



**Research Center**

NRC-TR-2006-004

## **Personal Website on a Mobile Phone**

Johan Wikman, Ferenc Dosa and Mikko Tarkiainen

Nokia Research Center Helsinki

<http://research.nokia.com>

May 24, 2006

### **Abstract:**

Current mobile phones are increasingly becoming like general purpose computers and we are reaching the point when mobile phones can even act as servers. In this paper we briefly describe how the Apache httpd web server was ported to the Symbian/S60 platform and how access from the Internet to a web server on a mobile phone is arranged. We further explore what implications the presence of a generally accessible web server on a mobile phone might have and how the concept of a website changes when it resides on a personal device.

### **Index Terms:**

personal website

mobsite

web server

HTTP

mobile phone

## 1 INTRODUCTION

Most high-end mobile phones of today are equipped with web browsers that allow users to browse the web. In general, mobile phones — in particular so-called smartphones — are increasingly becoming like general purpose computers and functionality that traditionally have been available only on regular computers is being migrated to them.

However, so far mobile phones have not really been used as servers — in particular as web servers — although they in terms of processing power and the amount of available memory are on the same level as, or even surpassing, the servers that were used when the web was in its infancy in the early nineties.

The idea of running a web server on a mobile phone or another similar device is not a particularly novel. For instance, [13] and [14] present an architecture for a web server running on Microsoft PocketPC and source code for a web server for Symbian is even available for downloading at [8]. The assumption seems to be, though, that a web server on a mobile device necessarily must be simple and constrained when compared with web servers on regular computers.

In the beginning of 2004 we started a project at Nokia Research Center with the aim of exploring what implications it would have if it were possible to host a website on a mobile phone and if that website were addressable in and accessible from the Internet, using the operator networks of today.

## 2 WEB SERVER

We set out with the intention of putting a full-fledged web server, that implements the full HTTP protocol, on a mobile phone. While the creation of a simple web server is not particularly challenging, the creation of a feature-rich web server of production quality is. Due to the modest size of the project, creating a web server from scratch was not a realistic alternative.

The remaining option was to port an existing web server from another platform. We decided to choose Apache httpd[3], not only because it is the most popular web server but also since it typically is associated with large websites that run on big servers and that are administered by professionals or semi-professionals. Being able to show that something not intended for mobile phones nonetheless can run on such appealed to us as it would indicate that the general perception of the capability of a mobile phone may need to be adjusted.

### 2.1 Apache httpd

The Apache httpd web server<sup>1</sup> has quite a few appealing properties:

- It is the most popular web server in the world, which means that there are many people who know how to operate it.
- It is robust, commercial-grade, feature-rich and designed for easy porting.
- It is modular, so unneeded features can easily be left out, thus reducing the memory footprint of the server.
- It is Open Source and there is a large active developer community.

A slightly more intangible benefit for selecting Apache was that, should the Symbian port prove to work well and gain popularity, it is reasonable to expect that it eventually will be included in the mainstream release of Apache.

<sup>1</sup>Hereafter we will refer to the Apache httpd web server simply as Apache.

### 2.2 Porting Apache to Symbian

Apache is designed for easy porting and in principle it is operating system neutral. However, in practice there is still a certain bias towards Unix and Unix like systems, such as Windows. Symbian is quite different, but fortunately Symbian does have a so called Posix layer than provides many, but not all, of the typical functions found on a Unix system.

Apache is implemented on top of *APR* — *Apache Portable Runtime* — which is a software library whose intention is to provide a predictable and consistent interface to an underlying platform-specific implementation. The porting friendliness of Apache, including APR, stems from two sources: firstly they are modular in general, which allows you to remove functionality that is not needed, and, secondly, it is easy to replace a common implementation of some functionality with a platform specific implementation.

Both Apache and APR have rather small operating system specific parts that in principle have to be implemented for the operating system at hand. In the Symbian case, all of the operating system specific parts did not have to be re-implemented since a rather extensive fraction of the Unix specific implementation worked directly on top of the Posix layer of Symbian.

