Beyond the Web: Mobile WAP-based Management

[1]
Luca Deri
Centro Serra, University of Pisa

Mobility, tightening competition and changes in system administrator behaviour present new challenges in the management world. In particular, the need to constantly monitor services and rapidly identify and fix problems is becoming increasingly important. The Internet has made possible to offer a comprehensive range of management services and applications accessible over the web. Nevertheless, in the near future a new and improved way to manage systems will be done by means of mobile phones.
This paper shows how existing web-based solutions can be enhanced to support the new generation of media phones based on WAP (Wireless Application Protocol) technology. Although the specific focus of this work is on management solutions, the results are general and applicable to many other types of existing web-based applications. Finally, some implementation details and experiences drawn from adding a WAP interface to an existing web-based management application are described.

Keywords: Network Management, Internet Technologies, WAP.

1. Introduction

In mid 90s, some people [1, 2, 4] pioneered the integration of Internet technologies into the management world. Their vision was clear: give people the ability to manage networks and systems inexpensively over the web without having to use complex tools difficult to install, configure and use. The web demonstrated that using proxy applications [6] it was possible to abstract complex technologies such as those used in the management field and map them on simple languages and protocols such as those used by the web. The use of proxies enabled developers to enrich existing applications giving them a web appearance without having to rewrite or deeply modify the design of such applications. Today we can state that their vision was correct as almost every management system offers a web console from which operators have access to advanced management facilities in a user friendly way using a standard web browser.
Once the web-based management was born, it was clear that this was not the end but just the beginning of a transition towards a new management paradigm. The Internet has made it possible to offer a comprehensive range of management applications and services inexpensively over the web. However, one channel for improved, ubiquitous, dynamic services will be the mobile phone. This statement is motivated by several reasons:

- There are already more mobile subscribers than web clients and their number is rapidly increasing.
- Users tend to be mobile and the mobile phone allows people to be located wherever they are. This is especially true in the management field where administrators may be moving between physically separate locations in order to perform their duties.
- Administrators need to be promptly informed about problems and service interruptions. This can be easily done notifying the problem to the administrator's mobile phone rather than using email that requires users to be connected to a network and have the mail client open.
- User experiences drawn from PDA (Personal Digital Assistant) usage, have shown that users are willing to accept a simplified user interface in favour of mobility. Being able to check network status and perform actions as problem arise is much more convenient using a mobile phone than a desktop computer or even a laptop. Mobile phones are easier to use, smaller and lighter and they are instant, ubiquitous, and truly portable.

A few years ago, the author decided to simplify an existing web-based network management platform [5] he developed in order to make it usable from mobile devices. The outcome of the project

[1.] Centro Serra, University of Pisa, Lungarno Pacinotti 43, Pisa, Italy, deri@ntop.org, http://www.ntop.org/.

[11, 12] has been web-based management console running on a PDA interconnected to the Internet via the embedded GSM phone. It was clear that this solution was rather primitive because this PDA was a sort of mini sub-notebook not able to truly exploit any GSM features such as SMS (Short Messaging System) that could have been used to asynchronously notify the user when specific condition happened.
With the advent of the new generation of media phones based on WAP technology, it is now possible to take fully advantage of mobility with no compromises contrary to what happened with the previous project. This paper introduces WAP-based management, a novel approach to network and system management tailored for mobile users. The goal is to show how existing management applications can profitably take advantage of WAP technology for creating a new generation of management applications suitable for mobile network and system administrators. Therefore, this work should be viewed as a supplement and not substitute for existing network management systems and applications, which addresses the needs of certain types of mobile end-users.
The following sections describe some experiments that the author did in the pre-WAP age using conventional GSM phones, then introduce the WAP, highlights how existing web-based applications can take advantage of it, and show how an existing web-based traffic monitor has been extended in order to provide a WAP interface.

