenance Channel

The problem of effective internetworking between two different wireless systems, as explained in Fig. 1 When linking two different architectures in order to reach optimal maintenance of the channel, two major issues need to be addressed: Network Creation and Connection Switching.

The former consists in selecting the best available interface, while the latter is related to maintaining the users choice within the session duration. There are two possible cases in which the application is allowed to switch: when a better connection is available or when a current connection is lost. In both cases the connection should be automatically and seamlessly switched by the application.

In the filed of ubiquitous network a great deal of studies have been conducted searching for the best algorithm to select the optimal connection [8,10].

However, in the proposed scenario it is supposed that the GPRS connection is always available and the Bluetooth connection is selected whenever it is possible to provide better user satisfaction and lower cost. As to this aspect, it has been considered the cost of the alternative component of the non-homogeneous network following the Always Best Connected (ABC) approach. Gustafsson [9] explains that through the ABC concept a user application is allowed to be always on by using the devices and access technologies that best suit QoS requirements. This can be reached by means of combining infrastructured access technologies, such as Bluetooth and WLAN, with cellular systems to provide an enhanced user experience for 2.5G, 3G and beyond.

It is worth noticing that, while the Bluetooth connection is free from any kind of fee, the access to GPRS system is subjected to a subscription with a public telephone company. Therefore, the proposed architecture is able to provide a mobile user the faster and cheaper connection too, in a transparent way.

To accomplish the previous requirements a new software is to be developed on a mobile phone. Before analyzing the proposed architecture we must understand the needs of our device. In relation to the ISO-OSI model the device must have the capabilities of joining the Bluetooth stack with the GPRS architecture by way of the session layer the Bluetooth stack with the GPRS architecture.

Referring to the ISO-OSI model the device might be able to join the Bluetooth stack with the GPRS architecture, as explained in Fig. 1 and it is reached by means of a novel software at the application layer.

B. Proposed Solution

The proposed solution allows a context-aware application, i.e., the mobile user might recognize the context, where context means any kind of information that characterizes the status of an [7]. An entity could be a place, an object or a person considered relevant to the interaction between user and application. Some examples of context information are the users location, the time, the users profile, the service available on the mobile phone, etc.

To this purpose we used a prototype of a mobile phone with modified hardware, where a Radio Frequency Identifier reader has been embedded in the circuitry, realized within the MIMOSA project [14]. This permits to acquire the connection information moving through the environment.

To this existing platform a novel connectivity management agent has been realized adding it to the CAPnet middleware [8], as it is shown in Fig. 2.

![Fig. 1. Overview Architecture: new pieces in the session layer are added to connect the two different stacks: in dark gray the new component.](image-url)
The CAPNET middleware is located below the application layer and about the layer of existing technologies. It supports different devices in collecting data which comes from different distributing software components. In particular, we have been interested in developing one of the Connectivity Management (CM) component. The CM structure includes both the CM Agent (CMA) and the Socket Engine. The CMA plays a crucial role for the adaptive network connectivity management, as it maintains the channels life cycle, where it means evaluation after a channel opening or switching whether the event connection does not persist anymore. The evaluation is done when the user makes an upload or download request.

The Socket Engine is the main responsible of the channel creation: through the Symbian OS classes it manages the channels request. It needs of all the connection parameters such as the addresses, the type of the connection and the ports, some of which are stored in the Radio Frequency Identifier (RFID).

Without expound in the architectures of the Bluetooth [16], the GPRS and the RFID [15] the Fig. 3 present a basic scheme of what has been our work.

The overall architecture of CAPnet middleware [8] is not reported, since we present only the novel component related to the underlying layers.

The CMA, taking care of the channel opening, has the following particular events to process: connection request and connection switching. For the former, the CMA manages the connection request creating an interface with two sockets shown in Fig. 3: in particular, the link between the RFCOMM Bluetooth layer with the application layer and the link between the TCP with the application layer. The RFCOMM is an interface that allows an application to treat a Bluetooth link in a similar way as if it were communicating over a serial port.

Another CMAs task is to manage the connection that is to switch. The event of "connection not anymore available" is processed by CMA, interrupting the data transfer on the current connection and resuming it on the other channel previously created by the Socket Engine.

On the other hand, the Socket Engine is the real responsible of opening a connection and creating a channel within the whole heterogeneous network. The Engine under request from CMA opens every socket component, providing that the considered member is available. To permit the data transferring the CMA require two different object from the engine, that is opening two dedicate channels: a "Writer" and a "Reader".

The server side software needs to accept the connection depending on the users request, using the Writer or the Reader according to the user location: Is in fact the user who decides when and where to move the mobile device.

III. DESIGN AND IMPLEMENTATION

To provide the functionalities above described in order to set up and maintain a stable connection, the software layer to be developed needs to be focused on three different aspects: establish a GPRS connection, establish a Bluetooth connection, allow the handover.