### 2.3 Scripting

Apache modules are written in C or in Symbian's case also in C++. While that makes it possible to access all functionality of Symbian and in our case S60 it requires a certain amount of diligence. Fortunately, while we were working with Apache, another group at Nokia Research Center were in the process of porting Python[6] to S60[1]. We first created a custom module for integrating the two and as the results were very promising we proceeded to port *mod\_python*[4] — an Apache module that closely integrates Python with Apache — to Symbian as well. The porting was successful so now content generating scripts can be written in Python, which obviously lowers the threshold for developing web applications by a fair amount.

### 2.4 Not Complete

The port is currently by no means complete but, nonetheless, a rather large number of Apache modules build unmodified on Symbian. For instance, the modules for supporting *WebDAV*[7], that allow the file system of the mobile phone to be mounted on a PC, work without modifications.

The most significant limitation is that the dynamic loading of modules is not supported but all modules have to be linked with the main binary. This is caused by the fact that Symbian does not support having static data in DLLs, the workaround of which could not have been done without extensive modifications. Fortunately this is a problem that has gone away by itself, as Symbian from version 9.0 onwards supports static data in DLLs.

## 3 CONNECTIVITY

Being able to run a full-fledged web server on a mobile phone is rather interesting in itself but still not much more of a quirk unless it also can be accessed from the Internet. We experimented for some time with Bluetooth PAN (Personal Area Network) but soon concluded that it supports too narrow a set of usecases to be interesting.

### 3.1 Cellular Connectivity

As our goal was to make a web server on a mobile phone accessible from any regular browser on the Internet any time, there was

really no alternative but to provide the connectivity using the cellular network. However, that is not quite as straightforward as it may seem as operators typically employ Network Address Translation and firewalls that effectively prevent a device on the inside of the firewall to be reached from the outside.

The restriction can be circumvented by arranging for the connection to be created by the device on the inside of the firewall and then deliver traffic to the device over that connection. That is, there is no need to be able to reach the mobile phone directly. In order to create that outbound connection the phone obviously needs something to connect to and for that we created a custom gateway that runs on a computer on the Internet.

To make it appear as if the web server on the mobile phone is directly addressable we employ a generic DNS mapping like:

\*.at.openlaboratory.net ⇒ Gateway IP Address

Thus, when someone browses to, for instance, `http://john.doe.at.openlaboratory.net` and the browser does a domain name lookup the name will resolve to the IP address of our gateway. However, when the browser subsequently opens a connection to the gateway and submits the request, the header will still contain the original host name `http://john.doe.at.openlaboratory.net`.

From the header the gateway knows that the request is intended for the mobile phone known as *john.doe* and if it is connected, the request can be sent forward over the connection opened by the phone. On the phone the request can then be delivered to the actual web server.

Thus, to all parties concerned it seems that there is a normal HTTP connection between the browser and the web server on the mobile phone. However, as so often is the case, the devil is in the details and there were numerous issues that had to be solved before the connection between the browser and the web server became reliable. For a more in-depth analysis of the connectivity issues please refer to [2].

## 4 WEBSITE ON A MOBILE PHONE

From the fact that it is possible to run a web server on a mobile phone does not follow that it would be meaningful to have any kind of websites on mobile phones. The processing power, the amount of memory and the available network bandwidth of a mobile phone will, for the foreseeable future, be inferior to that of regular computers.

Furthermore, since mobile phones still occasionally are turned off, a mobile website—or *mobsite*, as we call it—will not always be accessible. In addition, the cost for the cellular connectivity is likely to remain rather high for still some time. Consequently, it is probably a mistake to simply move what could be characterized as a “regular” website to a mobile phone.

However, a phone equipped with a web server is very different from a phone without one and a website on a mobile phone is very different from a regular website, and if these specific aspects are taken into account, a number of new possibilities emerge. In the following sections we will explore some of these in greater detail.

## 5 A PHONE WITH A WEB SERVER IS DIFFERENT

### 5.1 Web UI

Even though the user interfaces of mobile phones are quite user friendly, they are still rather constrained compared with the possibilities offered by the big screens and proper keyboards of regular PCs. Considering that many users of mobile phones have access to PCs for a significant fraction of the day, at work and at home, it is a restriction that also in that context the only UI available for a mobile phone is the native one.