2. Towards Mobile Network Management: from SMS to WAP

In mid-1998 the Centro Serra, responsible for providing network services to the whole University of Pisa, needed an application for monitoring Internet services provided by Serra. The author developed a simple monitoring application named *Sniffy* mostly written using the Perl language [14]. This application was responsible to control services such as DNS, Internet news/mail, FTP/WWW, and dialup access. Sniffy was basically a very big infinite loop containing several checks for detecting the status of each monitored service. Some CGI scripts were used to present on the web the status of the services monitored by Sniffy. The management console was notified by Sniffy via SNMP traps whenever one of the checks failed. System administrators could define an action to be performed for each alarm received by the management console. Typical actions were email notification, and execution of short shell scripts for fixing simple problems (e.g. a service was not available because the corresponding daemon died unexpectedly). Although Sniffy was working reliably, some of its users requested the author to extend it in order to notify them about the most severe problems. As during the weekend Serra computer systems were mostly unattended, the only way to notify the system administrators was via SMS using a GSM phone. Early experiments using some HTML to SMS gateways freely available on the Internet demonstrated that the SMS solution was very promising. The only drawback of this approach was the monodirectional communication from Sniffy to the administrators via SMS. In some situations it would have been useful for administrators to perform some actions via SMS. For this reason the author decided to connect a GSM cellular modem [13] to the serial port of the workstation where Sniffy was running. A GSM modem is like a conventional modem: it just lacks of the phone socket that has been replaced by an antenna and a GSM SIM (Subscriber Interface Module) card reader. The modem can be controlled via a terminal emulator using AT commands. Each command is made up of three elements: the prefix (`AT`), the command body, and a termination character (usually the carriage return is used). For instance the command `AT+CPIN="<pin number>"` is used to enter the SIM PIN number. As the modem could be shared among several process both local and remote, the author decide to write a *gsmd* responsible to control the access to it. The gsmd opens the serial port to which the modem is attached and listens for requests on a specified UDP port. Gsmd clients register with the gsmd for receiving incoming SMSs, and use the gsmd for sending SMS. For this purpose the gsmd was able to understand a simple language made of three simple commands:

- `REGISTER <SMS prefix>`
  This command instructs the gsmd to deliver to the client all the received SMS that start with the specified prefix. For instance if a client sends the gsmd `"REGISTER CMD"`, the gsmd delivers it all the received SMS messages whose text starts with `CMD`. Please note that multiple

clients can register the same prefix, so in this case the gsmd delivers the received message to multiple clients.

- `SEND SMS <destination phone number> <message>`
  The client requests the gsmd to send the specified message to the destination number.
- `DEREGISTER`
  The client cancels all the previous `REGISTER` requests.

As SMS messages were rather short (up to 160 characters), a UDP packet was large enough to fit every message. In this architecture the gsmd was always active, whereas clients could be started as needed. Sniffy was then extended by adding a new Perl script named *Sniffy-GSM*. This script was both a parser for a simple language and a gsmd client. Each command started with CMD prefix, so that the gsmd could route to Sniffy-GSM all incoming SMS starting with such prefix. In this way, the network administrators could interact with Sniffy using this simple language and not only the other way round. For instance, suppose that Sniffy detected that on host `mailserver.unipi.it` the partition `/var/spool/mqueue` was 95% full. This problem might indicate that either sendmail was not working properly or that for some reason the mail queue needed to be flushed. In case Sniffy would have send to the network administrator a SMS message containing the following text `"mailserver:/var/spool/mqueue 95% full: 1=restart 2=flush"`. The network administrator could either decide not to perform any action, or ask Sniffy-GSM to perform either action 1 (restart sendmail) or 2 (flush queued messages). If the administrator decided to restart sendmail on host mailserver, it should have replied to the previous message with the following text `"CMD mailserver EXEC 1"`. As far as security was concerned, Sniffy-GSM did not process any incoming SMS message originated by a phone number not included in a list of numbers allowed to talk with it.