From a software point of view, the channel creation implies the connection establishing of two separate connections: in Fig. 4 it is shown the related time chart diagram. In particular, the involved agents, like the Mobile Application, the CMA Client, the Socket Engine and the CMA Server are pointed out, jointly with the GPRS and Bluetooth channels set up primitives.

Moreover, for the sake of clarity, the class diagram is been reported in Fig. 6. In particular, the class RFIDHandle which allow the user to get the parameters of the sockets from the RFID tag. the RFIDHandle class also handle the menu commands of the application. As shown in Fig. 6, it is directly linked with the CMA which calls the socket engine. If the server’s CMA accept the connection request, the channel is able to write and read. Then, the socket Engine sends an acknowledge to the application who shows a message of connection established. To create the connection the follow-
ing Symbian OS classes must be involved: RSocketServ and RSocket. The first class creates a session with the socket server, thus providing a channel with the socket server.

![Diagram](image.png)

**Fig. 4.** GPRS and Bluetooth channel creation: the connectivity management agent establishes both connections through the SocketEngine.

The RSocket class is a sub-session of the RSocketServ session, hence requiring a previously established socket server session. Its allowing the socket creation and reading or writing from a socket. It has also reference to access to the RFCOMM Bluetooth layer. Both the RFIDHandle and the AppView classes are classes which allows to show on the mobile device screen all the operations. The second functionality which has to be provided by the CM is the switch between the two different connections previously established. An RFIDHandle object makes a data exchange request therefore a ClientCMA's object checks if the Bluetooth connection is available with a method of the class RSocket. When a connection is down a socketWriter object is called: if the command request by the RFIDHandle was an upload, the data starts to be transferred on the listening server CMA, otherwise a SocketWriter is used for sending a signal recognizable from the ServerCMA class which can start to send data.

From the users point of view, the channel can be obtained after invoking the CMA's OpenChannel method. Immediately after the user request of channel, the device is put in a wait status and a "wait note" is shown on a mobile phone. The approach to request service by touching object has been widely discussed by Riekki [18]. When the RFID tag is reached the CMA can open both the connections taking the Bluetooth and GPRS informations (address and port) from the RFID.

The proposed middleware has to be dismantled in two different sides: the server and the client. On the mobile phone side, the application is implemented on a Symbian operating system, which is widely used open platform for mobile devices. The server side software has been implemented in Java language.

![Diagram](image.png)

**Fig. 5.** Channel switching: connectivity management agent check if the Bluetooth is still available and if is not enable the data transfer on the GPRS channel.

To clarify the implementation step of the mentioned software we had reported also a class diagram of the client side. The core of the implementation is provided by the four active classes: ClientCMA, CSocketsEngine, CSocketsReader, CSocketsWriter.

The CMA manage the channels request opening and closing it when necessary, check if the Bluetooth connection is available every time a request of data transfer has been made and store the data to transfer in the reader or in the writer in a buffer. It reports any change in a status variable and received data to the User Interface for display. The CSocketsEngine class creates the real session with the socket server. Performs any required name solution, creates and connects the socket to be used in communication with the server for both the connections. Moreover handles any feedback from the socket reader (CSocketsReader) and writer (CSocketsWriter). The CSocketsReader issues asynchronous read requests to the socket server and reports any errors through an interface. The Writer class issues asynchronous requests on demand and written reports errors on the same interface used by the reader.
A Team LAN. The mobile hardware, through the embedded RFID reader, communicates with the engine. A Bluetooth USB dongle is installed on a server side and a Java software is running on it. The mobile device is equipped with a Bluetooth interface, and the software based on the proposed architecture, as well as the Octopus network (along a project owned by the Municipality of Oulu) [19] have been used to access to the GPRS.

### A. File transfer

A case study to investigate the performance of the vertical handover solution has been realized by considering a text file transfer, in which the channel switching time is negligible with respect to the channel set-up time. As a result, the continuity of data transferring within a session is guaranteed.

Table 1 is reported as part of a log file created by the mobile device during the data transfer from the client to the server. The impact of the different size file transfer is analyzed, recognizing that the packet delivery ratio (PDR) is unaffected by the number of byte sent.