If a web interface is created for the core applications of a mobile phone, it would mean that whenever you have access to a PC—any PC, not necessarily your own—with Internet connectivity you would be able to access the functionality of your phone using a proper keyboard and a big display.

Reading and answering SMSs could take place from within a regular Internet browser and you could even arrange for an RSS feed for received messages. It could even turn out that some users would more frequently use the web interface than the native UI of the phone.

A web interface to the an application on a mobile phone could obviously also be used remotely. Every mobile phone user is likely to have forgotten his phone at home at some time or another. With a mobile web server on the phone it is easy to browse to it remotely and check, for instance, whether someone has called or sent an SMS, and even answer SMSs.

### 5.2 No Need for Synchronization

For many people the possibility to synchronize the data of their mobile phone with a PC is important. One example is the calendar that, unless the data can be synchronized in both directions, can either be used on the phone or on the PC, but not on both.

If the calendar on the phone can be accessed and manipulated from a web browser, then the primary location of the calendar could be on the phone and there would be no need for syncing. When at a PC, you access the calendar from a web browser, when away from a PC, you access it using the native interface.

### 5.3 Web Services

If you have a web server running on the mobile phone then you can have web service providers as well. That opens up new possibilities for making mobile phones even more general purpose than they currently are.

Referring to the calendar example of the previous subsection, an electronic calendar does not really become useful before it can be made accessible to other people and it can be used for setting up meetings. In practice that has meant that you need to have a centralized server where the calendar data is held, which has largely implied a corporate environment.

With a web service interface to the calendar it would be easy to create a distributed calendar application without the presence of any centralized server. Consequently, the calendar application on a mobile phone could become quite useful for ordinary users and perhaps even sufficient as such for small enterprises.

### 5.4 New Messaging Concepts

Currently the means for communicating with the phone are basically defined by standards, phone manufacturers and operators. In practice, you can call a phone or send it an SMS or MMS message.

Granted, there are, for instance, implementations of *instant messaging* and *push-to-talk*, but common to these is that they are clients to a server somewhere else and in order to use the functionality you need to have the appropriate client installed.

With a generic HTTP connectivity to the phone it is possible to create new messaging concepts without a-priori standardization or support from the operator. For instance, we have created a web application that allows you to send a message to the phone from a regular web browser. The message is either shown directly on the screen as an instant message or placed in the inbox of the phone along with the “real” SMSs and MMSs.

From a cost perspective this is also quite interesting. Provided you already have 2.5G/3G access — for instance, for browsing the web — functionality similar to SMS can be provided essentially for

free as the amount of data in a textual message is negligible compared to the amount of data transferred during a regular browsing session.

### 5.5 Pull Instead of Push

To disseminate any data from a mobile phone currently requires that the phone owner actively, for instance, sends an SMS or an MMS, or uploads a picture to a website.

With a web server on the mobile phone we can change that. For instance, instead of sending MMSs to your friends when you are on a vacation, you could provide them with a pointer to a picture gallery on your phone that they can browse at their leisure. It obviously would also be possible to allow people browsing the gallery to leave comments and write blog entries. However, a public web-gallery on the mobile phone is likely to be a realistic option only as long as the number of people accessing it remains fairly low.

## 6 A WEBSITE ON A PHONE IS DIFFERENT

### 6.1 Personal

A regular website can be personal but the personal content is largely explicitly created and has to be manually uploaded, because the website does not reside on a personal device. Contrary to that, a mobile phone is personal and both contains and collects personal data that implicitly and immediately are available. Contact entries, logs of dialled and received calls, sent and received SMSs, taken pictures and video clips are all data that easily can be used for (semi)automatically creating a mobile home page.

If you wanted to reach the similar kind of personal level on a regular web site you would have to continuously upload data. It would be possible, but at some point it becomes easier to simply share the data from the location where it is already stored.

### 6.2 Interactive

Currently the content a website returns depends largely on the input parameters of the request and sometimes on the identity of the one browsing. The content may be dynamic to its nature but there is no explicit human participation in the content generation. When the website is personal and resides on a device that is carried along for most of the time, the situation changes, as it is possible to involve the owner of the phone in the content generation.