The use of a GSM modem made Sniffy more powerful that the previous version. Unfortunately this approach had still some rough edges:

- Error Prone
  The administrator should have replied to Sniffy-GSM with a message in the specified format.
- Limited Interaction
  This solution was perfect for reporting problems and performing simple predefined actions, but unsuitable for more complex tasks due to the limited SMS message length.
- User Friendliness
  Management information was presented in a simple textual fashion not as rich as the one presented by a web browser.

Seen all these limitations, the author decided to explore WAP, a very promising and standard technology that started to become available at the end of the Sniffy project. The following sections describe what is WAP and how it has been profitably used for monitoring the campus network of the University of Pisa.

3. Introduction to WAP

WAP is an open protocol for wireless applications developed by the WAP forum, a non-profit industry association to which all industry participants can belong. WAP [15] specifies an application framework and network protocols for wireless devices such as mobile telephones, pagers, and PDAs. While most of the technology developed for the Internet has been designed for desktop and large computers with medium to high bandwidth, wireless data networks present a more constrained communication environment compared to wired networks. The WAP specifications extend and leverage mobile networking technologies (such as digital data networking standards) and Internet technologies (such as XML, URLs, scripting, and various content formats).

Being the WAP programming model similar to the WWW programming model, it allows developers to leverage existing tools (e.g., web servers, XML tools, etc.) and take advantage of a familiar programming model, and proven architecture. In addition, the WAP forum has introduced optimisation and extensions in order to exploit peculiar characteristics of the wireless environment taking advantage of the lessons learned using the web.

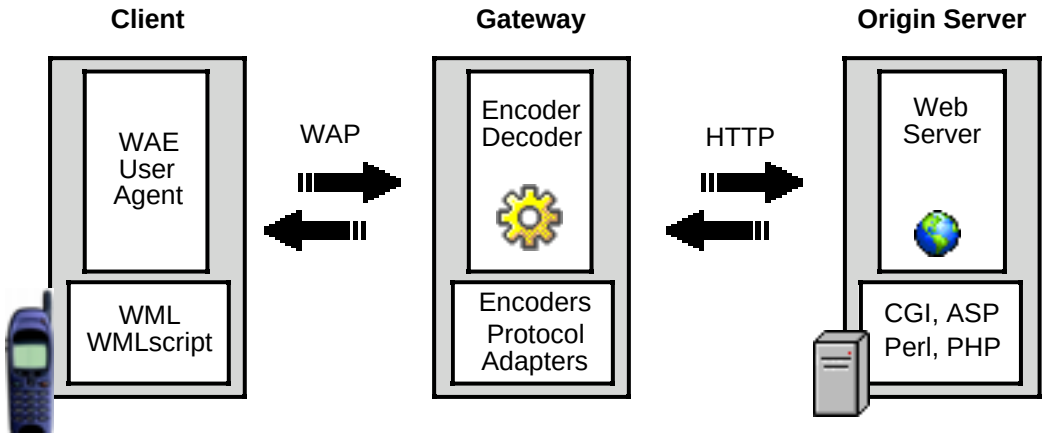

Figure 1. WAE (Wireless Application Environment) Logical Model

All content is specified in formats that are similar to the standard Internet formats. Content is transported using standard protocols in the WWW domain and an optimised HTTP-like protocol in the wireless domain. The WAE architecture allows all contents and services to be hosted on standard Web origin servers that can incorporate proven technologies (e.g. CGI). All content is located using standard URLs. WAE enhances some of the WWW standards in ways that reflect the device and network characteristics adding support for mobile network services and paying attention to memory and CPU processing constraints that are found in mobile terminals. This means that WAE provides support for low bandwidth and high latency networks.