*Fig. 7. Packet Latency during the downgrade switch for different buffer size.*

**TABLE I**

<table>
<thead>
<tr>
<th>Socket and Buffer Size Used</th>
<th>Packet Delivery Ratio</th>
<th>Transmission Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPRS 40 byte</td>
<td>1</td>
<td>190916</td>
</tr>
<tr>
<td>Bluetooth 40 byte</td>
<td>1</td>
<td>106021</td>
</tr>
<tr>
<td>Handover 40 byte</td>
<td>0.99992</td>
<td>157193</td>
</tr>
<tr>
<td>GPRS 80 byte</td>
<td>1</td>
<td>1188896</td>
</tr>
<tr>
<td>Bluetooth 80 byte</td>
<td>1</td>
<td>53151</td>
</tr>
<tr>
<td>Handover 80 byte</td>
<td>0.99984</td>
<td>111025</td>
</tr>
<tr>
<td>GPRS 160 byte</td>
<td>1</td>
<td>18902</td>
</tr>
<tr>
<td>Bluetooth 160 byte</td>
<td>1</td>
<td>126170</td>
</tr>
<tr>
<td>Handover 160 byte</td>
<td>0.99968</td>
<td>72229</td>
</tr>
<tr>
<td>GPRS 250 byte</td>
<td>1</td>
<td>194866</td>
</tr>
<tr>
<td>Bluetooth 250 byte</td>
<td>1</td>
<td>19982</td>
</tr>
<tr>
<td>Handover 250 byte</td>
<td>0.9995</td>
<td>90637</td>
</tr>
<tr>
<td>GPRS 500 byte</td>
<td>1</td>
<td>190658</td>
</tr>
<tr>
<td>Bluetooth 500 byte</td>
<td>1</td>
<td>11402</td>
</tr>
<tr>
<td>Handover 500 byte</td>
<td>0.999</td>
<td>71525</td>
</tr>
</tbody>
</table>

The data concerning the packet latency in the downgrade switch has been plotted in Fig. 7.

Half a megabyte of the text data has been uploaded from the mobile phone to the server side moving outside of the Bluetooth range. Decreasing the buffer size the latency increases the agent call methods. In the meantime, the Packet Loss Rate increases because the maximum amount of data which can be lost (buffer size) has been reduced.

However, a maximum efficiency implies a strong increase of delay and a decrease in bandwidth. The agent calls the Symbian OS class every five hundreds bytes instead of every forty, eighty or more and the consequence is the different slope (bandwidth) showed in the five different curves.

To notice is the variation of the slope strongly visible in every curve especially the faster. The GPRS speed does not decrease as the Bluetooth when we decrease the buffer size.

It is possible to see the little delay in the five curves reported during the switch even though there is no synchronization during the handover. The reason for that is the structure of the baseband layer in the second layer of the Bluetooth stack. The Bluetooth baseband is capable of supporting one asynchronous connectionless (ACL) data link and up to three synchronous connection-oriented (SCO) links. In the tries we used the ACL. The ACL link includes packet retransmission so when the user goes out from the Bluetooth range the transfer gets slow until the socket is completely lost.

Last but not least, observation in regards to the graphics is the absence of tract with a null differential.
In future applications the limit of a non unitary PDR for each buffer size will be solved with a synchronization. During that time no data will be transmitted and therefore the variation of the bandwidth will have a tract with a value equal to zero; in the case analyzed, the time of switching is negligible.

The previous work done in a CAPNET project leads us to believe that the total amount of data transmission time plus the switching time, will be greatly reduced.

It is hard to predict instead the optimum solution for a particular case in which the user is located in a zone where the power of the Bluetooth signal requires too many handover in a very small amount of time. In this case, it is probably better to leave the inferior connection rather than continuously switching.

It should be recognized that in the tested proposals, as the time of the switching is negligible, the speed that the user has to utilize for entering and coming out from the Bluetooth is irrelevant. The same methodology would not apply the synchronization mechanism.

B. Experiment scenario

The last step of our work has been to implement a scenario realized for the validation of the architecture proposed. Fig. 8 shows a windows of the software implemented on a server side for the test. The Java server side software receives data from the channel and it shows through two forms: one for each connection created. When the channel has been created, the user can "upload" some text: a well and a priori-defined text starts to pass from the mobile phone to the server side through the Bluetooth channel; during the transfer the mobile user moves his phone out of the range frequency of the Bluetooth. Then, the automatic transfer continues over the GPRS connection and the next letter appears on the form dedicated to the GPRS.

![Fig. 8. Experiments and validation result: the form appear on a server side during the text transfer: GPRS always available; Bluetooth availability is conditioned form the context.](image)

V. CONCLUSION

In this paper, a novel software component for the CAPnet middleware has been proposed, allowing the management of the connectivity also with mobile phone in a heterogeneous network environment. In particular, the horizontal layer architecture has been improved by linking two different stacks and creating a Symbian OS application well-suited for that environment.

The architecture provides the mobile with the requested applications, introducing a switching between two stacks not linked with a common overlooking technology.

The proposed approach has been tested in a practical case study as a file transfer, claiming for a data synchronization mechanism in order to recover the data lost during the time switching.

It was also shown that within the overall handover duration, the passage from the Bluetooth to the GPRS systems is negligible in terms of data transmitted in a time unit.

REFERENCES