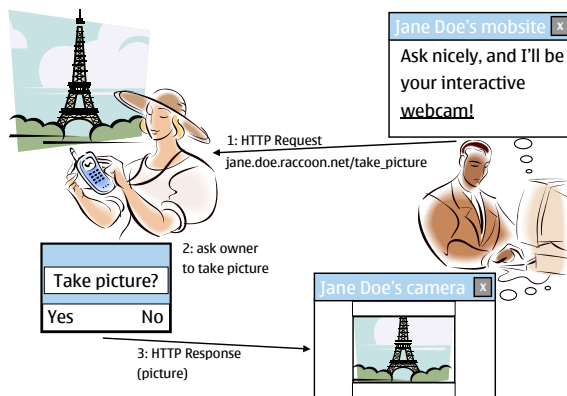


Figure 1: Interactive website.

For instance, Figure 1 describes a web application we have created that, when activated, beeps and displays a dialog box with

the question *Take Picture?* and two buttons *Yes* and *No*. If *Yes* is pressed, then a picture is taken, converted to a JPG and returned. That is, the fact that the website resides on a device that the owner most of the time carries with him means that websites can now become interactive.

Conceptually this is quite interesting as well, because effectively the content does not exist before it is asked for, it is generated at the spot, and it no longer exists on the website once it has been returned. Traditional search engines that rely upon a *crawl-index-search* approach will not work with this kind of content that does not exist before it is asked for.

### 6.3 Dynamic

The website on a mobile phone can be dynamic in the sense that the content can change depending on implicit state changes on the phone. For instance, the background color of a page or even the style sheet could change depending on the mode — general, meeting, silent, etc. — of the phone.

This possibility could also be utilized so that when you intend to contact someone, you would first browse to his or her mobile home page. This would, of course, not necessarily require explicit actions but occur automatically when you select an entry from the contacts on your phone.

Based on the state of the phone the default way for contacting — call, SMS, email, etc. — could be presented on the page. It would even be possible to take into account the identity of the one browsing. Thus, it could be the one being contacted who would decide the way the contacting is made, not the one who is contacting.

### 6.4 Context and Location Dependent

In traditional websites the geographical location of the actual web-server lacks meaning since it never changes and it has no impact on the returned content. With a mobsite this is no longer the case as the returned content may depend on the geographical location and surrounding context.

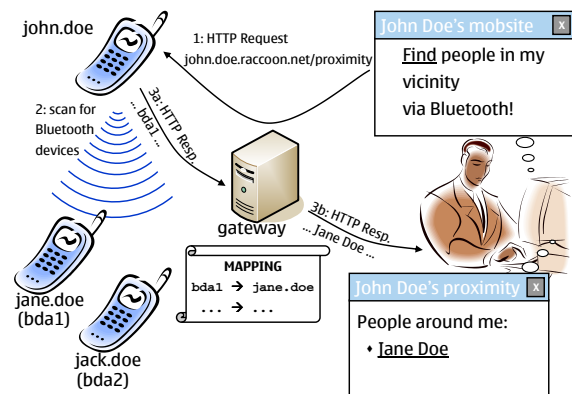


Figure 2: Mobile hopping.

An example of this is a concept we call *mobsite hopping*, which is illustrated in Figure 2. On the mobsite there is a link that when followed activates a certain web application. That web application does a Bluetooth[15] *device discovery*, which means that the BDA — *Bluetooth Device Address* — of all devices in the proximity (that are equipped with Bluetooth) will be found out.

After the discovery, the web application generates an HTML page where the found BDAs are embedded inside a list as HTML comments like

```
<!--bda:00123756851d-->
```

When the page containing those comments passes through the gateway on its way to the browser, the gateway translates *known* BDAs into the equivalent URLs, for instance,

```
<a href="http://jane.doe.at...">Jane Doe</a> .
```

The gateway knows which URL corresponds to a particular BDA because when the owner of a mobile website enables mobsite hopping, the BDA of the mobile phone is submitted to the gateway via a simple web service interface. If he disables it, then the mapping is simply removed.