WAE key elements include:

- WAE User Agents
  Each WAP device includes built-in browsers that display content to the end-user. Browsers interpret network content referenced by a standard URL. WAE includes user agents for the two primary standard contents: encoded WML (Wireless Markup Language) [16] similar to HTML and compiled WMLScript (Wireless Markup Language Script) [17] similar to JavaScript. Contrary to what happens in the Internet where HTTP transports unencoded, plain text HTML, WML is transmitted in encoded form whereas WMLScript is compiled in bytecode format.
- Content Generators
  Applications (or services) on origin servers (e.g., CGI scripts) produce standard content formats in response to requests from user agents in the mobile terminal. Typical content generators are servers used in the WWW today that have been enhanced to serve WML information. In this case, the conversion from HTML to WML can be either done by the server or by means of a gateway.
- Standard Content Encoding
  Standard content encoding includes compressed encoding for WML, bytecode encoding for WMLScript, standard image formats, and multi-part container format.
- Wireless Telephony Applications (WTA)
  A collection of telephony specific extensions for call and feature control mechanisms that provide authors and end-users advanced mobile network services.

As it happens with the web, a user agent on the terminal typically initiates a request for content that is served by the origin server. In order to return content that can be handled by the terminal, WTA includes mechanisms that allow origin servers to deliver generated content without a terminal's request. The user agent characteristics are communicated to the server via standard capability negotiation mechanisms that allows applications on the origin server to determine characteristics of the mobile terminal device including global device characteristics as WML version supported, WMLScript version supported, floating-point support, and supported image formats. Whenever appropriate, an end-to-end secure connection can be established using WTLS (Wireless Transport Layer Security Specification) [18] that stands to WTP (Wireless Transaction Protocol) [19] as HTTPS stands to HTTP.

4. Why WAP?

Author's interest for WAP comes from the fact that it is the first open effort towards bringing Internet-like services to mobile devices. Several PDA manufacturers have tried over the past few years to incorporate lightweight browsers into their handheld devices in order to give access to Internet services. Although they tried hard, the result are devices that sport several limitations compared to desktop computers and laptops but that do not take great advantage of the mobile environment. Talking about HTML for instance, many PDA manufacturers decided to support a subset of HTML because the full standard was far too complex for handheld devices. The WAP consortium instead acknowledged this and defined WML that is specifically tailored for mobile devices.

Another point to take into account is that web services are accessed using client applications that pull data out of a server. Due to this design solution, the server has no means to push data to the client when specific condition arise and the client has not other way other than doing a periodic active poll or to implement tricks [8] to partially overcome this limitation. Fortunately WAE defines a standard mechanism for the server to push data to the client hence enable development of mobile applications such as online stock trading, news broadcast or system and network management where it is very important to notify the administrator as soon as specific situations occur.

For the above reasons WAP looks to the author as a good technology for implementing mobile remote services that will help administrators to meet their targets by creating better management services that can be put side by side to the ones based on the web. The idea is not to replace existing web-based management applications with WAP solutions, but rather to provide a simple path for accessing management services using mobile devices.

The following section covers the design and implementation of a WAP-based traffic monitoring application based on an existing web-based application. This is to demonstrate that WAP-based management is feasible and adds great value with limited development effort.

5. Ntop: a Web/WAP-based Traffic Monitoring Application

Ntop [3] is an open-source [13] web-based traffic monitoring and network intrusion detection (NIDS) application, available under the GNU public licence. This statement does not just mean that ntop's source code is freely available on the Internet, but also that many requirements came directly from people who deployed ntop. The author designed the first version of ntop and then accommodated new requirements and extensions to the original architecture, strongly influenced by the *Webbin* [4] architecture. As depicted in the following figure, the ntop architecture is divided in two parts: network traffic analysis and data representation.

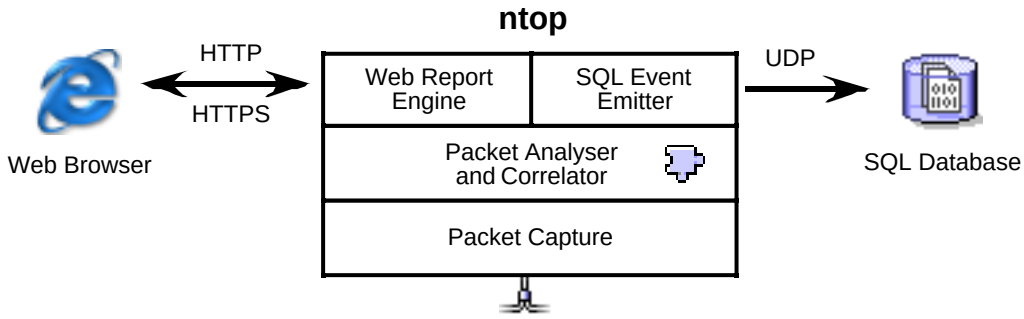

Figure 2. ntop Architecture

The ntop engine captures packets from the network and builds an internal data representation of the actual network traffic status. In addition, the engine verifies whether each captured packet, matches one or more NIDS rules and keeps track of the security violations identified. Ntop contains an internal HTTP/HTTPS engine that allows web browsers to see ntop as a web server. Network traffic reports are generated on the fly by the embedded web report engine. Application extensibility is guaranteed via plugins that are loaded at application startup. A ntop plugin is a software component with a well-defined interface that is not statically linked to the application but it is loaded at runtime.

As shown in figure 1, a simple way to give an existing web application a WAP appearance is by means of a WAP gateway. The gateway takes care of the conversion of HTML into WML, and maps on the fly WAP requests into HTTP requests. Using this approach it is rather easy to move from the web to WAP. Unfortunately, the ntop project experience has shown that this transition is not completely smooth. In fact, the conversion from HTML to WML can present several challenges due to the different nature of the devices involved:

- Hyperlinks
  HTML pages usually contain links to other pages. As HTTP can retrieve only one page, everytime the user selects a link pointing to an external page, the web browser has to contact the web server and retrieve the requested page. Contrary to HTML, WML pages are grouped in decks. This means that a link to card in the same deck does not require the mobile device to contact the server. For this reason, it is necessary to efficiently group in the same deck several pages related to similar information in order to decrease the number of communications with the server. In general, HTML to WML gateways split the HTML page into several cards of the same deck without making any optimisation that could place in the same deck different related pages.
- Tables
  Like HTML, WML supports tables. Unfortunately mobile devices have a rather small screen usually with vertical scrolling only. This means that tables with three or more columns are not rendered nicely on a WAP device. For this reason, it is necessary to either avoid tables or to use small ones because otherwise the gateway would split them in several smaller tables. These tables are difficult to read as the HTTP server has no way to send the gateway hints about how to divide them unless using non standard HTML tags or exploiting comments.
- Images
  WML supports images. Common image formats such as GIF and JPEG have been dropped in favour of WBMP (Wireless BitMaP) that are much smaller and more suitable for low-resolution devices. Supposing that the gateway supports on the fly image conversion and scaling, for better results HTML pages should include, if any, tiny icons able to fit on a small screen.
- Forms

Due to the way HTML forms work, each time the user submits a form, the web browser sends the request to the web server that process it and sends back a response page. Although this mechanism works in WAE, WML is much more powerful of HTML in this respect. For instance, WML allows people to specify the form field format and compute locally simple forms using WMLScript without the need to contact the server every time a form is submitted. As these features are not present in HTML, the gateway cannot take advantage of them.

Seen all the limitations of automatic HTML to WML conversion, the author decided to write a new plugin to give ntop a native WAP interface. This plugin implements a WML report facility similar to the existing ntop web interface. The WML view is much simpler than the one offered by HTML and contains only the essential information. As ntop is a self-founded project, for the purpose of reducing development costs and because in Italy WAP is not yet fully supported by the mobile carriers, the author decided to use a WAP simulator part of the Nokia WAP Toolkit [9].