An important point to realize is that Bluetooth is *only* used for finding devices in the proximity. Once the list of nearby mobsites is returned to the browser and the user clicks on a link, the other mobsite is accessed directly and not via the mobsite from which the list was obtained. That is, the access of the second mobsite is not dependent on the first mobsite *remaining* in the proximity of the second.

Currently we only use this for finding other mobsites, but there is no reason why it could not be extended to stationary websites as well. Shops, restaurants, or any other businesses could have a Bluetooth beacon for no other reason than to make their stationary website visible via mobile personal ones.

## 6.5 Offline

With ordinary websites administrators go to great lengths in order to ensure that the website stays up. With mobile websites, the assumption can no longer be that the website is always available, as people occasionally simply turn off their mobile phones. Instead of considering this as evidence that a mobile phone is unsuitable for hosting a website, we view this as an inherent property of mobile websites that simply has to be taken into account.

If the mobile phone were directly addressable it would be difficult to deal with this in an end-user friendly way, since, when the phone is turned off, it would simply disappear and web browsers would report it in some browser specific way. The owner of the mobile phone could not provide any additional information.

The presence of the gateway provides means for dealing with this issue in a more end-user friendly way. Since the requests intended for some mobile phone will always go through the gateway, it is possible to explicitly report that the mobsite is offline. For instance, currently when turning off the webserver on the phone, the user is asked whether he wants to leave an *out-of-site* message. If he chooses to, the text he enters is uploaded to the gateway and if someone tries to browse to the mobile website while it is offline, the gateway will display the customized message.

## 7 CHALLENGES

That mobile phones are capable of running a web server and that it is possible to make a mobile website visible in the operator networks of today is only a part of the story. There are numerous challenges that need to be addressed before mobsites can become ubiquitous.

### 7.1 Cost

Currently many operators still charge by the byte and the rates are quite high. However, the situation is not quite as bleak as might initially be assumed:

- The prices for data transfer are decreasing and there already are operators offering flatrates, although they may not yet have thought of the implications brought by websites being hosted on mobile phones.

- Mobile phones are being equipped with WLAN, which means that for much of the time, the wireless cost would be zero.
- The presence of the gateway means that there is a point where caching can be employed. For instance, in the case of a picture gallery it can easily be arranged so that a picture is downloaded at most once.

Another aspect is that even if access control is employed at the mobile website in order to control who can access content and thus cause costs, it is still possible for a hostile user to circumvent that by simply over and over again trying and being denied access. If access control is employed only in the mobile phone there is no alternative but to transfer the request there and that alone already causes cost.

To prevent cost from being generated, the access control must take place before the traffic enters the cellular connection. So, from a cost perspective, as long as there are no true cellular flatrates, the access control should take place already at the gateway.

### 7.2 Battery Consumption

The battery consumption is a problematic issue. Currently all activities that increase the battery consumption — for instance, browsing the web or talking — are quite explicit for phone user and thus will not surprise him.

If a web server, or any other server for that matter, is running on the mobile phone, an external party can increase the battery consumption without it being obvious to the owner of the phone. To prevent surprises, some means for controlling the access based on the state of the battery may be needed. But as in the cost case, any activity for reducing the battery consumption should take place before a traffic reaches the phone.

Again it turns out to be beneficial to have a gateway between the browser and the phone. For instance, the gateway could allow anonymous browsing to the mobile website only when the battery is fully charged or the phone is attached to a charger.

### 7.3 Security and Privacy

Because the phone contains a fair amount of personal information, it is important that the phone owner can control who can access what. That is a challenge because traditional means such as accounts and passwords may not work that well in the context of a personal website.

Applying access control in a straightforward manner could easily imply that a phone user would become an administrator who explicitly manages accounts and passwords and who answers “support requests” when someone has forgotten his. And from an other perspective, the amount of passwords a person would have to remember could easily grow unwieldy if mobile websites become popular.

There is also a more subtle aspect that stems from the fact that mobile websites are context dependent. Traditionally, access control on websites is essentially a function of the identity of the one browsing and the content he tries to access. With mobile websites that alone will not always suffice when deciding whether or not to allow access.