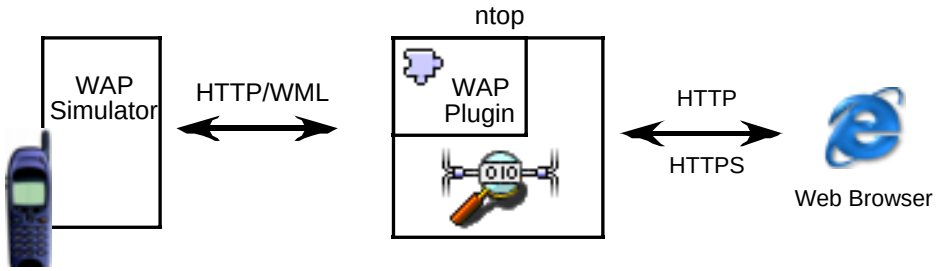

Figure 3. Ntop for WAP: Architecture

The simulator is a Java application that reproduces a real WAP phone that connects to ntop via HTTP. For connecting ntop to a real mobile WAP device, it is necessary to download the Nokia WAP gateway able to push WML pages to the wireless network. In this case the architecture shown in the previous figure will be modified as follows: a real WAP-enabled phone will be used instead of the simulator, and the gateway will communicate with ntop over HTTP as the simulator does. The simulator connects via HTTP to the URL `http://<ntop host>:<ntop port>/plugins/WAPPlugin` that is the navigation entry point for navigating through network traffic information. All the hyperlinks generated by the WAP plugin are contained under the `/plugins/WAPPlugin` subtree so that simulator requests are always handled by the plugin and do not interfere with other ntop components. As the plugin is loaded dynamically at ntop startup, it has full access to ntop internal data so it allows WAP users to navigate through traffic information with no restrictions with respect to ntop web users. The use of plugins has several advantages over monolithic applications:

- Cleaner code design: the boundaries defined by the plugin interface help the developer specify the information and the services (WAP-specific code is not mixed with ntop).
- Simple tailoring: users can selectively activate functionality at runtime only when needed.
- Low cost and resource usage: memory and disk space are saved because there is no need to load/use all the components but just the ones really needed. General performance is improved because only the necessary resources are used.
- Easy extensibility: developers can easily add new functionality to ntop by creating new plugins without having to the hack the ntop kernel code.

The following figure shows how ntop for WAP looks from the simulator.

Figure 4. Ntop for WAP: Simulator Screenshots

Users can scroll vertically using the arrow keys, whereas hyperlinks are selected by using the middle key. The two menus on the bottom of the screen are accessed by using the keys below the menu name. The WAP interface allows seeing relevant network activities such as top senders/receivers, view traffic statistics and see alarms produced by the ntop IDS. In addition, it is possible to see basic information about each relevant host as shown in the last picture of the previous figure. The use of WML decks allowed to greatly reduce the number of communications between ntop and the simulator, and allowed large pages to be divided in subpages accessible using the bottom screen menus. As WML is rather similar to HTML, the WAP plugin has been mostly developed reusing the web report code. It is worth to note that although the WAP plugin uses HTTP such as the web interface, the HTTP header is slightly different as WML uses an encoding schema different than the one used by HTML.

In the ntop WAP architecture, the WAP gateway is needed just for pushing ntop WML pages from the LAN to the mobile network and vice-versa but not for HTML to WML conversion. Beside obvious speed enhancements due to the lack of HTML to WML conversion, this solution allowed the author to greatly optimise the decks and produce pages that render nicely on small displays. However, even supposing to have an almost perfect and performant HTML to WML gateway, the ntop experience drawn from user feedback has taught the author that the goal of a WAP application is usually different from the equivalent web-based application, at least in the management field. The difference is due to the different nature of users. A mobile user does not pretend to have a full network view, but uses its mobile WAP device for receiving unsolicited notifications about network problems and performing some basic operations when specific situations occur. The same user attached to a web terminal wants to have a richer view of the network and be able to perform complex operations unlikely to do with a GSM phone while in the street. This means that the network traffic view offered by ntop to a WAP user has to be simple, should contain only the necessary information, must be fast to load, and should render nicely on a small screen.

6. Final Remarks

Network managers are becoming increasingly mobile and this trend will require extending existing applications in order to make them accessible to mobile users. This paper has shown how existing web-based applications can be extended in order to take advantage of emerging standard technologies such as SMS and WAP. Two main directions different approaches for adding a WAP interface to existing web-based applications have been analysed and compared. The paper also presented the design and implementation of ntop for WAP, the first WAP/Web-based traffic monitoring application freely available on the Internet. The results and experiences drawn from the ntop project are general and applicable to generic web-based applications and not only to the field of network and system management.

7. Availability

Ntop for both Unix and Win32 is distributed under the GPL2 licence and can be downloaded free of charge from both the ntop home page (http://www.ntop.org/) and other mirrors on the Internet.

Some Unix distributions including but not limited to FreeBSD and Linux, come with ntop preinstalled.

8. Acknowledgements

The authors would like to thank all the ntop users and early adopters who deeply influenced the design of the overall architecture with all their comments and suggestions.

9. References

1. G. Bogler, *Internet - New inspiration for Telecommunications Management Network*, Proceedings of IS&N'97, Como, Italy, May 1997.
2. B. Bruins, Some Experiences with Emerging Management Technologies, *Simple Times 4(3)*, 1996.
3. L. Deri and S.Suin, Effective Traffic Measurement using ntop, *IEEE Communications Magazine*, May 2000.
4. L. Deri and A. Weder, *Webbin' CMIP*, Poster proceeding of the 4th International WWW Conference, Darmstadt, Germany, April 1995.
5. L. Deri, *Surfin' Network Management Applications Across the Web*, Proceedings of 2nd Int. IEEE Workshop on System and Network Management, 1996.
6. E. Gamma, and other, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
7. M. Jander, *Web-based Management: Welcome to the Revolution*, Data Communications, 1996.
8. J.P. Martin-Flatin, *Push vs. Pull in Web-Based Network Management*, EPLF Technical Report SSC/1998/022, July 1998.
9. Nokia, *Nokia WAP Toolkit version 1.2*, http://www.forum.nokia.com/, September 1999.
10. E. Raymond, *The Cathedral & the Bazaar*, O'Reilly & Associates, ISBN 1-56592-724-9, 1999.
11. J. Reilly, *The World Wide Web and Programming Future Broadband Network and Service Management Application*s, Seventh bi-annual Nordic Workshop on Programming Environment Research (NWPER'96), Aalborg Denmark, May 1996.
12. J. Reilly, P. Niska, L. Deri and D. Gantenbein, *Enabling Mobile Network Managers*, Proceedings of the 6th Int. WWW Conference, Santa Clara, CA, April 1997.
13. Siemens, *Cellular Engine M20/M20 Terminal: Technical Description*, Version 5, March 1999.
14. L. Wall and others, *Programming Perl*, O'Reilly & Associates, 2nd Edition, ISBN 1-56592-149-6, September 1996.
15. WAP Forum, *Wireless Application Environment Overview*, http://www.wapforum.org/, April 1998.
16. WAP Forum, *Wireless Markup Language*, http://www.wapforum.org/, April 1998.
17. WAP Forum, *Wireless Markup Language Script*, http://www.wapforum.org/, April 1998.
18. WAP Forum,*Wireless Transport Layer Security Specification*, http://www.wapforum.org/, April 1998.
19. WAP Forum, *Wireless Transaction Protoco*l, http://www.wapforum.org/, April 1998.

10.Biography

Luca Deri is currently sharing his time between Finsiel S.p.A. and the Centro Serra at the University of Pisa. He received his Ph.D. in Computer Science with a thesis on Software Components from the University of Berne in 1997. He previously worked as research scientist at the IBM Zurich Research Laboratory, and as research fellow at the University College of London. His professional interests include network management, software components and object-oriented technology. His home page is http://jake.unipi.it/~deri/.