Consider for instance the earlier mentioned case of being able to find out other mobile websites in the surrounding. You may want to allow that depending not only on the identity of the one browsing but also on the context and even your mood. That can be arranged by utilizing the interactive aspects of the personal website. When someone tries to access some link, traditional means for access control can first be applied. Then, since the identity of the person at that point is known, it is possible to show a dialog with the question *Allow access to X for Y*. That is, for some content the phone owner

can explicitly decide on a case by case basis who can access it and when.

#### 7.4 Backup and Recovery

We envision that the mobile phone could in many cases turn into the primary device of a person. All relevant personal data are stored there and only there, and accessed from other devices using a browser.

But mobile phones are lost and stolen, and, as other hardware, also break down at times. If all your personal data is on the device, it will be imperative that it is not irrevocably lost in cases like that. Thus, efficient means for backing up and recovering the data of a mobile website are needed.

### 8 IMPLICATIONS

According to Netcraft's web server survey[10] there were in February 2006 over 70M websites of which some 39M are considered active. In Q4/05 Nokia alone shipped over 9M smartphones[11] and the growth rate of shipped smartphones is much higher than the growth rate of websites. Consequently, it is not unrealistic to estimate that within a few years the number of smartphones will greatly surpass the number of websites.

Should smartphones be equipped with web servers, it is fair to say that at some point the majority of websites will reside on mobile devices. It is quite clear that a large number of websites, such as Amazon and eBay to take but a few examples, will not move anywhere, but for many people the easiest way to get a homepage would simply be to acquire a smartphone.

### 9 CONCLUSION

It is possible to run a full-fledged web server on the mobile phones of today and, with the help of an intermediate gateway, it is possible to make it appear as if a website on a mobile phone would be directly accessible from the Internet.

A phone with a web server is different from a phone without one because it opens up a range of possibilities that hitherto have been impossible. A website on a mobile phone is different from a regular website because it resides on a personal device that consequently contains a lot of personal data and the "administrator" is always nearby.

The amount of smartphones is rapidly increasing and within the foreseeable future their number will greatly surpass the number of regular websites. If every smartphone is equipped with a web server, then quickly the majority of websites will reside on mobile phones. That is bound to have some impact on how mobile phones are perceived and how the web evolves.

All code has been open sourced and is available from SourceForge[16].

### REFERENCES

- [1] Nokia Research Center. Python for s60. <http://opensource.nokia.com/projects/pythonfors60/index.html>.
- [2] Wikman Johan Dosa Ferenc. Providing http access to web servers running on mobile phones. <http://research.nokia.com/tr/NRC-TR-2006-005.pdf>.
- [3] Apache Software Foundation. Apache httpd web server. <http://httpd.apache.org>.
- [4] Apache Software Foundation. Apache/python integration. <http://www.modpython.org>.
- [5] Apache Software Foundation. Apr. <http://apr.apache.org/>.
- [6] Python Software Foundation. Python. <http://www.python.org>.

- [7] IETF WEBDAV Working Group. World wide web distributed authoring and versioning. <http://www.ics.uci.edu/ejw/authoring>.
- [8] McAleely John. Smallserv: a simple http server for symbian os. <http://www.symbian.com/developer/techlib/apps/smallserv.html>.
- [9] Symbian Ltd. Symbian os. <http://www.symbian.com>.
- [10] Netcraft. Web server survey. [http://news.netcraft.com/archives/web\\_server\\_survey.html](http://news.netcraft.com/archives/web_server_survey.html).
- [11] Nokia. Quarterly and annual information. <http://www.nokia.com/A4100008>.
- [12] Nokia. S60. <http://www.s60.com>.
- [13] Nicoloudis Nicholas Pratistha Dennis. .net compact framework mobile web server architecture. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/NETCFMA.asp>.
- [14] Nicoloudis Nicholas Pratistha Dennis and Cuse Simon. A micro-services framework on mobile devices. 2003.
- [15] Bluetooth SIG. Bluetooth. <http://www.bluetooth.com/bluetooth>.
- [16] SourceForge. Mobile web server. <http://sourceforge.net/projects/raccoon>.